Workgroup: JOSE
Internet-Draft: draft-rha-jose-hpke-encrypt-07
Published: 31 March 2024
Intended Status: Standards Track
Expires: 2 October 2024
Authors: T. Reddy H. Tschofenig A. Banerjee O. Steele
Nokia Nokia Transmute
M. Jones
Self-Issued Consulting

Use of Hybrid Public Key Encryption (HPKE) with JSON Object Signing and Encryption (JOSE)

Abstract

This specification defines Hybrid Public Key Encryption (HPKE) for use with JSON Object Signing and Encryption (JOSE). HPKE offers a variant of public key encryption of arbitrary-sized plaintexts for a recipient public key.

HPKE works for any combination of an asymmetric key encapsulation mechanism (KEM), key derivation function (KDF), and authenticated encryption with additional data (AEAD) function. Authentication for HPKE in JOSE is provided by JOSE-native security mechanisms or by one of the authenticated variants of HPKE.

This document defines the use of the HPKE with JOSE.

About This Document

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

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-rha-jose-hpke/.

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/.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 2 October 2024.

Copyright Notice

Copyright (c) 2024 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction
2. Conventions and Definitions
3. Conventions and Terminology
4. HPKE for JOSE
  4.1. Overview
  4.2. HPKE Encryption
  4.3. HPKE Decryption
  4.4. Encapsulated JSON Web Keys
    4.4.1. HPKE Direct Encryption
    4.4.2. HPKE Key Encryption
5. Ciphersuite Registration
6. Security Considerations
  6.1. Plaintext Compression
  6.2. Header Parameters
  6.3. Ensure Cryptographic Keys Have Sufficient Entropy
  6.4. Validate Cryptographic Inputs
  6.5. Use Appropriate Algorithms
7. IANA Considerations
  7.1. JSON Web Key Types
  7.2. JSON Web Key Parameters
  7.3. JSON Web Signature and Encryption Algorithms
  7.4. JSON Web Signature and Encryption Header Parameters
8. References
  8.1. Normative References
  8.2. Informative References

Acknowledgments
Document History
Authors' Addresses

1. Introduction

Hybrid Public Key Encryption (HPKE) [RFC9180] is a scheme that provides public key encryption of arbitrary-sized plaintexts given a recipient's public key.

This specification enables JSON Web Encryption (JWE) to leverage HPKE, bringing support for KEMs and the possibility of Post Quantum or Hybrid KEMs to JWE.

2. Conventions and Definitions

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. Conventions and Terminology

This specification uses the following abbreviations and terms:

*Key Type (kty), see [RFC7517].

*Content Encryption Key (CEK), is defined in [RFC7517].

*Hybrid Public Key Encryption (HPKE) is defined in [RFC9180].

*pkR is the public key of the recipient, as defined in [RFC9180].

*skR is the private key of the recipient, as defined in [RFC9180].

*Key Encapsulation Mechanism (KEM), see [RFC9180].

*Key Derivation Function (KDF), see [RFC9180].

*Authenticated Encryption with Associated Data (AEAD), see [RFC9180].

*Additional Authenticated Data (AAD), see [RFC9180].

4. HPKE for JOSE

4.1. Overview

JSON Web Encryption (JWE) [RFC7516] defines several serializations for expressing encrypted content with JSON:

*Compact JWE Serialization

*General JWE JSON Serialization

*Flattened JWE JSON Serialization

JSON Web Algorithms (JWA) Section 4.6 of [RFC7518] defines two ways to use public key cryptography with JWE:

*Direct Key Agreement

*Key Agreement with Key Wrapping

The specification enables Hybrid Public Key Encryption (HPKE) [RFC9180] to be used with the serializations defined in JWE.

Unless otherwise stated, no changes to the processes described in [RFC7516] have been made.

This specification describes two modes of use for HPKE in JWE:

*HPKE Direct Encryption mode, where HPKE is used to encrypt the plaintext. This mode can only be used with a single recipient. This setup is conceptually similar to Direct Key Agreement.

*HPKE Key Encryption mode, where HPKE is used to encrypt a content encryption key (CEK) and the CEK is subsequently used to encrypt the plaintext. This mode supports multiple recipients. This setup is conceptually similar to Key Agreement with Key Wrapping.

When the alg value or enc value is set to any of algorithms registered by this specification then the 'epk' header parameter **MUST** be present, and it **MUST** be a JSON Web Key as defined in Section 4.4 of this document.

The "ek" member of an 'epk' will contain the base64url encoded "enc" value produced by the encapsulate operation of the HPKE KEM.

In all serializations, "ct" will be base64url encoded.

If the 'alg' header parameter is set to the "dir" value (as defined in Section 7), HPKE is used in Direct Encryption mode; otherwise, it is in Key Encryption mode.

Interested readers will observe this is due to all recipients using the same JWE Protected Header when JSON Serializations are used, as described in Section 7.2.1 of [RFC7516].

We provide the following table for additional clarity:

Name	Recipients	Serializations	Content Encryption Key	Similar to
Direct Encryption	1	Compact, JSON	Derived from HPKE	Direct Key Agreement
Key Encryption	1 or More	Compact, JSON	Encrypted by HPKE	Key Agreement with Key Wrapping

Table 1: JOSE HPKE Serializations and Modes

4.2. HPKE Encryption

The message encryption process is as follows.

1. The sending HPKE context is created by invoking invoking SetupBaseS() (Section 5.1.1 of [RFC9180]) with the recipient's public key "pkR" and "info". The HPKE specification defines the "info" parameter as a context information structure that is used to ensure that the derived keying material is bound to the context of the transaction. The SetupBaseS function will be called with the default value of an empty string for the 'info' parameter. This yields the context "sctxt" and an encapsulation key "enc".

There exist two cases of HPKE plaintext which need to be distinguished:

*In HPKE Direct Encryption mode, the plaintext "pt" passed into Seal is the content to be encrypted. Hence, there is no intermediate layer utilizing a CEK.

*In HPKE Key Encryption mode, the plaintext "pt" passed into Seal is the CEK. The CEK is a random byte sequence of length appropriate for the encryption algorithm. For example, AES-128-GCM requires a 16 byte key and the CEK would therefore be 16 bytes long.

4.3. HPKE Decryption

The recipient will create the receiving HPKE context by invoking SetupBaseR() (Section 5.1.1 of [RFC9180]) with "skR", "enc" (output of base64url decoded 'ek'), and "info" (empty string). This yields

the context "rctxt". The receiver then decrypts "ct" by invoking the Open() method on "rctxt" (Section 5.2 of [RFC9180]) with "aad", yielding "pt" or an error on failure.

The Open function will, if successful, decrypts "ct". When decrypted, the result will be either the CEK (when Key Encryption mode is used), or the content (if Direct Encryption mode is used). The CEK is the symmetric key used to decrypt the ciphertext.

4.4. Encapsulated JSON Web Keys

An encapsulated key is represented as JSON Web Key as described in { Section 4 of RFC7515 }.

The "kty" parameter **MUST** be "EK".

The "ek" parameter **MUST** be present, and **MUST** be the base64url encoded output of the encap operation defined for the HPKE KEM.

As described in { Section 4 of RFC7515 }, additional members can be present in the JWK; if not understood by implementations encountering them, they **MUST** be ignored.

This example demonstrates the representaton of an encapsulated key as a JWK.


```
{
   "kty": "EK",
   "ek": "BHpP-u5JKziyUpqxNQqb0apHx1ecH2UzcRlhHR4ngJVS__gNu21DqqgPweuPpj
}
```


4.4.1. HPKE Direct Encryption

This mode only supports a single recipient.

HPKE is employed to directly encrypt the plaintext, and the resulting ciphertext is included in the JWE ciphertext.

In HPKE Direct Encryption mode:

* The "epk" Header Parameter **MUST** be present, it **MUST** contain an Encapsulated JSON Web Key and it **MUST** occur only within the JWE Protected Header.

* The "alg" Header Parameter **MUST** be "dir", "enc" **MUST** be an HPKE algorithm from JSON Web Signature and Encryption Algorithms in [JOSE-IANA] and they **MUST** occur only within the JWE Protected Header.

*The JWE Ciphertext **MUST** be the resulting HPKE ciphertext ('ct' value) encoded using base64url.

*The JWE Initialization Vector value **MUST** be absent.

*The JWE Authentication Tag **MUST** be absent.

*The JWE Encrypted Key **MUST** be absent.

*The HPKE "aad" parameter **MUST** be set to the JWE Additional Authenticated Data encryption parameter defined in Step 14 of Section 5.1 of [RFC7516] as input.

The following example demonstrates the use of Direct Encryption with Compact Serialization:


```
eyJhbGciOiJkaXIiLCJlbmMiOiJIUEtFLUJhc2UtUDI1Ni1TSEEyNTYtQUVTMTI4R0NNIiwi
```


Figure 1: Direct Encryption with Compact Serialization

In the above example, the JWE Protected Header value is:


```
{
  "alg": "dir",
  "enc": "HPKE-Base-P256-SHA256-AES128GCM",
  "epk": {
    "kty": "EK",
    "ek": "BGNkjzt076bsRGj78aX5AzT_HEOJBbY9q2Zo_5e7tbK0aPqu4eT1WI16jvRlZ
  }
}
```

```
{
    "protected": "eyJhbGciOiJkaXIiLCJlbmMiOiJIUEtFLUJhc2UtUDI1Ni1TSEEyNT
    "ciphertext": "1ATsw0jshqPrv8CFcm9Rem9Wfi1Ygv30sozlRTtNNzcaaZ828GqP0
}
```


Figure 2: Direct Encryption with JSON Serialization

In the above example, the JWE Protected Header value is:


```
{
  "alg": "dir",
  "enc": "HPKE-Base-P256-SHA256-AES128GCM",
  "epk": {
    "kty": "EK",
    "ek": "BGNkjzt076bsRGj78aX5AzT_HEOJBbY9q2Zo_5e7tbK0aPqu4eT1WI16jvRlZ
  }
}
```

4.4.2. HPKE Key Encryption

This mode supports more than one recipient.

HPKE is used to encrypt the Content Encryption Key (CEK), and the resulting ciphertext is included in the JWE Encrypted Key. The plaintext will be encrypted using the CEK as explained in Step 15 of Section 5.1 of [RFC7516].

When there are multiple recipients, the sender **MUST** place the 'epk' parameter in the per-recipient unprotected header to indicate the use of HPKE. In this case, the 'enc' (Encryption Algorithm) Header Parameter **MUST** be a content encryption algorithm from JSON Web Signature and Encryption Algorithms in [JOSE-IANA], and it **MUST** be present in the JWE Protected Header. The integrity-protected 'enc' parameter provides protection against an attacker who manipulates the encryption algorithm in the 'enc' parameter. This attack is discussed in [I-D.draft-ietf-lamps-cms-cek-hkdf-sha256].

In HPKE Key Encryption mode:

* The JWE Encrypted Key **MUST** be the resulting HPKE ciphertext ('ct' value) encoded using base64url.

The following example demonstrates the use of Key Encryption with General JSON Serialization:

```
{
  "protected": "eyJlbmMiOiJBMTI4R0NNIn0",
  "ciphertext": "S0qqrM3xXPUavbmL9LQkgUKRBu8BZ7DQWoT-mdNIZVU-ip_V-fbMoki
  "iv": "AzaXpooLg3ZxEASQ",
  "aad": "8J-SgCBhYWQ",
  "tag": "S0omWw35S0H7tyEHsmGLDw",
  "recipients": [
    {
      "encrypted_key": "yDVZLsO7-ecy_GCgEluwn9U723TCHNAzeYRRQPOfpHM",
      "header": {
        "kid": "urn:ietf:params:oauth:jwk-thumbprint:sha-256:adjwW6fyyZ9
        "alg": "HPKE-Base-P256-SHA256-AES128GCM",
        "epk": {
          "kty": "EK",
          "ek": "BHpP-u5JKziyUpqxNQqb0apHx1ecH2UzcRlhHR4ngJVS__gNu21Dqqg
        }
      }
    },
    {
      "encrypted_key": "iS73TFqJ61gkmh4DHAXADx4wyftA7pnY",
      "header": {
        "kid": "urn:ietf:params:oauth:jwk-thumbprint:sha-256:D2FKlj9MTIQ
        "alg": "ECDH-ES+A128KW",
        "epk": {
          "kty": "EC",
          "crv": "P-256",
          "x": "nX6Y3DWC0olVe5H7-NkCzVDghsYSa_L9da3jzkHYkV8",
          "y": "wDshQdcaY0J08wx25V3ystQSNe_qjsCaaFeeRWJqcE0"
        }
      }
    }
  ]
}
```


Figure 3: Key Encryption (multiple recipient) General JSON Serialization

In the above example, the JWE Protected Header value is:


```
{
   "enc": "A128GCM"
}
```

```
{
  "protected": "eyJhbGciOiAiSFBLRS1CYXNlLVAyNTYtU0hBMjU2LUFFUzEyOEdDTSIs
  "encrypted_key": "zR0ArfrVVRQ9-X_heDU2riwx36QxLBffRrKAWU-tLC4",
  "iv": "o3v11Hw6gUxUN-pY",
  "ciphertext": "Ny-2IDGHMI3MzVsUAVMGNoKAZfoewTZ1dkAIBikPy4eZUfHW_LPhhKp
  "tag": "0sfzHJvxVoWt02EPxMTh8w"
}
```


Figure 4: Key Encryption (single recipient) Flattened JSON Serialization

In the above example, the JWE Protected Header value is:


```
{

  "alg": "HPKE-Base-P256-SHA256-AES128GCM",
  "enc": "A128GCM",
  "epk": {
    "kty": "EK",
    "ek": "BPRTKn8mQL4hN1ayokR8gkpTy5HQld4N0HXXB9cXtjUIQ37zsJDL7TugVkmD1
  }
}
```



```
eyJhbGciOiAiSFBLRS1CYXNlLVAyNTYtU0hBMjU2LUFFUzEyOEdDTSIsImVuYyI6IkExMjhH
```


Figure 5: Key Encryption (single recipient) Compact

In the above example, the JWE Protected Header value is:


```
{
  "alg": "HPKE-Base-P256-SHA256-AES128GCM",
  "enc": "A128GCM",
  "epk": {
    "kty": "EK",
    "ek": "BJ7rdNbkvwunssdju5WDkAazLaBX3IdcLRjy1RDSA9sij00jdybaHuQPTt6P3
  }
}
```


5. Ciphersuite Registration

This specification registers a number of ciphersuites for use with HPKE. A ciphersuite is a group of algorithms, often sharing component algorithms such as hash functions, targeting a security level. An HPKE ciphersuite, is composed of the following choices:

*HPKE Mode

*KEM Algorithm

*KDF Algorithm

*AEAD Algorithm

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

For readability the algorithm ciphersuites labels are built according to the following scheme:

```
HPKE-<Mode>-<KEM>-<KDF>-<AEAD>
```

The "Mode" indicator may be populated with the following values from Table 1 of [RFC9180]:

*"Base" refers to "mode_base" described in Section 5.1.1 of [RFC9180], which only enables encryption to the holder of a given KEM private key.

*"PSK" refers to "mode_psk", described in Section 5.1.2 of [RFC9180], which authenticates using a pre-shared key.

*"Auth" refers to "mode_auth", described in Section 5.1.3 of [RFC9180], which authenticates using an asymmetric key.

*"Auth_Psk" refers to "mode_auth_psk", described in Section 5.1.4 of [RFC9180], which authenticates using both a PSK and an asymmetric key.

For a list of ciphersuite registrations, please see Section 7.

6. Security Considerations

This specification is based on HPKE and the security considerations of [RFC9180] are therefore applicable also to this specification.

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. In this case JOSE constructs like JWS and JSON Web Tokens (JWTs) can be used to add authentication. HPKE also offers modes that offer authentication.

HPKE relies on a source of randomness to be available on the device. In Key Agreement with Key Wrapping mode, CEK has to be randomly generated and it **MUST** be ensured that the guidelines in [RFC8937] for random number generations are followed.

6.1. Plaintext Compression

Implementers are advised to review Section 3.6 of [RFC8725], which states: Compression of data **SHOULD NOT** be done before encryption, because such compressed data often reveals information about the plaintext.

6.2. Header Parameters

Implementers are advised to review Section 3.10 of [RFC8725], which comments on application processing of JWE Protected Headers. Additionally, Unprotected Headers can contain similar information which an attacker could leverage to mount denial of service, forgery or injection attacks.

6.3. Ensure Cryptographic Keys Have Sufficient Entropy

Implementers are advised to review Section 3.5 of [RFC8725], which provides comments on entropy requirements for keys. This guidance is relevant to both public and private keys used in both Key Encryption and Direct Encryption. Additionally, this guidance is applicable to content encryption keys used in Key Encryption mode.

6.4. Validate Cryptographic Inputs

Implementers are advised to review Section 3.4 of [RFC8725], which provides comments on the validation of cryptographic inputs. This guidance is relevant to both public and private keys used in both Key Encryption and Direct Encryption, specifically focusing on the structure of the public and private keys, as well as the 'ek' value. These inputs are crucial for the HPKE KEM operations.

6.5. Use Appropriate Algorithms

Implementers are advised to review Section 3.2 of [RFC8725], which comments on the selection of appropriate algorithms. This is guidance is relevant to both Key Encryption and Direct Encryption. When using Key Encryption, the strength of the content encryption algorithm should not be significantly different from the strengh of the Key Encryption algorithms used.

7. IANA Considerations

This document adds entries to [JOSE-IANA].

7.1. JSON Web Key Types

The following entry is added to the "JSON Web Key Types" registry:

*"kty" Parameter Value: "EK"

*Key Type Description: HPKE Encapsulated Key Type (See issue #18)

*JOSE Implementation Requirements: Optional

*Change Controller: IETF

*Specification Document(s): [[TBD: This RFC]]

7.2. JSON Web Key Parameters

The following entry is added to the "JSON Web Key Parameters" registry:

*Parameter Name: "ek"

*Parameter Description: Encapsulated Key

*Parameter Information Class: Public

*Used with "kty" Value(s): "EK"

*Specification Document(s): [[TBD: This RFC]]

7.3. JSON Web Signature and Encryption Algorithms

The following entries are added to the "JSON Web Signature and Encryption Algorithms" registry:

*Algorithm Name: HPKE-Base-P256-SHA256-AES128GCM

*Algorithm Description: Cipher suite for JOSE-HPKE in Base Mode that uses the DHKEM(P-256, HKDF-SHA256) KEM, the HKDF-SHA256 KDF and the AES-128-GCM AEAD.

*Algorithm Usage Location(s): "alg, enc"

*JOSE Implementation Requirements: Optional

*Change Controller: IETF

*Specification Document(s): [[TBD: This RFC]]

*Algorithm Analysis Documents(s): TODO

*Algorithm Name: HPKE-Base-P384-SHA384-AES256GCM

*Algorithm Description: Cipher suite for JOSE-HPKE in Base Mode that uses the DHKEM(P-384, HKDF-SHA384) KEM, the HKDF-SHA384 KDF, and the AES-256-GCM AEAD.

*Algorithm Usage Location(s): "alg, enc"

*JOSE Implementation Requirements: Optional

*Change Controller: IETF

*Specification Document(s): [[TBD: This RFC]]

*Algorithm Analysis Documents(s): TODO

*Algorithm Name: HPKE-Base-P521-SHA512-AES256GCM

*Algorithm Description: Cipher suite for JOSE-HPKE in Base Mode that uses the DHKEM(P-521, HKDF-SHA512) KEM, the HKDF-SHA512 KDF, and the AES-256-GCM AEAD.

*Algorithm Usage Location(s): "alg, enc"

*JOSE Implementation Requirements: Optional

*Change Controller: IETF

*Specification Document(s): [[TBD: This RFC]]

*Algorithm Analysis Documents(s): TODO

*Algorithm Name: HPKE-Base-X25519-SHA256-AES128GCM

*Algorithm Description: Cipher suite for JOSE-HPKE in Base Mode that uses the DHKEM(X25519, HKDF-SHA256) KEM, the HKDF-SHA256 KDF, and the AES-128-GCM AEAD.

*Algorithm Usage Location(s): "alg, enc"

*JOSE Implementation Requirements: Optional

*Change Controller: IETF

*Specification Document(s): [[TBD: This RFC]]

*Algorithm Analysis Documents(s): TODO

*Algorithm Name: HPKE-Base-X25519-SHA256-ChaCha20Poly1305

*Algorithm Description: Cipher suite for JOSE-HPKE in Base Mode that uses the DHKEM(X25519, HKDF-SHA256) KEM, the HKDF-SHA256 KDF, and the ChaCha20Poly1305 AEAD.

*Algorithm Usage Location(s): "alg, enc"

*JOSE Implementation Requirements: Optional

*Change Controller: IETF

*Specification Document(s): [[TBD: This RFC]]

*Algorithm Analysis Documents(s): TODO

*Algorithm Name: HPKE-Base-X448-SHA512-AES256GCM

*Algorithm Description: Cipher suite for JOSE-HPKE in Base Mode that uses the DHKEM(X448, HKDF-SHA512) KEM, the HKDF-SHA512 KDF, and the AES-256-GCM AEAD.

*Algorithm Usage Location(s): "alg, enc"

*JOSE Implementation Requirements: Optional

*Change Controller: IETF

*Specification Document(s): [[TBD: This RFC]]

*Algorithm Analysis Documents(s): TODO

*Algorithm Name: HPKE-Base-X448-SHA512-ChaCha20Poly1305

*Algorithm Description: Cipher suite for JOSE-HPKE in Base Mode that uses the DHKEM(X448, HKDF-SHA512) KEM, the HKDF-SHA512 KDF, and the ChaCha20Poly1305 AEAD.

*Algorithm Usage Location(s): "alg, enc"

*JOSE Implementation Requirements: Optional

*Change Controller: IETF

*Specification Document(s): [[TBD: This RFC]]

*Algorithm Analysis Documents(s): TODO

7.4. JSON Web Signature and Encryption Header Parameters

The following entries are added to the "JSON Web Key Parameters" registry:

*Parameter Name: "psk_id"

*Parameter Description: A key identifier (kid) for the pre-shared key as defined in { Section 5.1.1 of RFC9180 }

*Parameter Information Class: Public

*Used with "kty" Value(s): *

*Change Controller: IETF

*Specification Document(s): [[This specification]]

*Parameter Name: "auth_kid"

*Parameter Description: A key identifier (kid) for the asymmetric key as defined in { Section 5.1.4 of RFC9180 }

*Parameter Information Class: Public

*Used with "kty" Value(s): *

*Change Controller: IETF

*Specification Document(s): [[This specification]]

8. References

8.1. Normative References

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

[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>.

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

[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>.

[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>.

[RFC9180] Barnes, R., Bhargavan, K., Lipp, B., and C. Wood, "Hybrid Public Key Encryption", RFC 9180, DOI 10.17487/RFC9180, February 2022, <https://www.rfc-editor.org/rfc/rfc9180>.

8.2. Informative References

[HPKE-IANA] IANA, "Hybrid Public Key Encryption (HPKE) IANA Registry", October 2023, <https://www.iana.org/assignments/hpke/hpke.xhtml>.

[I-D.draft-ietf-lamps-cms-cek-hkdf-sha256] Housley, R., "Encryption Key Derivation in the Cryptographic Message Syntax (CMS) using HKDF with SHA-256", Work in Progress, Internet-Draft, draft-ietf-lamps-cms-cek-hkdf-sha256-01, 18 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-cms-cek-hkdf-sha256-01>.

[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-07, 22 October 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-cose-hpke-07>.

[RFC8937] Cremers, C., Garratt, L., Smyshlyaev, S., Sullivan, N., and C. Wood, "Randomness Improvements for Security Protocols", RFC 8937, DOI 10.17487/RFC8937, October 2020, <https://www.rfc-editor.org/rfc/rfc8937>.

Acknowledgments

This specification leverages text from [I-D.ietf-cose-hpke]. We would like to thank Matt Chanda, Ilari Liusvaara, Aaron Parecki and Filip Skokan for their feedback.

Document History

-07

*Corrected some working group and document titles.

*Added Document History section.

-06

*Use existing "alg": "dir" value for HPKE Direct Encryption mode.

-05

*Defined both Integrated Encryption mode and Key Encryption mode.

*Aligned choices more closely with those of draft-ietf-cose-hpke.

-04

*Revamped Introduction and Overview sections.

*Removed use of the zip parameter.

*Populated IANA registries.

-03

*Described key encryption.

-02

*Removed Post-Quantum Considerations.

-01

*Described direct key agreement mode and key agreement with key wrapping mode.

*Improved example.

-00

*Initial version.

Authors' Addresses

Tirumaleswar Reddy
Nokia
Bangalore
Karnataka
India

Email: kondtir@gmail.com

Hannes Tschofenig
Austria

Email: hannes.tschofenig@gmx.net

Aritra Banerjee
Nokia
Munich

Germany

Email: aritra.banerjee@nokia.com

Orie Steele
Transmute
United States

Email: orie@transmute.industries

Michael B. Jones
Self-Issued Consulting
United States

Email: michael_b_jones@hotmail.com
URI: https://self-issued.info/