Workgroup: COSE
Internet-Draft: draft-ietf-cose-hpke-07
Published: 22 October 2023
Intended Status: Standards Track
Expires: 24 April 2024
Authors: H. Tschofenig O. Steele, Ed. D. Ajitomi
Transmute
L. Lundblade
Security Theory LLC

Use of Hybrid Public-Key Encryption (HPKE) with CBOR Object Signing and Encryption (COSE)

Abstract

This specification defines hybrid public-key encryption (HPKE) for use with CBOR Object Signing and Encryption (COSE). 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 COSE is provided by COSE-native security mechanisms or by one of the authenticated variants of HPKE.

This document defines the use of the HPKE with COSE.

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 24 April 2024.

Table of Contents

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. HPKE utilizes a non-interactive ephemeral-static Diffie-Hellman exchange to establish a shared secret. The motivation for standardizing a public key encryption scheme is explained in the introduction of [RFC9180].

The HPKE specification provides a variant of public key encryption of arbitrary-sized plaintexts for a recipient public key. It also includes three authenticated variants, including one that authenticates possession of a pre-shared key, one that authenticates possession of a key encapsulation mechanism (KEM) private key, and one that authenticates possession of both a pre-shared key and a KEM private key.

This specification utilizes HPKE as a foundational building block and carries the output to COSE ([RFC9052], [RFC9053]).

2. Conventions and Terminology

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.

This specification uses the following abbreviations and terms:

- *Content-encryption key (CEK), a term defined in CMS [RFC2630].
- *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].

3. HPKE for COSE

3.1. Overview

This specification supports two uses of HPKE in COSE, namely

* HPKE in a single recipient setup. This use case utilizes a one layer COSE structure. Section 3.1.1 provides the details.

* HPKE in a multiple recipient setup. This use case requires a two layer COSE structure. Section 3.1.2 provides the details. While it is possible to support the single recipient use case with a two layer structure, the single layer setup is more efficient.

In both cases a new COSE header parameter, called 'encapsulated_key', is used to convey the content of the enc structure defined in the HPKE specification. "Enc" represents the serialized public key.

For use with HPKE the 'encapsulated_key' header parameter MUST be present in the unprotected header parameter and MUST contain the encapsulated key, which is output of the HPKE KEM, and it is a bstr.

3.1.1. Single Recipient / One Layer Structure

With the one layer structure the information carried inside the COSE_recipient structure is embedded inside the COSE_Encrypt0.

HPKE is used to directly encrypt the plaintext and the resulting ciphertext is either included in the COSE_Encrypt0 or is detached. If a payload is transported separately then it is called "detached content". A nil CBOR object is placed in the location of the ciphertext. See Section 5 of [RFC9052] for a description of detached payloads.

The sender MUST set the alg parameter in the protected header, which indicates the use of HPKE.

The sender MUST place the 'encapsulated_key' parameter into the unprotected header. Although the use of the 'kid' parameter in COSE_Encrypt0 is discouraged by RFC 9052, this profile allows the use of the 'kid' parameter (or other parameters) to identify the static recipient public key used by the sender. If the COSE_Encrypt0 contains the 'kid' then the recipient may use it to select the appropriate private key.

HPKE defines an API and this API uses an "aad" parameter as input. When COSE_Encrypt0 is used then there is no AEAD function executed by COSE natively and HPKE offers this functionality.

The "aad" parameter provided to the HPKE API is constructed as follows (and the design has been re-used from [RFC9052]):

```
Enc_structure = [
    context : "Encrypt0",
    protected : empty_or_serialized_map,
    external_aad : bstr
]

empty_or_serialized_map = bstr .cbor header_map / bstr .size 0
```

The protected field in the Enc_structure contains the protected attributes from the COSE_Encrypt0 structure at layer 0, encoded in a bstr type.

The external_aad field in the Enc_structure contains the Externally Supplied Data described in Section 4.3 and Section 5.3 in RFC 9052. If this field is not supplied, it defaults to a zero-length byte string.

The HPKE APIs also use an "info" parameter as input and the details are provided in Section 3.2.

Figure 1 shows the COSE_Encrypt0 CDDL structure.

```
COSE_Encrypt0_Tagged = #6.16(COSE_Encrypt0)

; Layer 0
COSE_Encrypt0 = [
    Headers,
    ciphertext : bstr / nil,
]
```

Figure 1: CDDL for HPKE-based COSE_Encrypt0 Structure

The COSE_Encrypt0 MAY be tagged or untagged.

An example is shown in Section 5.1.

3.1.2. Multiple Recipients / Two Layer Structure

With the two layer structure the HPKE information is conveyed in the COSE_recipient structure, i.e. one COSE_recipient structure per recipient.

In this approach the following layers are involved:

* Layer 0 (corresponding to the COSE_Encrypt structure) contains the content (plaintext) encrypted with the CEK. This ciphertext

MAY be detached. If not detached, then it is included in the COSE_Encrypt structure.

* Layer 1 (corresponding to a recipient structure) contains parameters needed for HPKE to generate a shared secret used to encrypt the CEK. This layer conveys the encrypted CEK in the encCEK structure. The protected header MUST contain the HPKE alg parameter and the unprotected header MUST contain the 'encapsulated_key' parameter. The unprotected header MAY contain the kid parameter to identify the static recipient public key the sender has been using with HPKE.

This two-layer structure is used to encrypt content that can also be shared with multiple parties at the expense of a single additional encryption operation. As stated above, the specification uses a CEK to encrypt the content at layer 0.

The COSE_recipient structure, shown in Figure 2, is repeated for each recipient.

```
COSE_Encrypt_Tagged = #6.96(COSE_Encrypt)

/ Layer 0 /
COSE_Encrypt = [
  Headers,
  ciphertext : bstr / nil,
  recipients : + COSE_recipient
]

/ Layer 1 /
COSE_recipient = [
  protected   : bstr .cbor header_map,
  unprotected : header_map,
  encCEK      : bstr,
]

header_map = {
  Generic_Headers,
  * label => values,
}
```

Figure 2: CDDL for HPKE-based COSE_Encrypt Structure

The COSE_Encrypt MAY be tagged or untagged.

An example is shown in Section 5.2.

3.2. Info Parameter

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.

This section provides a suggestion for constructing the info structure. HPKE leaves the info parameter for these two functions as optional. Application profiles of this specification MAY populate the fields of the COSE_KDF_Context structure or MAY use a different structure as input to the "info" parameter. If no content for the "info" parameter is not supplied, it defaults to a zero-length byte string.

This specification re-uses the context information structure defined in [RFC9053] as a foundation for the info structure. This payload becomes the content of the info parameter for the HPKE functions, when utilized. For better readability of this specification the COSE_KDF_Context structure is repeated in Figure 3.

```
PartyInfo = (
    identity : bstr / nil,
    nonce : bstr / int / nil,
    other : bstr / nil
)

COSE_KDF_Context = [
    AlgorithmID : int / tstr,
    PartyUInfo : [ PartyInfo ],
    PartyVInfo : [ PartyInfo ],
    SuppPubInfo : [
        keyDataLength : uint,
        protected : empty_or_serialized_map,
        ? other : bstr
    ],
    ? SuppPrivInfo : bstr
]
```

Figure 3: COSE_KDF_Context Data Structure as 'info' Parameter for HPKE

4. Ciphersuite Registration

This specification registers a number of ciphersuites for use with HPKE. A ciphersuite is thereby a combination of several algorithm configurations:

- HPKE Mode
- KEM algorithm

*KDF algorithm

*AEAD algorithm

The "KEM", "KDF", and "AEAD" values are conceptually taken from the HPKE IANA registry [HPKE-IANA]. Hence, COSE-HPKE cannot use a algorithm combination that is not already available with HPKE.

For better readability of the algorithm combination ciphersuites labels are build according to the following scheme:

HPKE-<Version>-<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. The following table summarizes the relationship between the ciphersuites registered in this document and the values registered in the HPKE IANA registry [HPKE-IANA].

COSE-HPKE Cipher Suite Label	HPKE KEM	HPKE KDF	HPKE AEAD
HPKE-Base-P256-SHA256-AES128GCM	0x10	0x1	0x1
HPKE-Base-P256-SHA256-ChaCha20Poly1305	0x10	0x1	0x3
HPKE-Base-P384-SHA384-AES256GCM	0x11	0x2	0x2
HPKE-Base-P384-SHA384-ChaCha20Poly1305	0x11	0x2	0x3
HPKE-Base-P521-SHA512-AES256GCM	0x12	0x3	0x2
HPKE-Base-P521-SHA512-ChaCha20Poly1305	0x12	0x3	0x3
HPKE-Base-X25519-SHA256-AES128GCM	0x20	0x1	0x1
HPKE-Base-X25519-SHA256-ChaCha20Poly1305	0x20	0x1	0x3
HPKE-Base-X448-SHA512-AES256GCM	0x21	0x3	0x2
HPKE-Base-X448-SHA512-ChaCha20Poly1305	0x21	0x3	0x3
HPKE-Base-X25519Kyber768-SHA256-AES256GCM	0x30	0x1	0x2
HPKE-Base-X25519Kyber768-SHA256-ChaCha20Poly1305	0x30	0x1	0x3
HPKE-Base-CP256-SHA256-ChaCha20Poly1305	0x13	0x1	0x3
HPKE-Base-CP256-SHA256-AES128GCM	0x13	0x1	0x1
HPKE-Base-CP521-SHA512-ChaCha20Poly1305	0x15	0x3	0x3
HPKE-Base-CP521-SHA512-AES256GCM	0x15	0x3	0x2

Note that the last four entries in the table refer to the compact encoding of the public keys defined in [I-D.irtf-cfrg-dnhpke].

As the list indicates, the ciphersuite labels have been abbreviated at least to some extend to maintain the tradeoff between readability and length.

5. Examples

This section provides a set of examples that shows all COSE message types (COSE_Encrypt0, COSE_Encrypt and COSE_MAC) to which the COSE-HPKE can be applied. Each example includes the following information that can be used to check the interoperability of COSE-HPKE implementations:

- *plaintext: Original data of the encrypted payload.
- *external_aad: Externally supplied AAD.
- *skR: A recipient private key.
- *skE: An ephemeral sender private key paired with the encapsulated_key.

5.1. Single Recipient / One Layer Example

This example assumes that a sender wants to communicate an encrypted payload to a single recipient in the most efficient way.

An example of the COSE_Encrypt0 structure using the HPKE scheme is shown in Figure 4. Line breaks and comments have been inserted for better readability.

This example uses the following:

- *alg: HPKE-Base-P256-SHA256-AES128GCM
- *plaintext: "This is the content."
- *external_aad: "COSE-HPKE app"
- *skR:
  h'57c92077664146e876760c9520d054aa93c3afb04e306705db6090308507b4d3'
- *skE:
  h'42dd125eefc409c3b57366e721a40043fb5a58e346d51c133128a77237160218'

```
16([
    / alg = HPKE-Base-P256-SHA256-AES128GCM (Assumed: 35) /
    h'a1011823',
    {
        / kid /
        4: h'3031',
        / encapsulated_key /
        -4: h'045df24272faf43849530db6be01f42708b3c3a9
              df8e268513f0a996ed09ba7840894a3fb946cb28
              23f609c59463093d8815a7400233b75ca8ecb177
              54d241973e',
    },
    / encrypted plaintext /
    h'35aa3d98739289b83751125abe44e3b977e4b9abbf2c8cfaade
      b15f7681eef76df88f096',
])
```

Figure 4: COSE_Encrypt0 Example for HPKE

5.2. Multiple Recipients / Two Layer

In this example we assume that a sender wants to transmit a payload to two recipients using the two-layer structure. Note that it is possible to send two single-layer payloads, although it will be less efficient.

5.2.1. COSE_Encrypt

An example of the COSE_Encrypt structure using the HPKE scheme is shown in Figure 5. Line breaks and comments have been inserted for better readability.

This example uses the following:

* Encryption alg: AES-128-GCM

* plaintext: "This is the content."

* detatched ciphertext:
  h'cc168c4e148c52a83010a75250935a47ccb8682deebcef8fce5d60c161e849f53a2dc664'

* kid:"01"

  - alg: HPKE-Base-P256-SHA256-AES128GCM

  - external_aad: "COSE-HPKE app"

  - skR:
    h'57c92077664146e876760c9520d054aa93c3afb04e306705db6090308507b4d3'

  - skE:
    h'97ad883f949f4cdcb1301b9446950efd4eb519e16c4a3d78304eec832692f9f6'

* kid:"02"

  - alg: HPKE-Base-X25519-SHA256-CHACHA20POLY1305

  - external_aad: "COSE-HPKE app"

  - skR:
    h'bec275a17e4d362d0819dc0695d89a73be6bf94b66ab726ae0b1afe3c43f41ce'

  - skE:
    h'b8ed3f4df56c230e36fa6620a47f24d08856d242ea547c5521ff7bd69af8fd6f'

```
96_0([
    / alg = AES-128-GCM (1) /
    h'a10101',
    {
        / iv /
        5: h'b3fb95dde18c6f90a9f0ae55',
    },
    / detached ciphertext /
    null,
    [
        [
            / alg = HPKE-Base-P256-SHA256-AES128GCM (Assumed: 35) /
            h'a1011823',
            {
                / kid /
                4: h'3031',
                / encapsulated_key /
                -4: h'04d97b79486fe2e7b98fb1bd43
                      c4faee316ff38d28609a1cf568
                      40a809298a91e601f1cc0c2ba4
                      6cb67b41f4651b769cafd9df78
                      e58aa7f5771291bd4f0f420ba6',
            },
            / ciphertext containing encrypted CEK /
            h'24450f54ae93375351467d17aa7a795cfede2
              c03eced1ad21fcb7e7c2fe64397',
        ],
        [
            / alg = HPKE-Base-X25519-SHA256-CHACHA20POLY1305 (Assumed: 4
            h'a101182a',
            {
                / kid /
                4: h'3032',
                / encapsulated_key /
                -4: h'd1afbdc95b0e735676f6bca34f
                      be50f2822259ac09bfc3c500f1
                      4a05de9b2833',
            },
            / ciphertext containing encrypted CEK /
            h'079b443ec6dfcda6a5f8748aff3875146a8ed
              40359e1279b545166385d8d9b59',
        ],
    ],
])
```

Figure 5: COSE_Encrypt Example for HPKE

To offer authentication of the sender the payload in Figure 5 is signed with a COSE_Sign1 wrapper, which is outlined in Figure 6. The payload in Figure 6 is meant to contain the content of Figure 5.

```
18(
  [
    / protected / h'a10126' / {
            \ alg \ 1:-7 \ ECDSA 256 \
          } / ,
    / unprotected / {
          / kid / 4:'sender@example.com'
        },
    / payload /     h'AA19...B80C',
    / signature /   h'E3B8...25B8'
  ]
)
```

Figure 6: COSE_Encrypt Example for HPKE

5.2.2. COSE_MAC

An example of the COSE_MAC structure using the HPKE scheme is shown in Figure 7.

This example uses the following:

- *MAC alg: HMAC 256/256
- *payload: "This is the content."
- *kid:"01"
  - -alg: HPKE-Base-P256-SHA256-AES128GCM
  - -external_aad: "COSE-HPKE app"
  - -skR: h'57c92077664146e876760c9520d054aa93c3afb04e306705db6090308507b4d3'
  - -skE: h'e5dd9472b5807636c95be0ba2575020ba91cbb3561b52be141da89678c664307'
- *kid:"02"
  - -alg: HPKE-Base-X25519-SHA256-CHACHA20POLY1305
  - -external_aad: "COSE-HPKE app"

```
-skR:
 h'bec275a17e4d362d0819dc0695d89a73be6bf94b66ab726ae0b1afe3c43f41ce'

-skE:
 h'78a49d7af71b5244498e943f361aa0250184afc48b8098a68ae97ccd2cd7e56f'
```

```
97_0([
    / alg = HMAC 256/256 (5) /
    h'a10105',
    {},
    / payload = 'This is the content.' /
    h'546869732069732074686520636f6e74656e742e',
    / tag /
    h'5cdcf6055fcbdb53b4001d8fb88b2a46b200ed28e1ed77e16ddf43fb3cac3a98',
    [
        [
            / alg = HPKE-Base-P256-SHA256-AES128GCM (Assumed: 35) /
            h'a1011823',
            {
                / kid = '01' /
                4: h'3031',
                / encapsulated_key /
                -4: h'043ac21632e45e1fbd733f002a
                      621aa4f3d94737adc395d5a7cb
                      6e9554bd1ad273aec991493786
                      d72616d9759bf8526e6e20c1ed
                      c41ba5739f2b2e441781aa0eb4',
            },
            / ciphertext containing encrypted MAC key /
            h'5cee2b4235a7ff695164f7a8d1e79ccf3ca3d
              e8b22f3592626020a95b2a8d3fb4d7aa7fe37
              432426ee70073a368f29d1',
        ],
        [
            / alg = HPKE-Base-X25519-SHA256-CHACHA20POLY1305 (Assumed: 4
            h'a101182a',
            {
                / kid = '02' /
                4: h'3032',
                / encapsulated_key /
                -4: h'02cffacc60def3bb3d0a1c3661
                      227c9de8dc2b1d3939dd2c07d4
                      49ebb0bba324',
            },
            / ciphertext containing encrypted MAC key /
            h'3f5b8b60271d5234dbea554dc1461d0239e9f
              4589f6415e8563b061dbcb37795a616111b78
              2b4c589b534309327ffadc',
        ],
    ],
])
```

Figure 7: COSE_MAC Example for HPKE

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 COSE makes the same assumptions. Hence, some form of public key distribution mechanism is assumed to exist but outside the scope of this document.

HPKE relies on a source of randomness to be available on the device. Additionally, with the two layer structure the CEK is randomly generated and it MUST be ensured that the guidelines in [RFC8937] for random number generations are followed.

HPKE in Base mode does not offer authentication as part of the HPKE KEM. In this case COSE constructs like COSE_Sign, COSE_Sign1, COSE_MAC, or COSE_MAC0 can be used to add authentication. HPKE also offers modes that offer authentication.

If COSE_Encrypt or COSE_Encrypt0 is used with a detached ciphertext then the subsequently applied integrity protection via COSE_Sign, COSE_Sign1, COSE_MAC, or COSE_MAC0 does not cover this detached ciphertext. Implementers MUST ensure that the detached ciphertext also experiences integrity protection. This is, for example, the case when an AEAD cipher is used to produce the detached ciphertext but may not be guaranteed by non-AEAD ciphers.

7. IANA Considerations

This document requests IANA to add new values to the 'COSE Algorithms' and to the 'COSE Header Parameters' registries.

7.1. COSE Algorithms Registry

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

*Value: TBD1 (Assumed: 35)

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

*Capabilities: [kty]

*Change Controller: IESG

*Reference: [[TBD: This RFC]]

*Recommended: Yes

*Name: HPKE-Base-P256-SHA256-ChaCha20Poly1305

*Value: TBD2 (Assumed: 36)

*Description: Cipher suite for COSE-HPKE in Base Mode that uses the DHKEM(P-256, HKDF-SHA256) KEM, the HKDF-SHA256 KDF and the ChaCha20Poly1305 AEAD.

*Capabilities: [kty]

*Change Controller: IESG

*Reference: [[TBD: This RFC]]

*Recommended: Yes

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

*Value: TBD3 (Assumed: 37)

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

*Capabilities: [kty]

*Change Controller: IESG

*Reference: [[TBD: This RFC]]

*Recommended: Yes

*Name: HPKE-Base-P384-SHA384-ChaCha20Poly1305

*Value: TBD4 (Assumed: 38)

*Description: Cipher suite for COSE-HPKE in Base Mode that uses the DHKEM(P-384, HKDF-SHA384) KEM, the HKDF-SHA384 KDF, and the ChaCha20Poly1305 AEAD.

*Capabilities: [kty]

*Change Controller: IESG

*Reference: [[TBD: This RFC]]

*Recommended: Yes

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

*Value: TBD5 (Assumed: 39)

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

*Capabilities: [kty]

*Change Controller: IESG

*Reference: [[TBD: This RFC]]

*Recommended: Yes

*Name: HPKE-Base-P521-SHA512-ChaCha20Poly1305

*Value: TBD6 (Assumed: 40)

*Description: Cipher suite for COSE-HPKE in Base Mode that uses the DHKEM(P-521, HKDF-SHA512) KEM, the HKDF-SHA512 KDF, and the ChaCha20Poly1305 AEAD.

*Capabilities: [kty]

*Change Controller: IESG

*Reference: [[TBD: This RFC]]

*Recommended: Yes

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

*Value: TBD7 (Assumed: 41)

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

*Capabilities: [kty]

*Change Controller: IESG

*Reference: [[TBD: This RFC]]

*Recommended: Yes

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

*Value: TBD8 (Assumed: 42)

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

*Capabilities: [kty]

*Change Controller: IESG

*Reference: [[TBD: This RFC]]

*Recommended: Yes

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

*Value: TBD9 (Assumed: 43)

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

*Capabilities: [kty]

*Change Controller: IESG

*Reference: [[TBD: This RFC]]

*Recommended: Yes

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

*Value: TBD10 (Assumed: 44)

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

*Capabilities: [kty]

*Change Controller: IESG

*Reference: [[TBD: This RFC]]

*Recommended: Yes

*Name: HPKE-Base-X25519Kyber768-SHA256-AES256GCM

*Value: TBD11 (Assumed: 250)

*Description: Cipher suite for COSE-HPKE in Base Mode that uses the X25519Kyber768Draft00 KEM, the HKDF-SHA256 KDF, and the AES-256-GCM AEAD.

*Capabilities: [kty]

*Change Controller: IESG

*Reference: [[TBD: This RFC]]

*Recommended: No

*Name: HPKE-Base-X25519Kyber768-SHA256-ChaCha20Poly1305

*Value: TBD12 (Assumed: 251)

*Description: Cipher suite for COSE-HPKE in Base Mode that uses
 the X25519Kyber768Draft00 KEM, the HKDF-SHA256 KDF, and the
 ChaCha20Poly1305 AEAD.

*Capabilities: [kty]

*Change Controller: IESG

*Reference: [[TBD: This RFC]]

*Recommended: No

*Name: HPKE-Base-CP256-SHA256-ChaCha20Poly1305

*Value: TBD13 (Assumed: 45)

*Description: Cipher suite for COSE-HPKE in Base Mode that uses
 the DHKEM(CP-256, HKDF-SHA256) KEM, the HKDF-SHA256 KDF and the
 ChaCha20Poly1305 AEAD.

*Capabilities: [kty]

*Change Controller: IESG

*Reference: [[TBD: This RFC]]

*Recommended: Yes

*Name: HPKE-Base-CP521-SHA512-ChaCha20Poly1305

*Value: TBD14 (Assumed: 46)

*Description: Cipher suite for COSE-HPKE in Base Mode that uses
 the DHKEM(CP-521, HKDF-SHA512) KEM, the HKDF-SHA512 KDF, and the
 ChaCha20Poly1305 AEAD.

*Capabilities: [kty]

*Change Controller: IESG

*Reference: [[TBD: This RFC]]

*Recommended: Yes

*Name: HPKE-Base-CP256-SHA256-AES128GCM

*Value: TBD15 (Assumed: 47)

*Description: Cipher suite for COSE-HPKE in Base Mode that uses the DHKEM(CP-256, HKDF-SHA256) KEM, the HKDF-SHA256 KDF and the AES128GCM AEAD.

*Capabilities: [kty]

*Change Controller: IESG

*Reference: [[TBD: This RFC]]

*Recommended: Yes

*Name: HPKE-Base-CP521-SHA512-AES256GCM

*Value: TBD16 (Assumed: 47)

*Description: Cipher suite for COSE-HPKE in Base Mode that uses the DHKEM(CP-521, HKDF-SHA512) KEM, the HKDF-SHA512 KDF, and the AES256GCM AEAD.

*Capabilities: [kty]

*Change Controller: IESG

*Reference: [[TBD: This RFC]]

*Recommended: Yes

7.2. COSE Header Parameters

*Name: encapsulated_key

*Label: TBDX (Assumed: -4)

*Value type: bstr

*Value Registry: N/A

*Description: HPKE encapsulated key

*Reference: [[This specification]]

8. References

8.1. Normative References

[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>.

[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>.

[RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://www.rfc-editor.org/rfc/rfc9052>.

[RFC9053] Schaad, J., "CBOR Object Signing and Encryption (COSE): Initial Algorithms", RFC 9053, DOI 10.17487/RFC9053, August 2022, <https://www.rfc-editor.org/rfc/rfc9053>.

[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.irtf-cfrg-dnhpke]
Harkins, D., "Deterministic Nonce-less Hybrid Public Key Encryption", Work in Progress, Internet-Draft, draft-irtf-cfrg-dnhpke-03, 19 October 2023, <https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-dnhpke-03>.

[RFC2630] Housley, R., "Cryptographic Message Syntax", RFC 2630, DOI 10.17487/RFC2630, June 1999, <https://www.rfc-editor.org/rfc/rfc2630>.

[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>.

Appendix A. Contributors

We would like thank the following individuals for their contributions to the design of embedding the HPKE output into the COSE structure following a long and lively mailing list discussion:

*Richard Barnes

*Ilari Liusvaara

Finally, we would like to thank Russ Housley and Brendan Moran for their contributions to the draft as co-authors of initial versions.

Appendix B. Acknowledgements

We would like to thank John Mattsson, Mike Prorock, Michael Richardson, and Goeran Selander for their review feedback.

Authors' Addresses

Hannes Tschofenig
Austria

Email: hannes.tschofenig@gmx.net

Orie Steele (editor)
Transmute
United States

Email: orie@transmute.industries

Daisuke Ajitomi
Japan

Email: dajiaji@gmail.com

Laurence Lundblade
Security Theory LLC
United States

Email: lgl@securitytheory.com