JOSE
Internet-Draft
Updates: 7516 (if approved)
Intended status: Standards Track
Expires: 3 June 2026

T. Reddy
Nokia
H. Tschofenig
H-BRS
A. Banerjee
Nokia
O. Steele
Tradeverifyd
M. Jones
Self-Issued Consulting

30 November 2025

Use of Hybrid Public Key Encryption (HPKE) with JSON Web Encryption (JWE)
draft-ietf-jose-hpke-encrypt-15

Abstract

This specification defines how to use Hybrid Public Key Encryption (HPKE) with JSON Web Encryption (JWE). HPKE enables public key encryption of arbitrary-sized plaintexts to a recipient's public key, and provides security against adaptive chosen ciphertext attacks. This specification chooses a specific subset of the HPKE features to use with JWE.

This specification updates RFC 7516 (JWE) to enable use of the Integrated Encryption Key Establishment Mode.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://ietf-wg-jose.github.io/draft-ietf-jose-hpke-encrypt/draft-ietf-jose-hpke-encrypt.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-jose-hpke-encrypt/.

Discussion of this document takes place on the jose Working Group mailing list (mailto:jose@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/jose/. Subscribe at https://www.ietf.org/mailman/listinfo/jose/.

Source for this draft and an issue tracker can be found at https://github.com/ietf-wg-jose/draft-ietf-jose-hpke-encrypt.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Hybrid Public Key Encryption (HPKE) [I-D.ietf-hpke-hpke] is a public key encryption (PKE) scheme that provides encryption of arbitrary-sized plaintexts to a recipient's public key. This specification enables JSON Web Encryption (JWE) [RFC7516] to leverage HPKE, bringing support for HPKE encryption and KEMs to JWE, and the possibility of utilizing future HPKE algorithms.

2. 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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Terminology

This specification uses the following abbreviations and terms:

* Content Encryption Key (CEK), Header Parameter, and JOSE Header, as defined in [RFC7516].

* Hybrid Public Key Encryption (HPKE), as defined in [I-D.ietf-hpke-hpke].

* pkR is the public key of the recipient, as defined in [I-D.ietf-hpke-hpke].

* skR is the private key of the recipient, as defined in [I-D.ietf-hpke-hpke].

* Key Encapsulation Mechanism (KEM), per [I-D.ietf-hpke-hpke].

* Key Derivation Function (KDF), per [I-D.ietf-hpke-hpke].

* Authenticated Encryption with Associated Data (AEAD); see [I-D.ietf-hpke-hpke] and [RFC7516].

* Additional Authenticated Data (AAD); see [I-D.ietf-hpke-hpke] and [RFC7516].

This specification defines the following terms:

Key Management Mode A method of determining whether a Content Encryption Key (CEK) value is used and, if so, what CEK value to use. Each method used for making these determinations uses a specific Key Management Mode. Key Management Modes employed by this specification are Key Encryption, Key Wrapping, Direct Key Agreement, Key Agreement with Key Wrapping, Direct Encryption, and Integrated Encryption.

Integrated Encryption A Key Management Mode in which the plaintext is directly encrypted without the use of a Content Encryption Key (CEK).

The definition of Key Management Mode above replaces the one in JWE [RFC7516].

4. Overview

This specification defines the use of HPKE in JWE for two Key Management Modes:

* Key Encryption, and

* Integrated Encryption.

It specifies the Integrated Encryption Key Management Mode and registers the corresponding JWE algorithm identifiers for both modes. Distinct JWE algorithms are defined for Key Encryption and Integrated Encryption so that they are fully specified, as required by [RFC9864].

When the Key Management Mode is Integrated Encryption, HPKE is used to directly encrypt the plaintext, and the "enc" header parameter MUST NOT be included. This specification updates the definition of the "enc" header parameter in Section 4.1.2 of [RFC7516] to require that it be omitted when Integrated Encryption is used.

When the Key Management Mode is Key Encryption, HPKE is used to encrypt the Content Encryption Key (CEK). In this mode, the "enc" header parameter is used as specified in JWE [RFC7516]. The HPKE AEAD encryption function used internally by HPKE is distinct from the JWE AEAD algorithm specified in "enc".

In both Key Management Modes, the HPKE key encapsulation mechanism (KEM), key derivation function (KDF), and authenticated encryption with additional data (AEAD) encryption function utilized depend on the JWE algorithm used.

HPKE supports two modes, which are described in Table 1 of [I-D.ietf-hpke-hpke]. In this specification, both "mode_base" and "mode_psk" are supported for both Key Management Modes. When the "psk_id" header parameter is present, the HPKE mode is "mode_psk"; otherwise, the HPKE mode is "mode_base".

JWE supports two kinds of serializations:

* the JWE Compact Serialization described in Section 3.1 of [RFC7516], and

* the JWE JSON Serialization described in Section 3.2 of [RFC7516].

Certain JWE features are only supported in specific serializations. For example, the JWE Compact Serialization does not support:

* the additional authenticated data header parameter "aad",

* multiple recipients, and

* unprotected header parameters.

Key Encryption can be used with the "aad" header parameter when using the JWE JSON Serialization. Single recipient Key Encryption with no "aad" header parameter can be expressed in the JWE Compact Serialization.

4.1. Encapsulated Secrets

HPKE encapsulated secret is defined in Section 5 of [I-D.ietf-hpke-hpke].

When using Integrated Encryption, the JWE Encrypted Key of the sole recipient is the HPKE encapsulated secret.

When using Key Encryption, each recipient's JWE Encrypted Key is the encrypted content encryption key, and the value of header parameter "ek" is the base64url encoding of the HPKE encapsulated secret.

5. Integrated Encryption

When using Integrated Encryption with HPKE:

* The protected header MUST contain an "alg" value that is an HPKE JWE algorithm using Integrated Encryption.

* The "enc" header parameter MUST NOT be present. This is because no separate content encryption algorithm is used in this mode.

* The protected header parameter "psk_id" MAY be present.

* The protected header parameter "ek" MUST NOT be present.

* There MUST be exactly one recipient.

* The JWE Encrypted Key MUST be the encapsulated secret, as defined in Section 5 of [I-D.ietf-hpke-hpke].

* The JWE Initialization Vector and JWE Authentication Tag MUST be the empty octet sequence.

* The JWE AAD MAY be present when using the JWE JSON Serialization.

* The HPKE aad parameter MUST be set to the "Additional Authenticated Data encryption parameter" value specified in Step 15 of Section 7.1.

* The HPKE info parameter defaults to the empty octet sequence; mutually known private information (a concept also utilized in [NIST.SP.800-56Ar3]) MAY be used instead so the application can include it during key derivation.

* The JWE Ciphertext is the ciphertext from the HPKE encryption, as defined in Section 5.2 of [I-D.ietf-hpke-hpke].

5.1. Integrated Encryption Algorithms using HPKE

The following JWE algorithms using HPKE are defined for use with the Integrated Encryption Key Establishment Mode:

"alg"	HPKE KEM	HPKE KDF	HPKE AEAD
HPKE-0	DHKEM(P-256, HKDF-SHA256)	HKDF-SHA256	AES-128-GCM
HPKE-1	DHKEM(P-384, HKDF-SHA384)	HKDF-SHA384	AES-256-GCM
HPKE-2	DHKEM(P-521, HKDF-SHA512)	HKDF-SHA512	AES-256-GCM
HPKE-3	DHKEM(X25519, HKDF-SHA256)	HKDF-SHA256	AES-128-GCM
HPKE-4	DHKEM(X25519, HKDF-SHA256)	HKDF-SHA256	ChaCha20Poly1305
HPKE-5	DHKEM(X448, HKDF-SHA512)	HKDF-SHA512	AES-256-GCM
HPKE-6	DHKEM(X448, HKDF-SHA512)	HKDF-SHA512	ChaCha20Poly1305
HPKE-7	DHKEM(P-256, HKDF-SHA256)	HKDF-SHA256	AES-256-GCM

Figure 1: Algorithms using HPKE for Integrated Encryption

The HPKE KEM, KDF, and AEAD values are chosen from the IANA HPKE registry [IANA.HPKE].

5.2. JWE Compact Serialization Example

Below is an example of a JWE using the Compact Serialization and Integrated Encryption with HPKE:

eyJhbGciOiJIUEtFLTAiLCJraWQiOiJ5Q25mYm1ZTVpjV3JLRHRfRGpOZWJSQ0IxdnhWb3F2NHVtSjRXSzhSWWprI
n0.BLAJX8adrFsDKaoJAc3iy2dq-6jEH3Uv-bSgqIoDeREqpWglMoTS67XsXere1ZYxiQKEFU6MbWe8O7vmdlSmcU
k..NcN9ew5aijn8W7piLVRU8r2cOP0JKqxOF4RllVsJM4qsAfVXW5Ka6so9zdUmXXNOXyCEk0wV_s8ICAnD4LbRa5
TkhTeuhijIfAt9bQ2fMLOeyed3WyArs8yaMraa9Zbh4i6SaHunM7jU_xoz_N2WbykSOSySmCO49H4mP3jLW9L_TYQ
feVfYsrB8clqokZ8h-3eQGNwmOPtkjWdpAfaHUsp4-HC9nRd6yrTU6mV65Nn2iYynu3Xkgy2Lm-kQKDavIEW3PBpE
eiw6mtPJE9o8sT-0lZ9kpWtqog2XbNGEfjSOjujvNe1b0g4-FdNFMFO_fo0rxe902W1pGT7znv4Q-xBkIydK4Zwji
FN6dAXutnococ37A0Hr5esPLwHRTTrBFw.

The key used for this example is in Appendix A.1.

5.3. Flattened JWE JSON Serialization Example

Below is an example of a JWE using the Flattened JSON Serialization and Integrated Encryption with HPKE:


```
{
  "ciphertext": "LabI8_KIPDbymUSbyVctj8AfISXQ07sMt1xQ1lrS-0heU2jjejpQIK75K1KXcvwn15E6Kil_
tJ6LBcYCu02O1H8_aooJGuoLw1vEzQn16h498YX9e2SA2IcVrJTkcCjL7YpF9fsAF3JEzGfsmmrpZPPVdxCn7g8dk
GRcyulnHrNvBu4BFtub-URtf-nYCFIJHZ4k-ul9fDddquicFzCxQonx66-ZX5nbj6azHG65tAZntd6VFkRgihdxTv
IpvTS4gfulQeKyShbiw-OCJNbzFdEnOKEMnsyqRjwG7iVrFEilFAMsvLJ14-lcuR5btIkUntIwlnsfUa2Ytk33znC
fAFN0wYukdDvJe-V0nnNUFlOeLyYV0eEGisgC9dQQ1kFu3g",
  "encrypted_key": "BAOlZ-VnbhQu4NOlTlDAVYwUJB-Q6YcWwnRNWK6YLSiHHlW4rN0qUzBJ3Rc2_y8nkasn8
nUVGBzdq7OhdKKiLq4",
  "aad": "VGhlIEZlbGxvd3NoaXAgb2YgdGhlIFJpbmc",
  "protected": "eyJhbGciOiJIUEtFLTAiLCJraWQiOiJ5Q25mYm1ZTVpjV3JLRHRfRGpOZWJSQ0IxdnhWb3F2N
HVtSjRXSzhSWWprIn0"
}
```


The key used for this example is in Appendix A.1.

6. Key Encryption

When using the JWE JSON Serialization, recipients using Key Encryption with HPKE can be added alongside other recipients (e.g., those using ECDH-ES+A128KW or RSA-OAEP-256), since HPKE is used to encrypt the Content Encryption Key (CEK).

When using Key Encryption with HPKE:

* The "alg" header parameter MUST be a HPKE JWE algorithm using Key Encryption.

* The header parameter "psk_id" MAY be present.

* The header parameter "ek" MUST be present and contain the base64url-encoded HPKE encapsulated secret.

* The HPKE aad parameter defaults to the empty octet sequence.

* The HPKE info parameter is set to the value of the Recipient_structure defined below.

* THE HPKE plaintext MUST be set to the CEK.

* The recipient’s JWE Encrypted Key is the ciphertext from the HPKE Encryption, as defined in Section 5.2 of [I-D.ietf-hpke-hpke].

6.1. Recipient_structure

The Recipient_structure used as the value of the HPKE info parameter when performing Key Encryption with HPKE provides context information used in key derivation. To ensure compactness and interoperability, this structure is encoded in a binary format. The encoding is as follows:

```
Recipient_structure = ASCII("JOSE-HPKE rcpt") ||
                      BYTE(255) ||
                      ASCII(content_encryption_alg) ||
                      BYTE(255) ||
                      recipient_extra_info
```

Where:

* ASCII("JOSE-HPKE rcpt"): A fixed ASCII string identifying the context of the structure.

* BYTE(255): A separator byte (0xFF) used to delimit fields.

* ASCII(content_encryption_alg): Identifies the content encryption algorithm with which the HPKE-encrypted Content Encryption Key (CEK) is used. Its value MUST be the "enc" (encryption algorithm) header parameter value in the JOSE Header. This field provides JWE context information to the HPKE key schedule, which ensures that the encapsulated secret is bound to the selected content encryption algorithm.

* BYTE(255): A separator byte (0xFF) used to delimit fields.

* recipient_extra_info: An octet string containing additional context information that the application includes in the key derivation. Mutually known private information (a concept also utilized in [NIST.SP.800-56Ar3]) MAY be used in this input parameter. If no additional context information is provided, this field MUST be the empty octet sequence.

Note that Integrated Encryption does not use the Recipient_structure because the JWE Protected Header and JWE AAD are included in the HPKE aad value, which binds these parameters to the ciphertext.

6.1.1. Recipient_structure Example

The Recipient_structure encoded in binary as specified in Section 6.1, and using the field values (content_encryption_alg = "A128GCM", recipient_extra_info = ""), results in the following byte sequence:

"JOSE-HPKE rcpt\xffA128GCM\xff"

The corresponding hexadecimal representation is:

4a4f53452d48504b452072637074ff4131323847434dff

This value is used as the HPKE info parameter when performing Key Encryption with HPKE.

6.2. Key Encryption Algorithms using HPKE

The following JWE algorithms using HPKE are defined for use with the Key Encryption Key Establishment Mode:

"alg"	HPKE KEM	HPKE KDF	HPKE AEAD
HPKE-0-KE	DHKEM(P-256, HKDF-SHA256)	HKDF-SHA256	AES-128-GCM
HPKE-1-KE	DHKEM(P-384, HKDF-SHA384)	HKDF-SHA384	AES-256-GCM
HPKE-2-KE	DHKEM(P-521, HKDF-SHA512)	HKDF-SHA512	AES-256-GCM
HPKE-3-KE	DHKEM(X25519, HKDF-SHA256)	HKDF-SHA256	AES-128-GCM
HPKE-4-KE	DHKEM(X25519, HKDF-SHA256)	HKDF-SHA256	ChaCha20Poly1305
HPKE-5-KE	DHKEM(X448, HKDF-SHA512)	HKDF-SHA512	AES-256-GCM
HPKE-6-KE	DHKEM(X448, HKDF-SHA512)	HKDF-SHA512	ChaCha20Poly1305
HPKE-7-KE	DHKEM(P-256, HKDF-SHA256)	HKDF-SHA256	AES-256-GCM

Figure 2: Algorithms using HPKE for Key Encryption

The HPKE KEM, KDF, and AEAD values are chosen from the IANA HPKE registry [IANA.HPKE].

6.3. General JWE JSON Serialization Example

Below is an example of a JWE using the General JSON Serialization and Key Encryption with HPKE:


```
{
  "ciphertext": "uF1XBbVZWhYm_pDbeJvI_fkuqFJiKd1WMP3O_BAGOP-LkpTLE3Et2VQNcOpPAIBfyx8rUzsh
GqiOFOWzcoWZ3mIwYuDvvAW3-P1RCS8Dtq70JRvahO5O8sAN1vzJg8_dyBPnwsQY6Cy3RhMD6sSSCjjSw0FYmmx67
IiI2zJ6Wr8z69k0f34ZTh43k4C-pTwaUSvjl2XI_YrUgdDVYmY_MJ5vmlPTcceMaefP8Onz_fx5xOcGfnVBVz2gpM
QPuQL8k5Rk5KJvPGfFfN6hrgWkK_LDzi4lrfnIrvNsk3BCBeZPpc-n19-u7W4-GQxLjAlVyMHeGk5K4tU6gHB8Pnn
Q4ND5ZTtyXrJWQW-Qr1iFev6g",
  "iv": "mLiHjYaQA42nPm1L",
  "recipients": [
    {
      "encrypted_key": "hU6b0hp4-y4ZoK1Qz8YWmDmqDmgTto3HW25-RyPhcLU",
      "header": {
        "alg": "HPKE-0-KE",
        "kid": "9CfUPiGcAcTp7oXgVbDStw2FEjka-_KHU_i-X3XMCEA",
        "ek": "BGWPWLoD5BUjFEDIjMS-yvtcCXBn5A-kuv2RjzUY_2hKUjgZINqtEy1aHZ8dWxAiyApV5JafG7
6W8O_yZzy5T54"
      }
    }
  ],
  "tag": "K22C64ZhFABEu2S2F00PLg",
  "aad": "VGhlIEZlbGxvd3NoaXAgb2YgdGhlIFJpbmc",
  "protected": "eyJlbmMiOiJBMTI4R0NNIn0"
}
```

The key used for this example is in Appendix A.2.

7. Producing and Consuming JWEs

Sections 5.1 (Message Encryption) and 5.2 (Message Decryption) of [RFC7516] are replaced by the following sections, which add processing rules for using the Integrated Encryption Key Management Mode.

7.1. Message Encryption

The message encryption process is as follows. The order of the steps is not significant in cases where there are no dependencies between the inputs and outputs of the steps.

1. Determine the Key Management Mode employed by the algorithm used to determine the Content Encryption Key value. (This is the algorithm recorded in the alg (algorithm) Header Parameter of the resulting JWE.)

2. When Key Wrapping, Key Encryption, or Key Agreement with Key Wrapping are employed, generate a random CEK value to use for subsequent steps unless one was already generated for a previously processed recipient, in which case, let that be the one used for subsequent steps. See [RFC4086] for considerations on generating random values. The CEK MUST have a length equal to that required for the content encryption algorithm.

3. When Direct Key Agreement or Key Agreement with Key Wrapping are employed, use the key agreement algorithm to compute the value of the agreed upon key. When Direct Key Agreement is employed, let the CEK be the agreed upon key. When Key Agreement with Key Wrapping is employed, the agreed upon key will be used to wrap the CEK.

4. When Key Wrapping, Key Encryption, or Key Agreement with Key Wrapping are employed, encrypt the CEK to the recipient and let the result be the JWE Encrypted Key.

5. When Direct Key Agreement or Direct Encryption are employed, let the JWE Encrypted Key be the empty octet sequence.

6. When Direct Encryption is employed, let the CEK be the shared symmetric key.

7. When Integrated Encryption is employed, let the JWE Encrypted Key be as specified by the Integrated Encryption algorithm.

8. Compute the encoded key value BASE64URL(JWE Encrypted Key).

9. If the JWE JSON Serialization is being used, repeat this process (steps 1-8) for each recipient.

10. Generate a random JWE Initialization Vector of the correct size for the content encryption algorithm (if required for the algorithm); otherwise, let the JWE Initialization Vector be the empty octet sequence.

11. Compute the encoded Initialization Vector value BASE64URL(JWE Initialization Vector).

12. If a zip parameter was included, compress the plaintext using the specified compression algorithm and let M be the octet sequence representing the compressed plaintext; otherwise, let M be the octet sequence representing the plaintext.

13. Create the JSON object(s) containing the desired set of Header Parameters, which together comprise the JOSE Header: one or more of the JWE Protected Header, the JWE Shared Unprotected Header, and the JWE Per-Recipient Unprotected Header.

14. Compute the Encoded Protected Header value BASE64URL(UTF8(JWE Protected Header)). If the JWE Protected Header is not present (which can only happen when using the JWE JSON Serialization and no protected member is present), let this value be the empty string.

15. Let the Additional Authenticated Data encryption parameter be ASCII(Encoded Protected Header). However, if a JWE AAD value is present (which can only be the case when using the JWE JSON Serialization), instead let the Additional Authenticated Data encryption parameter be ASCII(Encoded Protected Header || '.' || BASE64URL(JWE AAD)).

16. If Integrated Encryption is not being employed, encrypt M using the CEK, the JWE Initialization Vector, and the Additional Authenticated Data value using the specified content encryption algorithm to create the JWE Ciphertext value and the JWE Authentication Tag (which is the Authentication Tag output from the encryption operation).

17. If Integrated Encryption is being employed, encrypt M using the specified Integrated Encryption algorithm to create the JWE Ciphertext value. Let the JWE Authentication Tag be the empty octet sequence.

18. Compute the encoded ciphertext value BASE64URL(JWE Ciphertext).

19. Compute the encoded Authentication Tag value BASE64URL(JWE Authentication Tag).

20. If a JWE AAD value is present, compute the encoded AAD value BASE64URL(JWE AAD).

21. Create the desired serialized output. The Compact Serialization of this result is the string BASE64URL(UTF8(JWE Protected Header)) || '.' || BASE64URL(JWE Encrypted Key) || '.' || BASE64URL(JWE Initialization Vector) || '.' || BASE64URL(JWE Ciphertext) || '.' || BASE64URL(JWE Authentication Tag). The JWE JSON Serialization is described in Section 7.2 of [RFC7516].

7.2. Message Decryption

The message decryption process is the reverse of the encryption process. The order of the steps is not significant in cases where there are no dependencies between the inputs and outputs of the steps. If any of these steps fail, the encrypted content cannot be validated.

When there are multiple recipients, it is an application decision which of the recipients' encrypted content must successfully validate for the JWE to be accepted. In some cases, encrypted content for all recipients must successfully validate or the JWE will be considered invalid. In other cases, only the encrypted content for a single recipient needs to be successfully validated. However, in all cases, the encrypted content for at least one recipient MUST successfully validate or the JWE MUST be considered invalid.

1. Parse the JWE representation to extract the serialized values for the components of the JWE. When using the JWE Compact Serialization, these components are the base64url-encoded representations of the JWE Protected Header, the JWE Encrypted Key, the JWE Initialization Vector, the JWE Ciphertext, and the JWE Authentication Tag, and when using the JWE JSON Serialization, these components also include the base64url-encoded representation of the JWE AAD and the unencoded JWE Shared Unprotected Header and JWE Per-Recipient Unprotected Header values. When using the JWE Compact Serialization, the JWE Protected Header, the JWE Encrypted Key, the JWE Initialization Vector, the JWE Ciphertext, and the JWE Authentication Tag are represented as base64url-encoded values in that order, with each value being separated from the next by a single period ('.') character, resulting in exactly four delimiting period characters being used. The JWE JSON Serialization is described in Section 7.2 of [RFC7516].

2. Base64url decode the encoded representations of the JWE Protected Header, the JWE Encrypted Key, the JWE Initialization Vector, the JWE Ciphertext, the JWE Authentication Tag, and the JWE AAD, following the restriction that no line breaks, whitespace, or other additional characters have been used.

3. Verify that the octet sequence resulting from decoding the encoded JWE Protected Header is a UTF-8-encoded representation of a completely valid JSON object conforming to [RFC8259]; let the JWE Protected Header be this JSON object.

4. If using the JWE Compact Serialization, let the JOSE Header be the JWE Protected Header. Otherwise, when using the JWE JSON Serialization, let the JOSE Header be the union of the members of the JWE Protected Header, the JWE Shared Unprotected Header and the corresponding JWE Per-Recipient Unprotected Header, all of which must be completely valid JSON objects. During this step, verify that the resulting JOSE Header does not contain duplicate Header Parameter names. When using the JWE JSON Serialization, this restriction includes that the same Header Parameter name also MUST NOT occur in distinct JSON object values that together comprise the JOSE Header.

5. Verify that the implementation understands and can process all fields that it is required to support, whether required by this specification, by the algorithms being used, or by the crit Header Parameter value, and that the values of those parameters are also understood and supported.

6. Determine the Key Management Mode employed by the algorithm specified by the alg (algorithm) Header Parameter.

7. If using Integrated Encryption, Direct Encryption or Direct Key Agreement, verify that there is exactly one recipient.

8. Verify that the JWE uses a key known to the recipient.

9. When Direct Key Agreement or Key Agreement with Key Wrapping are employed, use the key agreement algorithm to compute the value of the agreed upon key. When Direct Key Agreement is employed, let the CEK be the agreed upon key. When Key Agreement with Key Wrapping is employed, the agreed upon key will be used to decrypt the JWE Encrypted Key.

10. When Key Wrapping, Key Encryption, or Key Agreement with Key Wrapping are employed, decrypt the JWE Encrypted Key to produce the CEK. The CEK MUST have a length equal to that required for the content encryption algorithm. Note that when there are multiple recipients, each recipient will only be able to decrypt JWE Encrypted Key values that were encrypted to a key in that recipient’s possession. It is therefore normal to only be able to decrypt one of the per-recipient JWE Encrypted Key values to obtain the CEK value. Also, see Section 11.5 of [RFC7516] for security considerations on mitigating timing attacks.

11. When Direct Key Agreement or Direct Encryption are employed, verify that the JWE Encrypted Key value is an empty octet sequence.

12. When Direct Encryption is employed, let the CEK be the shared symmetric key.

13. If Integrated Encryption is not being employed, record whether the CEK could be successfully determined for this recipient or not.

14. If the JWE JSON Serialization is being used, repeat this process (steps 4-13) for each recipient contained in the representation.

15. Compute the Encoded Protected Header value BASE64URL(UTF8(JWE Protected Header)). If the JWE Protected Header is not present (which can only happen when using the JWE JSON Serialization and no protected member is present), let this value be the empty string.

16. Let the Additional Authenticated Data encryption parameter be ASCII(Encoded Protected Header). However, if a JWE AAD value is present (which can only be the case when using the JWE JSON Serialization), instead let the Additional Authenticated Data encryption parameter be ASCII(Encoded Protected Header || '.' || BASE64URL(JWE AAD)).

17. If Integrated Encryption is not being employed, decrypt the JWE Ciphertext using the CEK, the JWE Initialization Vector, the Additional Authenticated Data value, and the JWE Authentication Tag (which is the Authentication Tag input to the calculation) using the content encryption algorithm specified in the "enc" header parameter, returning the decrypted plaintext and validating the JWE Authentication Tag in the manner specified for the algorithm, rejecting the input without emitting any decrypted output if the JWE Authentication Tag is incorrect.

18. If Integrated Encryption is being employed, verify that no "enc" header parameter is present.

19. If Integrated Encryption is being employed, decrypt the JWE Ciphertext using the specified Integrated Encryption algorithm, returning the decrypted plaintext in the manner specified for the algorithm, rejecting the input without emitting any decrypted output if the decryption fails.

20. If a zip parameter was included, uncompress the decrypted plaintext using the specified compression algorithm.

21. If there was no recipient for which all of the decryption steps succeeded, then the JWE MUST be considered invalid. Otherwise, output the plaintext. In the JWE JSON Serialization case, also return a result to the application indicating for which of the recipients the decryption succeeded and failed.

Finally, note that it is an application decision which algorithms may be used in a given context. Even if a JWE can be successfully decrypted, unless the algorithms used in the JWE are acceptable to the application, it SHOULD consider the JWE to be invalid.

8. Distinguishing Between JWS and JWE Objects

Section 9 of [RFC7516] is updated to delete the last bullet, which says:

* The JOSE Header for a JWS can also be distinguished from the JOSE Header for a JWE by determining whether an enc (encryption algorithm) member exists. If the enc member exists, it is a JWE; otherwise, it is a JWS.

The deleted test no longer works when Integrated Encryption is used.

The other methods of distinguishing between JSON Web Signature (JWS) [RFC7515] and JSON Web Encryption (JWE) [RFC7516] objects continue to work.

9. JWK Representations for JWE HPKE Keys

The JSON Web Key (JWK) [RFC7517] representations for keys used with the JWE algorithms defined in this specification are as follows. The valid combinations of the "alg", "kty", and "crv" in the JWK are shown in Figure 3.

"alg" values	"kty"	"crv"
HPKE-0, HPKE-0-KE, HPKE-7, HPKE-7-KE	EC	P-256
HPKE-1, HPKE-1-KE	EC	P-384
HPKE-2, HPKE-2-KE	EC	P-521
HPKE-3, HPKE-3-KE, HPKE-4, HPKE-4-KE	OKP	X25519
HPKE-5, HPKE-5-KE, HPKE-6, HPKE-6-KE	OKP	X448

Figure 3: JWK Types and Curves for JWE HPKE Ciphersuites

9.1. JWK Representation of Key using JWE HPKE Ciphersuite

The example below is a JWK representation of a public and private key used with the Integrated Encryption Key Establishment Mode:

```
{
  "kty": "OKP",
  "crv": "X25519",
  "x": "3pPHgcHYVYpOpB6ISwHdoPRB6jNgd8mM4nRyyj4H3aE",
  "d": "nWGxne0tAiV8Hk6kcy4rN0wMskjl9yND0N3Xeho9n6g",
  "kid": "recipient-key-1",
  "alg": "HPKE-3",
  "use": "enc"
}
```

10. Security Considerations

This specification uses HPKE and the security considerations of [I-D.ietf-hpke-hpke] are therefore applicable.

HPKE assumes the sender is in possession of the public key of the recipient and HPKE JOSE makes the same assumptions. Hence, some form of public key distribution mechanism is assumed to exist but outside the scope of this document.

HPKE in Base mode does not offer authentication as part of the HPKE KEM.

HPKE relies on a source of randomness being available on the device. In Key Agreement with Key Wrapping mode, the CEK has to be randomly generated. The guidance on randomness in [RFC4086] applies.

10.1. Key Management

A single key MUST NOT be used with multiple KEM algorithms. Each key and its associated HPKE algorithm suite, comprising the KEM, KDF, and AEAD, SHOULD be managed independently. This separation prevents unintended interactions or vulnerabilities between algorithms, ensuring the integrity and security guarantees of each algorithm are preserved. Additionally, the same key SHOULD NOT be used for both Key Encryption and Integrated Encryption, as it may introduce security risks. It creates algorithm confusion, increases the potential for key leakage, cross-suite attacks, and improper handling of the key.

10.2. JWT Best Current Practices

The guidance in [RFC8725] about encryption is also pertinent to this specification.

11. IANA Considerations

11.1. JSON Web Signature and Encryption Algorithms

The following entries are added to the IANA "JSON Web Signature and Encryption Algorithms" registry [IANA.JOSE] established by [RFC7518]:

11.1.1. HPKE-0

* Algorithm Name: HPKE-0

* Algorithm Description: Integrated Encryption with HPKE using DHKEM(P-256, HKDF-SHA256) KEM, HKDF-SHA256 KDF and AES-128-GCM AEAD

* Algorithm Usage Location(s): "alg"

* JOSE Implementation Requirements: Optional

* Change Controller: IETF

* Specification Document(s): Section 5.1 of [[this specification]]

* Algorithm Analysis Documents(s): [I-D.ietf-hpke-hpke]

11.1.2. HPKE-1

* Algorithm Name: HPKE-1

* Algorithm Description: Integrated Encryption with HPKE using DHKEM(P-384, HKDF-SHA384) KEM, HKDF-SHA384 KDF, and AES-256-GCM AEAD

* Algorithm Usage Location(s): "alg"

* JOSE Implementation Requirements: Optional

* Change Controller: IETF

* Specification Document(s): Section 5.1 of [[this specification]]

* Algorithm Analysis Documents(s): [I-D.ietf-hpke-hpke]

11.1.3. HPKE-2

* Algorithm Name: HPKE-2

* Algorithm Description: Integrated Encryption with HPKE using DHKEM(P-521, HKDF-SHA512) KEM, HKDF-SHA512 KDF, and AES-256-GCM AEAD

* Algorithm Usage Location(s): "alg"

* JOSE Implementation Requirements: Optional

* Change Controller: IETF

* Specification Document(s): Section 5.1 of [[this specification]]

* Algorithm Analysis Documents(s): [I-D.ietf-hpke-hpke]

11.1.4. HPKE-3

* Algorithm Name: HPKE-3
* Algorithm Description: Integrated Encryption with HPKE using DHKEM(X25519, HKDF-SHA256) KEM, HKDF-SHA256 KDF, and AES-128-GCM AEAD
* Algorithm Usage Location(s): "alg"
* JOSE Implementation Requirements: Optional
* Change Controller: IETF
* Specification Document(s): Section 5.1 of [[this specification]]
* Algorithm Analysis Documents(s): [I-D.ietf-hpke-hpke]

11.1.5. HPKE-4

* Algorithm Name: HPKE-4
* Algorithm Description: Integrated Encryption with HPKE using DHKEM(X25519, HKDF-SHA256) KEM, HKDF-SHA256 KDF, and ChaCha20Poly1305 AEAD
* Algorithm Usage Location(s): "alg"
* JOSE Implementation Requirements: Optional
* Change Controller: IETF
* Specification Document(s): Section 5.1 of [[this specification]]
* Algorithm Analysis Documents(s): [I-D.ietf-hpke-hpke]

11.1.6. HPKE-5

* Algorithm Name: HPKE-5
* Algorithm Description: Integrated Encryption with HPKE using DHKEM(X448, HKDF-SHA512) KEM, HKDF-SHA512 KDF, and AES-256-GCM AEAD
* Algorithm Usage Location(s): "alg"

* JOSE Implementation Requirements: Optional

* Change Controller: IETF

* Specification Document(s): Section 5.1 of [[this specification]]

* Algorithm Analysis Documents(s): [I-D.ietf-hpke-hpke]

11.1.7. HPKE-6

* Algorithm Name: HPKE-6

* Algorithm Description: Integrated Encryption with HPKE using DHKEM(X448, HKDF-SHA512) KEM, HKDF-SHA512 KDF, and ChaCha20Poly1305 AEAD

* Algorithm Usage Location(s): "alg"

* JOSE Implementation Requirements: Optional

* Change Controller: IETF

* Specification Document(s): Section 5.1 of [[this specification]]

* Algorithm Analysis Documents(s): [I-D.ietf-hpke-hpke]

11.1.8. HPKE-7

* Algorithm Name: HPKE-7

* Algorithm Description: Integrated Encryption with HPKE using DHKEM(P-256, HKDF-SHA256) KEM, HKDF-SHA256 KDF, and AES-256-GCM AEAD

* Algorithm Usage Location(s): "alg"

* JOSE Implementation Requirements: Optional

* Change Controller: IETF

* Specification Document(s): Section 5.1 of [[this specification]]

* Algorithm Analysis Documents(s): [I-D.ietf-hpke-hpke]

11.1.9. HPKE-0-KE

* Algorithm Name: HPKE-0-KE

* Algorithm Description: Key Encryption with HPKE using DHKEM(P-256, HKDF-SHA256) KEM, HKDF-SHA256 KDF and AES-128-GCM AEAD

* Algorithm Usage Location(s): "alg"

* JOSE Implementation Requirements: Optional

* Change Controller: IETF

* Specification Document(s): Section 6.2 of [[this specification]]

* Algorithm Analysis Documents(s): [I-D.ietf-hpke-hpke]

11.1.10. HPKE-1-KE

* Algorithm Name: HPKE-1-KE

* Algorithm Description: Key Encryption with HPKE using DHKEM(P-384, HKDF-SHA384) KEM, HKDF-SHA384 KDF, and AES-256-GCM AEAD

* Algorithm Usage Location(s): "alg"

* JOSE Implementation Requirements: Optional

* Change Controller: IETF

* Specification Document(s): Section 6.2 of [[this specification]]

* Algorithm Analysis Documents(s): [I-D.ietf-hpke-hpke]

11.1.11. HPKE-2-KE

* Algorithm Name: HPKE-2-KE

* Algorithm Description: Key Encryption with HPKE using DHKEM(P-521, HKDF-SHA512) KEM, HKDF-SHA512 KDF, and AES-256-GCM AEAD

* Algorithm Usage Location(s): "alg"

* JOSE Implementation Requirements: Optional

* Change Controller: IETF

* Specification Document(s): Section 6.2 of [[this specification]]

* Algorithm Analysis Documents(s): [I-D.ietf-hpke-hpke]

11.1.12. HPKE-3-KE

* Algorithm Name: HPKE-3-KE
* Algorithm Description: Key Encryption with HPKE using DHKEM(X25519, HKDF-SHA256) KEM, HKDF-SHA256 KDF, and AES-128-GCM AEAD
* Algorithm Usage Location(s): "alg"
* JOSE Implementation Requirements: Optional
* Change Controller: IETF
* Specification Document(s): Section 6.2 of [[this specification]]
* Algorithm Analysis Documents(s): [I-D.ietf-hpke-hpke]

11.1.13. HPKE-4-KE

* Algorithm Name: HPKE-4-KE
* Algorithm Description: Key Encryption with HPKE using DHKEM(X25519, HKDF-SHA256) KEM, HKDF-SHA256 KDF, and ChaCha20Poly1305 AEAD
* Algorithm Usage Location(s): "alg"
* JOSE Implementation Requirements: Optional
* Change Controller: IETF
* Specification Document(s): Section 6.2 of [[this specification]]
* Algorithm Analysis Documents(s): [I-D.ietf-hpke-hpke]

11.1.14. HPKE-5-KE

* Algorithm Name: HPKE-5-KE
* Algorithm Description: Key Encryption with HPKE using DHKEM(X448, HKDF-SHA512) KEM, HKDF-SHA512 KDF, and AES-256-GCM AEAD
* Algorithm Usage Location(s): "alg"
* JOSE Implementation Requirements: Optional
* Change Controller: IETF

* Specification Document(s): Section 6.2 of [[this specification]]

* Algorithm Analysis Documents(s): [I-D.ietf-hpke-hpke]

11.1.15. HPKE-6-KE

* Algorithm Name: HPKE-6-KE

* Algorithm Description: Key Encryption with HPKE using DHKEM(X448, HKDF-SHA512) KEM, HKDF-SHA512 KDF, and ChaCha20Poly1305 AEAD

* Algorithm Usage Location(s): "alg"

* JOSE Implementation Requirements: Optional

* Change Controller: IETF

* Specification Document(s): Section 6.2 of [[this specification]]

* Algorithm Analysis Documents(s): [I-D.ietf-hpke-hpke]

11.1.16. HPKE-7-KE

* Algorithm Name: HPKE-7-KE

* Algorithm Description: Key Encryption with HPKE using DHKEM(P-256, HKDF-SHA256) KEM, HKDF-SHA256 KDF, and AES-256-GCM AEAD

* Algorithm Usage Location(s): "alg"

* JOSE Implementation Requirements: Optional

* Change Controller: IETF

* Specification Document(s): Section 6.2 of [[this specification]]

* Algorithm Analysis Documents(s): [I-D.ietf-hpke-hpke]

11.2. JSON Web Signature and Encryption Header Parameters

The following entries are added to the IANA "JSON Web Key Parameters" registry [IANA.JOSE]:

11.2.1. ek

* Header Parameter Name: "ek"

* Header Parameter Description: A base64url-encoded encapsulated secret, as defined in Section 5 of [I-D.ietf-hpke-hpke]

* Header Parameter Usage Location(s): JWE

* Change Controller: IETF

* Specification Document(s): Section 4.1 of [[this specification]]

11.2.2. psk_id

* Header Parameter Name: "psk_id"

* Header Parameter Description: A base64url-encoded key identifier (kid) for the pre-shared key, as defined in Section 5.1.2 of [I-D.ietf-hpke-hpke]

* Header Parameter Usage Location(s): JWE

* Change Controller: IETF

* Specification Document(s): Section 4 of [[this specification]]

12. Summary of Updates to RFC 7516 (JWE)

This specification updates JSON Web Encryption (JWE) [RFC7516] as follows:

* Adds the Integrated Encryption Key Management Mode and correspondingly updates the Key Management Mode definition (Section 3).

* Updates the "enc" header parameter to be absent when Integrated Encryption is used in (Section 4).

* Replaces the Message Encryption procedure (Section 7.1).

* Replaces the Message Decryption procedure (Section 7.2).

* Updates the methods for distinguishing between JWS and JWE objects (Section 8).

13. References

13.1. Normative References

[I-D.ietf-hpke-hpke]
Barnes, R., Bhargavan, K., Lipp, B., and C. A. Wood, "Hybrid Public Key Encryption", Work in Progress, Internet-Draft, draft-ietf-hpke-hpke-02, 4 November 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-hpke-hpke-02>.

[IANA.JOSE]
IANA, "JSON Web Signature and Encryption Algorithms", n.d., <https://www.iana.org/assignments/jose>.

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

[RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015, <https://www.rfc-editor.org/rfc/rfc7516>.

[RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <https://www.rfc-editor.org/rfc/rfc7517>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/rfc/rfc8259>.

[RFC8725] Sheffer, Y., Hardt, D., and M. Jones, "JSON Web Token Best Current Practices", BCP 225, RFC 8725, DOI 10.17487/RFC8725, February 2020, <https://www.rfc-editor.org/rfc/rfc8725>.

13.2. Informative References

[I-D.ietf-cose-hpke]
Tschofenig, H., Steele, O., Daisuke, A., and L. Lundblade, "Use of Hybrid Public-Key Encryption (HPKE) with CBOR Object Signing and Encryption (COSE)", Work in Progress, Internet-Draft, draft-ietf-cose-hpke-18, 19 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-cose-hpke-18>.

[IANA.HPKE]
IANA, "Hybrid Public Key Encryption (HPKE)", n.d., <https://www.iana.org/assignments/hpke>.

[NIST.SP.800-56Ar3]
National Institute of Standards and Technology, "Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography, NIST Special Publication 800-56A Revision 3", April 2018, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/rfc/rfc4086>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/rfc/rfc7515>.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <https://www.rfc-editor.org/rfc/rfc7518>.

[RFC9864] Jones, M.B. and O. Steele, "Fully-Specified Algorithms for JSON Object Signing and Encryption (JOSE) and CBOR Object Signing and Encryption (COSE)", RFC 9864, DOI 10.17487/RFC9864, October 2025, <https://www.rfc-editor.org/rfc/rfc9864>.

Appendix A. Keys Used in Examples

A.1. Integrated Encryption Key

This private key and its implied public key are used for the Integrated Encryption example in Section 5.2 and Section 5.3:

```
{
  "kty": "EC",
  "use": "enc",
  "alg": "HPKE-0",
  "kid": "yCnfbmYMZcWrKDt_DjNebRCB1vxVoqv4umJ4WK8RYjk",
  "crv": "P-256",
  "x": "gixQJ0qg4Ag-6HSMaIEDL_zbDhoXavMyKlmdn__AQVE",
  "y": "ZxTgRLWaKONCL_GbZKLNPsW9EW6nBsN4AwQGEFAFFbM",
  "d": "g2DXtKapi2oN2zL_RCWX8D4bWURHCKN2-ZNGC05ZaR8"
}
```

A.2. Key Encryption Key

This private key and its implied public key are used for the Key Encryption example in Section 6.3:


```
{
  "kty": "EC",
  "use": "enc",
  "alg": "HPKE-0-KE",
  "kid": "9CfUPiGcAcTp7oXgVbDStw2FEjka-_KHU_i-X3XMCEA",
  "crv": "P-256",
  "x": "WVKOswXQAgntIrLSYlwkyaU1dIE-FIhrbTEotFgMwIA",
  "y": "jpZT1WNmQH752Bh_pDK41IhLkiXLj-15wR4ZBZ-MWFk",
  "d": "MeCnMF65SaRVZ11Gf1Weacx3H9SdzO7MtWcDXvHWNv8"
}
```


Acknowledgments

This specification leverages text from [I-D.ietf-cose-hpke]. We would like to thank Richard Barnes, Brian Campbell, Matt Chanda, Ilari Liusvaara, Neil Madden, Aaron Parecki, Filip Skokan, and Sebastian Stenzel for their contributions to the specification.

Document History

-15

* Defined the Integrated Encryption Key Establishment Mode and updated JWE to enable its use.
* Specified distinct algorithms for use with Key Encryption and Integrated Encryption so that they are fully-specified.
* Updated the Message Encryption and Message Decryption procedures from JWE.
* Said that JWS and JWE objects can no longer be distinguished by the presence of an "enc" header parameter.
* Many editorial improvements.

-14

* Added HPKE-7.
* Update to Recipient_structure.
* Removed text related to apu and apv.

* Updated description of mutually known private information.

-13

* Removed orphan text about AKP kty field

* Fixed bug in "include-fold" syntax

* Switched reference from RFC 9180 to draft-ietf-hpke-hpke

* Editorial improvements to abstract and introduction.

* Removed Section 8.2 "Static Asymmetric Authentication in HPKE"

-12

* Added the Recipient_structure

-11

* Fix too long lines

-10

* Addressed WGLC review comments by Neil Madden and Sebastian Stenzel.

-09

* Corrected examples.

-08

* Use "enc":"int" for integrated encryption.

* Described reasons for excluding authenticated HPKE.

* Stated that mutually known private information MAY be used as the HPKE info value.

-07

* Clarifications

-06

* Remove auth mode and auth_kid from the specification.

* HPKE AAD for JOSE HPKE Key Encryption is now empty.

-05

* Removed incorrect text about HPKE algorithm names.

* Fixed #21: Comply with NIST SP 800-227 Recommendations for Key-Encapsulation Mechanisms.

* Fixed #19: Binding the Application Context.

* Fixed #18: Use of apu and apv in Recipient context.

* Added new Section 7.1 (Authentication using an Asymmetric Key).

* Updated Section 7.2 (Key Management) to prevent cross-protocol attacks.

* Updated HPKE Setup info parameter to be empty.

* Added details on HPKE AEAD AAD, compression and decryption for HPKE Integrated Encryption.

-04

* Fixed #8: Use short algorithm identifiers, per the JOSE naming conventions.

-03

* Added new section 7.1 to discuss Key Management.

* HPKE Setup info parameter is updated to carry JOSE context-specific data for both modes.

-02

* Fixed #4: HPKE Integrated Encryption "enc: dir".

* Updated text on the use of HPKE Setup info parameter.

* Added Examples in Sections 5.1, 5.2 and 6.1.

* Use of registered HPKE "alg" value in the recipient unprotected header for Key Encryption.

-01

* Apply feedback from call for adoption.

* Provide examples of auth and psk modes for JSON and Compact Serializations

* Simplify description of HPKE modes

* Adjust IANA registration requests

* Remove HPKE Mode from named algorithms

* Fix AEAD named algorithms

-00

* Created initial working group version from draft-rha-jose-hpke-encrypt-07

Authors' Addresses

Tirumaleswar Reddy
Nokia
Bangalore
Karnataka
India
Email: kondtir@gmail.com

Hannes Tschofenig
University of Applied Sciences Bonn-Rhein-Sieg
Germany
Email: hannes.tschofenig@gmx.net

Aritra Banerjee
Nokia
London
United Kingdom
Email: aritra.banerjee@nokia.com

Orie Steele
Tradeverifyd
United States
Email: orie@or13.io

Michael B. Jones
Self-Issued Consulting
United States
Email: michael_b_jones@hotmail.com
URI: https://self-issued.info/