COSE
Internet-Draft
Intended status: Standards Track
Expires: 13 July 2026

H. Tschofenig
H-BRS
O. Steele, Ed.
Tradeverifyd
D. Ajitomi
bibital
L. Lundblade
Security Theory LLC
M. Jones
Self-Issued Consulting
9 January 2026

Use of Hybrid Public-Key Encryption (HPKE) with CBOR Object Signing and Encryption (COSE)
draft-ietf-cose-hpke-20

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 is a general encryption framework utilizing an asymmetric key encapsulation mechanism (KEM), a key derivation function (KDF), and an Authenticated Encryption with Associated Data (AEAD) algorithm.

This document defines the use of HPKE with COSE. Authentication for HPKE in COSE is provided by COSE-native security mechanisms or by the pre-shared key authenticated variant of HPKE.

Discussion Venues

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

Discussion of this document takes place on the CBOR Object Signing and Encryption Working Group mailing list (cose@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/cose/.

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

Status of This Memo

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

This Internet-Draft will expire on 13 July 2026.

Copyright Notice

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.

This document defines the use of HPKE with COSE ([RFC9052], [RFC9053]) with the single-shot APIs defined in Section 6 of [RFC9180]. Multiple invocations of Open() / Seal() on the same context, as discussed in Section 9.7.1 of [RFC9180] are not supported.

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 [RFC5652].

* 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 modes of using HPKE in COSE, namely:

* HPKE Integrated Encryption mode, where HPKE is used to encrypt the plaintext. This mode can only be used with a single recipient. Section 3.2 provides the details.

* 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. Section 3.3 provides the details.

Distinct algorithm identifiers are defined and registered that are specific to each COSE HPKE mode so that they are fully specified, as required by [RFC9864]. Algorithm identifiers MUST only be used in the COSE HPKE mode that is specified for them.

In both cases, the new COSE header parameter 'ek' MUST be present. It contains the encapsulated KEM shared secret. The value of this parameter MUST be the 'enc' value output by the HPKE Seal() operation, as defined in Section 6.1 of [RFC9180]. The 'ek' header parameter MUST be encoded as a CBOR byte string.

HPKE defines several authentication modes, as described in Table 1 of [RFC9180]. In COSE HPKE, only 'mode_base' and 'mode_psk' are supported. The mode is 'mode_psk' if the 'psk_id' header parameter is present; otherwise, the mode defaults to 'mode_base'. 'mode_base' is described in Section 5.1.1 of [RFC9180], which only enables encryption to the holder of a given KEM private key. 'mode_psk' is described in Section 5.1.2 of [RFC9180], which authenticates using a pre-shared key.

3.2. HPKE Integrated Encryption Mode

This mode applies if the COSE_Encrypt0 structure uses a COSE-HPKE algorithm and has no recipient structure(s).

Because COSE-HPKE supports header protection, if the 'alg' parameter is present, it MUST be included in the protected header and MUST be a COSE-HPKE algorithm.

Although the use of the 'kid' parameter in COSE_Encrypt0 is discouraged by RFC 9052, this document RECOMMENDS the use of the 'kid' parameter (or other parameters) to explicitly identify the static recipient public key used by the sender. If the COSE_Encrypt0 structure includes a 'kid' parameter, the recipient MAY use it to select the corresponding private key.

When encrypting, the inputs to the HPKE Seal operation are set as follows:

* kem_id: Depends on the COSE-HPKE algorithm used.

* pkR: The recipient public key, converted into an HPKE public key.

* kdf_id: Depends on the COSE-HPKE algorithm used.

* info: Defaults to the empty string; externally provided information MAY be used instead.

* aad: MUST contain the byte string for the authenticated data structure according to the steps defined in Section 5.3 of RFC 9052. For the Integrated Encryption mode the context string will be "Encrypt0". Externally provided AAD information MAY be provided and MUST be passed into the Enc_structure via the external_aad field.

* aead_id: Depends on the COSE-HPKE algorithm used.

* pt: The raw message plaintext.

The outputs are used as follows:

* enc: MUST be placed raw into the 'ek' (encapsulated key) parameter in the unprotected bucket.

* ct: MUST be used as layer ciphertext. If not using detached content, this is directly placed as ciphertext in COSE_Encrypt0 structure. Otherwise, it is transported separately and the ciphertext field is nil. See Section 5 of [RFC9052] for a description of detached payloads.

If 'mode_psk' has been selected, then the 'psk_id' parameter MUST be present. If 'mode_base' has been chosen, then the 'psk_id' parameter MUST NOT be present.

When decrypting, the inputs to the HPKE Open operation are set as follows:

* kem_id: Depends on the COSE-HPKE algorithm used.

* skR: The recipient private key, converted into an HPKE private key.

* kdf_id: Depends on the COSE-HPKE algorithm used.

* aead_id: Depends on the COSE-HPKE algorithm used.

* info: Defaults to the empty string; externally provided information MAY be used instead.

* aad: MUST contain the byte string for the authenticated data structure according to the steps defined in Section 5.3 of RFC 9052. For the Integrated Encryption mode the context string will be "Encrypt0". Externally provided AAD information MAY be provided and MUST be passed into the Enc_structure via the external_aad field.

* enc: The contents of the layer ’ek’ parameter.

* ct: The contents of the layer ciphertext.

The plaintext output is the raw message plaintext.

The COSE_Encrypt0 MAY be tagged or untagged.

An example is shown in Section 5.1.

3.3. HPKE Key Encryption Mode

This mode is selected if the COSE_recipient structure uses a COSE-HPKE algorithm.

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, and 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 COSE_recipient structure using a COSE-HPKE algorithm. The unprotected header MAY contain the kid parameter to identify the static recipient public key that 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.

3.3.1. Recipient_structure

This section defines the Recipient_structure, which is used in place of COSE_KDF_Context for COSE-HPKE recipients. It MUST be used for COSE-HPKE recipients, as it provides integrity protection for recipient-protected header parameters.

The Recipient_structure is modeled after the Enc_structure defined in [RFC9052], but is specific to COSE_recipient structures and MUST NOT be used with COSE_Encrypt.

Furthermore, the use of COSE_KDF_Context is prohibited in COSE-HPKE; it MUST NOT be used.

```
Recipient_structure = [
    context: "HPKE Recipient",
    next_layer_alg: int/tstr,
    recipient_protected_header: empty_or_serialize_map,
    recipient_extra_info: bstr
]
```

* "next_layer_alg" is the algorithm ID of the COSE layer for which the COSE_recipient is encrypting a key. It is the algorithm that the key MUST be used with. This value MUST match the alg parameter in the next lower COSE layer. (This serves the same purpose as the alg ID in the COSE_KDF_Context. It also mitigates attacks where the attacker manipulates the content-encryption algorithm identifier. This attack has been demonstrated against CMS and the mitigation can be found in [I-D.ietf-lamps-cms-cek-hkdf-sha256].

* "recipient_protected_header" contains the protected header parameters from the COSE_recipient CBOR-encoded deterministically with the "Core Deterministic Encoding Requirements", specified in Section 4.2.1 of [RFC8949].

* "recipient_extra_info" contains any additional context the application wishes to include in the key derivation via the HPKE info parameter. If none, it is a zero-length string.

3.3.2. COSE-HPKE Recipient Construction

Because COSE-HPKE supports header protection, if the ’alg’ parameter is present, it MUST be in the protected header parameters and MUST be a COSE-HPKE algorithm.

The protected header MAY contain the kid parameter to identify the static recipient public key that the sender used. Use of the ’kid’ parameter is RECOMMENDED to explicitly identify the static recipient public key used by the sender. Including it in the protected header parameters ensures that it is input into the key derivation function of HPKE.

When encrypting, the inputs to the HPKE Seal operation are set as follows:

* kem_id: Depends on the COSE-HPKE algorithm used.

* pkR: The recipient public key, converted into HPKE public key.

* kdf_id: Depends on the COSE-HPKE algorithm used.

* aead_id: Depends on the COSE-HPKE algorithm used.

* info: Deterministic encoding of the Recipient_structure. Externally provided context information MAY be provided and MUST be passed into the Recipient_structure via the recipient_extra_info field.

* aad: Defaults to the empty string; externally provided information MAY be used instead.

* pt: The raw key for the next layer down.

The outputs are used as follows:

* enc: MUST be placed raw into the 'ek' (encapsulated key) parameter in the unprotected bucket.

* ct: MUST be placed raw in the ciphertext field in the COSE_recipient.

When decrypting, the inputs to the HPKE Open operation are set as follows:

* kem_id: Depends on the COSE-HPKE algorithm used.

* skR: The recipient private key, converted into HPKE private key.

* kdf_id: Depends on the COSE-HPKE algorithm used.

* aead_id: Depends on the COSE-HPKE algorithm used.

* info: Deterministic encoding of the Recipient_structure. Externally provided context information MAY be provided and MUST be passed into the Recipient_structure via the recipient_extra_info field.

* aad: Defaults to the empty string; externally provided information MAY be used instead.

* ct: The contents of the layer ciphertext field.

The plaintext output is the raw key for the next layer down.

It is not necessary to populate recipient_aad, as HPKE inherently mitigates the classes of attacks that COSE_KDF_Context, and SP800-56A are designed to address. COSE-HPKE use cases may still utilize recipient_aad for other purposes as needed; however, it is generally intended for small values such as identifiers, contextual information, or secrets. It is not designed for protecting large or bulk external data.

Any bulk external data that requires protection should be handled at layer 0 using external_aad.

The COSE_recipient structure is computed for each recipient.

When encrypting the content at layer 0, the instructions in Section 5.3 of [RFC9052] MUST be followed, including the calculation of the authenticated data structure.

An example is shown in Section 5.2.

3.4. Key Representation

The COSE_Key with the existing key types can be used to represent KEM private or public keys. When using a COSE_Key for COSE-HPKE, the following checks are made:

* If the "kty" field is "AKP", then the public and private keys SHALL be the raw HPKE public and private keys (respectively) for the KEM used by the algorithm.

* Otherwise, the key MUST be suitable for the KEM used by the algorithm. In case the "kty" parameter is "EC2" or "OKP", this means the value of "crv" parameter is suitable. The valid combinations of KEM, "kty" and "crv" for the algorithms defined in this document are shown in Figure 1.

* If the "key_ops" field is present, it MUST include only "derive bits" for the private key and MUST be empty for the public key.

Examples of the COSE_Key for COSE-HPKE are shown in Section 5.3.

4. Ciphersuite Registration

A ciphersuite is a group of algorithms, often sharing component algorithms such as hash functions, targeting a security level. A COSE-HPKE algorithm is composed of the following choices:

* COSE HPKE Mode

* KEM Algorithm

* KDF Algorithm

* AEAD Algorithm

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

The HPKE mode is determined by the presence or absence of the 'psk_id' parameter and is therefore not explicitly indicated in the ciphersuite.

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 Ciphersuite Label	COSE HPKE Mode	HPKE KEM	HPKE KDF	HPKE AEAD
HPKE-0	Integrated Encryption	0x10	0x1	0x1
HPKE-1	Integrated Encryption	0x11	0x2	0x2
HPKE-2	Integrated Encryption	0x12	0x3	0x2
HPKE-3	Integrated Encryption	0x20	0x1	0x1
HPKE-4	Integrated Encryption	0x20	0x1	0x3
HPKE-5	Integrated Encryption	0x21	0x3	0x2
HPKE-6	Integrated Encryption	0x21	0x3	0x3
HPKE-7	Integrated Encryption	0x10	0x1	0x2
HPKE-0-KE	Key Encryption	0x10	0x1	0x1
HPKE-1-KE	Key Encryption	0x11	0x2	0x2
HPKE-2-KE	Key Encryption	0x12	0x3	0x2
HPKE-3-KE	Key Encryption	0x20	0x1	0x1
HPKE-4-KE	Key Encryption	0x20	0x1	0x3
HPKE-5-KE	Key Encryption	0x21	0x3	0x2
HPKE-6-KE	Key Encryption	0x21	0x3	0x3
HPKE-7-KE	Key Encryption	0x10	0x1	0x2

The following list maps the ciphersuite labels to their textual description.

* HPKE-0: Integrated Encryption with DHKEM(P-256, HKDF-SHA256) KEM, HKDF-SHA256 KDF, and AES-128-GCM AEAD.

* HPKE-1: Integrated Encryption with DHKEM(P-384, HKDF-SHA384) KEM, HKDF-SHA384 KDF, and AES-256-GCM AEAD.

* HPKE-2: Integrated Encryption with DHKEM(P-521, HKDF-SHA512) KEM, HKDF-SHA512 KDF, and AES-256-GCM AEAD.

* HPKE-3: Integrated Encryption with DHKEM(X25519, HKDF-SHA256) KEM, HKDF-SHA256 KDF, and AES-128-GCM AEAD.

* HPKE-4: Integrated Encryption with DHKEM(X25519, HKDF-SHA256) KEM, HKDF-SHA256 KDF, and ChaCha20Poly1305 AEAD.

* HPKE-5: Integrated Encryption with DHKEM(X448, HKDF-SHA512) KEM, HKDF-SHA512 KDF, and AES-256-GCM AEAD.

* HPKE-6: Integrated Encryption with DHKEM(X448, HKDF-SHA512) KEM, HKDF-SHA512 KDF, and ChaCha20Poly1305 AEAD.

* HPKE-7: Integrated Encryption with DHKEM(P-256, HKDF-SHA256) KEM, HKDF-SHA256 KDF, and AES-256-GCM AEAD.

* HPKE-0: Key Encryption with DHKEM(P-256, HKDF-SHA256) KEM, HKDF-SHA256 KDF, and AES-128-GCM AEAD.

* HPKE-1: Key Encryption with DHKEM(P-384, HKDF-SHA384) KEM, HKDF-SHA384 KDF, and AES-256-GCM AEAD.

* HPKE-2: Key Encryption with DHKEM(P-521, HKDF-SHA512) KEM, HKDF-SHA512 KDF, and AES-256-GCM AEAD.

* HPKE-3: Key Encryption with DHKEM(X25519, HKDF-SHA256) KEM, HKDF-SHA256 KDF, and AES-128-GCM AEAD.

* HPKE-4: Key Encryption with DHKEM(X25519, HKDF-SHA256) KEM, HKDF-SHA256 KDF, and ChaCha20Poly1305 AEAD.

* HPKE-5: Key Encryption with DHKEM(X448, HKDF-SHA512) KEM, HKDF-SHA512 KDF, and AES-256-GCM AEAD.

* HPKE-6: Key Encryption with DHKEM(X448, HKDF-SHA512) KEM, HKDF-SHA512 KDF, and ChaCha20Poly1305 AEAD.

* HPKE-7: Key Encryption with DHKEM(P-256, HKDF-SHA256) KEM, HKDF-SHA256 KDF, and AES-256-GCM AEAD.

As the list indicates, the ciphersuite labels have been abbreviated at least to some extent to strike a balance between readability and length.

The ciphersuite list above is a minimal starting point. Additional ciphersuites can be registered into the already existing registry. For example, once post-quantum cryptographic algorithms have been standardized it might be beneficial to register ciphersuites for use with COSE-HPKE. Additionally, ciphersuites utilizing the compact encoding of the public keys, as defined in [I-D.irtf-cfrg-dnhpke], may be standardized for use in constrained environments.

As a guideline for ciphersuite submissions to the IANA COSE algorithm registry, the designated experts must only register combinations of (KEM, KDF, AEAD) triple that constitute valid combinations for use with HPKE, the KDF used should (if possible) match one internally used by the KEM, and components should not be mixed between global and national standards.

4.1. COSE_Keys for COSE-HPKE Ciphersuites

The COSE-HPKE algorithm uniquely determines the KEM for which a COSE_Key is used. The following mapping table shows the valid combinations of the KEM used, COSE_Key type, and its curve/key subtype. This holds for COSE algorithms using either of the COSE HPKE modes (Integrated Encryption and Key Encryption).

HPKE KEM id	COSE_Key	
	kty	crv
0x0010, 0x0013	EC2	P-256
0x0011, 0x0014	EC2	P-384
0x0012, 0x0015	EC2	P-521
0x0020	OKP	X25519
0x0021	OKP	X448

Figure 1: COSE_Key Types and Curves for COSE-HPKE Ciphersuites

5. Examples

This section provides a set of examples that show the HPKE Integrated Encryption Mode and the HPKE Key Encryption Mode, and illustrates the use of key representations for HPKE KEM.

5.1. COSE HPKE Integrated Encryption Mode

This example assumes that a sender wants to communicate an encrypted payload to a single recipient, named "bob".

An example of the HPKE Integrated Encryption Mode is shown in Figure 3. Line breaks and comments have been inserted for better readability.

This example uses the following:

* Suite: HPKE-0 (P-256 / HKDF-SHA256 / AES-128-GCM)

* Plaintext: "This is the content."

* External AAD: empty

* Info: empty

* Recipient kid: "bob"

The ciphertext (hex) transmitted to "bob" is:

```
d08344a1011823a1235841042fee971fa778fac9c095f835bdf4033d2ae8
d1b8e8dde4b1f6739a05df8bb338a9bccd52aea211b12d13496d1d5aad5f
26bc0a1789160d940130003176cf861e5825d4a351c896b4dd9c66fd9ab3
00bd5788ba8d0c9c895202bd0a42be864a5854c36b00280748
```

Figure 2: Hex-Encoding of COSE_Encrypt0

COSE_Encrypt0 pretty-printed:

```
{
  "protected": {
    1 /alg/: 35 /HPKE-0 (P-256 + HKDF-SHA256 + AES-128-GCM)/
  },
  "unprotected": {
    -4 /ek/: h'042fee971fa778fac9c095f835bdf4033d2ae8d1b8e8dde4b1f6739a0
5df8bb338a9bccd52aea211b12d13496d1d5aad5f26bc0a1789160d940130003176cf861
e'
  },
  "ciphertext": h'd4a351c896b4dd9c66fd9ab300bd5788ba8d0c9c895202bd0a42be
864a5854c36b00280748'
}
```

Figure 3: COSE_Encrypt0 Example for HPKE

The following COSE Key was used in this example:

```
{
   1 /kty/: 2,
   2 /kid/: h'626f62',
   3 /alg/: 35 /HPKE-0  (P-256 + HKDF-SHA256 + AES-128-GCM)/,
   -1 /crv/: 1 /P-256/,
   -2 /x/:
   h'02a8e3315f96bc7355dbf85740c6d8e53fb070cd8ba5c419be49a91d789ef55c',
   -3 /y/:
   h'96b6621abf5ca532e042dc5c346c1ef0c9186b83cb122e50a46f1458de023d35'
}
```

Figure 4: COSE Key

5.2. COSE HPKE Key Encryption Mode

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

This example uses the following input parameters:

* Suite: HPKE-0 (P-256 / HKDF-SHA256 / AES-128-GCM)

* Plaintext: "This is the content."

* External AAD: empty

* Info: empty

* Recipient kid: "alice"

Alice uses the following COSE Keys, in hex-encoding:

```
a701020245616c69636503182e20012158201f2124bdf7da2656a1e404fd
fb9958d762de1db78959d1ef7f9464371610718822582 02abfce2ff6083d
28be1cd915f5682932ce8f45c08e9db1d3dfc8bd5e89d4dcb423582031e9
ff686ed68feafa8774d2fb88370da73b4de5a8a8104545c2c3db412b8202
```

This hex-sequence decodes to the following COSE Key:

```
{
1 /kty/: 2,
2 /kid/: h'616c696365',
3 /alg/: 46 /HPKE-0-KE (DHKEM(P-256, HKDF-SHA256) / HKDF-SHA256 /
AES-128-GCM)/,
-1 /crv/: 1 /P-256/,
-2 /x/:
h'1f2124bdf7da2656a1e404fdfb9958d762de1db78959d1ef7f94643716107188',
-3 /y/:
h'2abfce2ff6083d28be1cd915f5682932ce8f45c08e9db1d3dfc8bd5e89d4dcb4',
-4 /d/:
h'31e9ff686ed68feafa8774d2fb88370da73b4de5a8a8104545c2c3db412b8202'
}
```

As a result, the following COSE_Encrypt payload is created:

```
d8608443a10101a10550311aa7cc6cf87c1068a2ca671ede2cc758253e17
fb3a5e244dc66e4d175424d364c24c6f85945b75f3788cf35116c67bc7b8
45df5757c381834ba201182e0445616c696365a123584104a97992eb7834
74b632289699b5b97218595f82c0fe2ed9097d505f6a64c8c530775a66fa
36b8728ae329a24c8c6580a59c80e7c5d583b197cd42222cdb4f674c5820
649f31f9e647c3d7ded69f8cd6ed1ca4e5171f681c39eed1f077a1d1a01b
f784
```

Decoded, this hex-sequence has the following content:

```
{
    "protected": {
        1 /alg/: 1 /A128GCM/
    },
    "unprotected": {
        5 /iv/: h'311aa7cc6cf87c1068a2ca671ede2cc7'
    },
    "ciphertext": h'3e17fb3a5e244dc66e4d175424d364c24c6f85945b75f3788cf3
5116c67bc7b845df5757c3',
    "recipients": [
    {
        "protected": {
            1 /alg/: 46 /HPKE-0-KE (DHKEM(P-256, HKDF-SHA256) / HKDF-SHA
256 / AES-128-GCM)/,
            4 /kid/: h'616c696365'
        },
        "unprotected": {
            -4 /ek/: h'04a97992eb783474b632289699b5b97218595f82c0fe2ed90
97d505f6a64c8c530775a66fa36b8728ae329a24c8c6580a59c80e7c5d583b197cd42222
cdb4f674c'
        },
        "encrypted_cek": h'649f31f9e647c3d7ded69f8cd6ed1ca4e5171f681c39e
ed1f077a1d1a01bf784'
    }
    ]
}
```

Figure 5: COSE_Encrypt Example for HPKE

5.3. Key Representation

Examples of private and public KEM key representation are shown below.

5.3.1. Public Key for HPKE-0

```
{
    / kty = 'EC2' /
    1: 2,
    / kid = '01' /
    2: h'3031',
    / alg = HPKE-0 (Assumed: 35) /
    3: 35,
    / crv = 'P-256' /
    -1: 1,
    / x /
    -2: h'65eda5a12577c2bae829437fe338701a10aaa375
          e1bb5b5de108de439c08551d',
    / y /
    -3: h'1e52ed75701163f7f9e40ddf9f341b3dc9ba860af
          7e0ca7ca7e9eecd0084d19c'
}
```

Figure 6: Public Key Representation Example for HPKE-0

5.3.2. Private Key for HPKE-0

```
{
    / kty = 'EC2' /
    1: 2,
    / kid = '01' /
    2: h'3031',
    / alg = HPKE-0 (Assumed: 35) /
    3: 35,
    / key_ops = ['derive_bits'] /
    4: [8],
    / crv = 'P-256' /
    -1: 1,
    / x /
    -2: h'bac5b11cad8f99f9c72b05cf4b9e26d244dc189f7
          45228255a219a86d6a09eff',
    / y /
    -3: h'20138bf82dc1b6d562be0fa54ab7804a3a64b6d72
          ccfed6b6fb6ed28bbfc117e',
    / d /
    -4: h'57c92077664146e876760c9520d054aa93c3afb04
          e306705db6090308507b4d3',
}
```

Figure 7: Private Key Representation Example for HPKE-0

5.3.3. KEM Public Key for HPKE-4

```
{
    / kty = 'OKP' /
    1: 1,
    / kid = '11' /
    2: h'3131',
    / alg = HPKE-4 (Assumed: 42) /
    3: 42,
    / crv = 'X25519' /
    -1: 4,
    / x /
    -2: h'cb7c09ab7b973c77a808ee05b9bbd373b55c06eaa
          9bd4ad2bd4e9931b1c34c22',
}
```

Figure 8: Public Key Representation Example for HPKE-4

6. Security Considerations

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

Both HPKE and HPKE COSE assume that the sender possesses the recipient's public key. Therefore, some form of public key distribution mechanism is assumed to exist, but this is 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 generation 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.

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

7.1.1. HPKE-0

* Name: HPKE-0
* Value: TBD1 (Assumed: 35)
* Description: COSE HPKE Integrated Encryption using DHKEM(P-256, HKDF-SHA256) KEM, HKDF-SHA256 KDF, and AES-128-GCM AEAD.
* Capabilities: [kty]
* Change Controller: IESG
* Reference: [[TBD: This RFC]]
* Recommended: Yes

7.1.2. HPKE-1

* Name: HPKE-1
* Value: TBD3 (Assumed: 37)
* Description: COSE HPKE Integrated Encryption using DHKEM(P-384, HKDF-SHA384) KEM, HKDF-SHA384 KDF, and AES-256-GCM AEAD.
* Capabilities: [kty]
* Change Controller: IESG
* Reference: [[TBD: This RFC]]
* Recommended: Yes

7.1.3. HPKE-2

* Name: HPKE-2
* Value: TBD5 (Assumed: 39)
* Description: COSE HPKE Integrated Encryption using DHKEM(P-521, HKDF-SHA512) KEM, HKDF-SHA512 KDF, and AES-256-GCM AEAD.
* Capabilities: [kty]
* Change Controller: IESG

* Reference: [[TBD: This RFC]]

* Recommended: Yes

7.1.4. HPKE-3

* Name: HPKE-3

* Value: TBD7 (Assumed: 41)

* Description: COSE HPKE Integrated Encryption using DHKEM(X25519, HKDF-SHA256) KEM, HKDF-SHA256 KDF, and AES-128-GCM AEAD.

* Capabilities: [kty]

* Change Controller: IESG

* Reference: [[TBD: This RFC]]

* Recommended: Yes

7.1.5. HPKE-4

* Name: HPKE-4

* Value: TBD8 (Assumed: 42)

* Description: COSE HPKE Integrated Encryption using DHKEM(X25519, HKDF-SHA256) KEM, HKDF-SHA256 KDF, and ChaCha20Poly1305 AEAD.

* Capabilities: [kty]

* Change Controller: IESG

* Reference: [[TBD: This RFC]]

* Recommended: Yes

7.1.6. HPKE-5

* Name: HPKE-5

* Value: TBD9 (Assumed: 43)

* Description: COSE HPKE Integrated Encryption using DHKEM(X448, HKDF-SHA512) KEM, HKDF-SHA512 KDF, and AES-256-GCM AEAD.

* Capabilities: [kty]

* Change Controller: IESG

* Reference: [[TBD: This RFC]]

* Recommended: Yes

7.1.7. HPKE-6

* Name: HPKE-6

* Value: TBD10 (Assumed: 44)

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

* Capabilities: [kty]

* Change Controller: IESG

* Reference: [[TBD: This RFC]]

* Recommended: Yes

7.1.8. HPKE-7

* Name: HPKE-7

* Value: TBD13 (Assumed: 45)

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

* Capabilities: [kty]

* Change Controller: IESG

* Reference: [[TBD: This RFC]]

* Recommended: Yes

7.1.9. HPKE-0-KE

* Name: HPKE-0-KE

* Value: TBD14 (Assumed: 46)

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

* Capabilities: [kty]

* Change Controller: IESG

* Reference: [[TBD: This RFC]]

* Recommended: Yes

7.1.10. HPKE-1-KE

* Name: HPKE-1-KE

* Value: TBD15 (Assumed: 47)

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

* Capabilities: [kty]

* Change Controller: IESG

* Reference: [[TBD: This RFC]]

* Recommended: Yes

7.1.11. HPKE-2-KE

* Name: HPKE-2-KE

* Value: TBD16 (Assumed: 48)

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

* Capabilities: [kty]

* Change Controller: IESG

* Reference: [[TBD: This RFC]]

* Recommended: Yes

7.1.12. HPKE-3-KE

* Name: HPKE-3-KE

* Value: TBD17 (Assumed: 49)

* Description: COSE HPKE Key Encryption using DHKEM(X25519, HKDF-SHA256) KEM, HKDF-SHA256 KDF, and AES-128-GCM AEAD.

* Capabilities: [kty]

* Change Controller: IESG

* Reference: [[TBD: This RFC]]

* Recommended: Yes

7.1.13. HPKE-4-KE

* Name: HPKE-4-KE

* Value: TBD18 (Assumed: 50)

* Description: COSE HPKE Key Encryption using DHKEM(X25519, HKDF-SHA256) KEM, HKDF-SHA256 KDF, and ChaCha20Poly1305 AEAD.

* Capabilities: [kty]

* Change Controller: IESG

* Reference: [[TBD: This RFC]]

* Recommended: Yes

7.1.14. HPKE-5-KE

* Name: HPKE-5-KE

* Value: TBD19 (Assumed: 51)

* Description: COSE HPKE Key Encryption using DHKEM(X448, HKDF-SHA512) KEM, HKDF-SHA512 KDF, and AES-256-GCM AEAD.

* Capabilities: [kty]

* Change Controller: IESG

* Reference: [[TBD: This RFC]]

* Recommended: Yes

7.1.15. HPKE-6-KE

* Name: HPKE-6-KE
* Value: TBD20 (Assumed: 52)
* Description: COSE HPKE Key Encryption using DHKEM(X448, HKDF-SHA512) KEM, HKDF-SHA512 KDF, and ChaCha20Poly1305 AEAD.
* Capabilities: [kty]
* Change Controller: IESG
* Reference: [[TBD: This RFC]]
* Recommended: Yes

7.1.16. HPKE-7-KE

* Name: HPKE-7-KE
* Value: TBD21 (Assumed: 53)
* Description: COSE HPKE Key Encryption using DHKEM(P-256, HKDF-SHA256) KEM, HKDF-SHA256 KDF, and AES-256-GCM AEAD.
* Capabilities: [kty]
* Change Controller: IESG
* Reference: [[TBD: This RFC]]
* Recommended: Yes

7.2. COSE Header Parameters

7.2.1. ek Header Parameter

* Name: ek
* Label: TBD11 (Assumed: -4)
* Value type: bstr
* Value Registry: N/A
* Description: HPKE encapsulated key

* Reference: [[TBD: This RFC]]

7.2.2. psk_id Header Parameter

* Name: psk_id

* Label: TBD12 (Assumed: -5)

* Value type: bstr

* Value Registry: N/A

* Description: A key identifier (kid) for the pre-shared key as defined in Section 5.1.2 of [RFC9180]

* Reference: [[TBD: This RFC]]

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>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/rfc/rfc8949>.

[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.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-sha256-05, 19 September 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-cms-cek-hkdf-sha256-05>.

[I-D.irtf-cfrg-dnhpke]
Harkins, D., "Deterministic Nonce-less Hybrid Public Key Encryption", Work in Progress, Internet-Draft, draft-irtf-cfrg-dnhpke-07, 16 October 2025, <https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-dnhpke-07>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/rfc/rfc5652>.

[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>.

[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. Contributors

We would like to 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, Thomas Fossati, and G旦ran Selander for their contributions to the specification.

Authors' Addresses

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

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

Daisuke Ajitomi
bibital
Japan
Email: dajiaji@gmail.com

Laurence Lundblade
Security Theory LLC
United States
Email: lgl@securitytheory.com

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