COSE
Internet-Draft
Intended status: Standards Track
Expires: 22 April 2026

H. Tschofenig
H-BRS
O. Steele, Ed.
Transmute
D. Ajitomi
bibital
L. Lundblade
Security Theory LLC
19 October 2025

Use of Hybrid Public-Key Encryption (HPKE) with CBOR Object Signing and Encryption (COSE)
draft-ietf-cose-hpke-18

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.

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 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 3.1.1 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 3.1.2 provides the details.

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

When used with HPKE, the 'ek' header parameter MUST be present in the unprotected header and MUST contain the encapsulated key, which is the output of the HPKE KEM. The value of 'ek' MUST be a bstr.

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.

3.1.1. 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.

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

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

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

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

* 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.

3.1.2. 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.

3.1.2.1. Recipient Encryption

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.

3.1.2.2. 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.

* 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.

* 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.

3.2. 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:

* 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	HPKE		
	KEM	KDF	AEAD
HPKE-0	0x10	0x1	0x1
HPKE-1	0x11	0x2	0x2
HPKE-2	0x12	0x3	0x2
HPKE-3	0x20	0x1	0x1
HPKE-4	0x20	0x1	0x3
HPKE-5	0x21	0x3	0x2
HPKE-6	0x21	0x3	0x3

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

* HPKE-0: DHKEM(P-256, HKDF-SHA256) KEM, HKDF-SHA256 KDF and AES-128-GCM AEAD.
* HPKE-1: DHKEM(P-384, HKDF-SHA384) KEM, HKDF-SHA384 KDF, and AES-256-GCM AEAD.

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

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

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

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

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

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 all COSE message types (COSE_Encrypt0 and COSE_Encrypt) to which the COSE-HPKE can be applied, and also provides some examples of key representation for HPKE KEM.

Each example of the COSE message 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. HPKE Integrated Encryption Mode

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 HPKE Integrated Encryption Mode is shown in Figure 2. Line breaks and comments have been inserted for better readability.

This example uses the following:

* alg: HPKE-0

* plaintext: "This is the content."

* external_aad: "COSE-HPKE app"

* skR: h'57c92077664146e876760c9520d054aa93c3afb04e306705db6090308507b4d3'

* skE: h'42dd125eefc409c3b57366e721a40043fb5a58e346d51c133128a77237160218'

```
16([
    / alg = HPKE-0 (Assumed: 35) /
    h'a1011823',
    {
        / kid /
        4: h'3031',
        / ek /
        -4: h'045df24272faf43849530db6be01f42708b3c3a9
              df8e268513f0a996ed09ba7840894a3fb946cb28
              23f609c59463093d8815a7400233b75ca8ecb177
              54d241973e',
    },
    / encrypted plaintext /
    h'35aa3d98739289b83751125abe44e3b977e4b9abbf2c8cfaade
      b15f7681eef76df88f096',
])
```

Figure 2: COSE_Encrypt0 Example for HPKE

5.2. HPKE Key Encryption Mode

In this example we assume that a sender wants to transmit a payload to two recipients using the HPKE Key Encryption mode. 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 key encryption using the COSE_Encrypt structure using HPKE is shown in Figure 3. Line breaks and comments have been inserted for better readability.

This example uses the following input parameters:

* Content encryption algorithm: AES-128-GCM

* plaintext: "This is the payload."

* kid:"alice"

* alg: HPKE-0 - DHKEM(P-256, HKDF-SHA256), KDF: HKDF-SHA256, AEAD: AES-128-GCM

* external_aad: "some externally provided aad"

Alice uses the following NIST P-256 ECC keys.

Private Key:

```
0xaf, 0xf9, 0x07, 0xc9, 0x9f, 0x9a, 0xd3, 0xaa,
0xe6, 0xc4, 0xcd, 0xf2, 0x11, 0x22, 0xbc, 0xe2,
0xbd, 0x68, 0xb5, 0x28, 0x3e, 0x69, 0x07, 0x15,
0x4a, 0xd9, 0x11, 0x84, 0x0f, 0xa2, 0x08, 0xcf
```

Public Key:

```
/* SEC Serialization of X and Y */
0x04,

/* X & Y */
0x65, 0xed, 0xa5, 0xa1, 0x25, 0x77, 0xc2, 0xba,
0xe8, 0x29, 0x43, 0x7f, 0xe3, 0x38, 0x70, 0x1a,
0x10, 0xaa, 0xa3, 0x75, 0xe1, 0xbb, 0x5b, 0x5d,
0xe1, 0x08, 0xde, 0x43, 0x9c, 0x08, 0x55, 0x1d,

0x1e, 0x52, 0xed, 0x75, 0x70, 0x11, 0x63, 0xf7,
0xf9, 0xe4, 0x0d, 0xdf, 0x9f, 0x34, 0x1b, 0x3d,
0xc9, 0xba, 0x86, 0x0a, 0xf7, 0xe0, 0xca, 0x7c,
0xa7, 0xe9, 0xee, 0xcd, 0x00, 0x84, 0xd1, 0x9c
```

As a result, the following COSE_Encrypt payload is created:

```
d8 60 84 43 a1 01 01 a1 05 50 7f 55 a2 6b 98 c0
49 b4 28 a7 cf 25 9d c3 0e 54 58 23 3f ae 53 ee
83 55 ee 40 4e 86 7c 00 74 f8 c3 8c 6d 13 6b 65
bb 61 93 92 79 b4 38 48 c5 8c b6 a4 76 03 55 81
83 4b a2 01 18 23 04 45 61 6c 69 63 65 a1 23 58
41 04 fe 73 6d 1d 93 11 4d f6 11 3b c2 87 cd 8e
63 67 e1 0a b4 78 d7 fe df ac a1 6e 12 6f f0 16
d6 95 d5 f7 22 34 03 e3 99 60 75 55 bc cf b9 65
17 5f 49 14 e0 47 73 f7 04 07 5b 46 58 bf 7a dd
84 a3 58 20 55 12 c2 35 7d 4c b6 bd 23 8a 5f bc
10 84 b6 c9 74 0a c2 41 1d 93 63 7a 51 e6 9d 51
0b 4f ae f8
```

Decoded, this hex-sequence has the following content:

```
================ NOTE: '\' line wrapping per RFC 8792 ================

  96([
       / alg = AES-128-GCM (1) /
       h'A10101',
       {
           / iv /
           5: h'33739C468ACB8EEC693C563EAEA12DD0'
       },
       / ciphertext /
       h'\
1F3EE9966D5CEE016E49365CF366FD608F271FC3B5ABDD5253844EE38EE6ABB7F555\
                                                                  9A',
       [
           [
               / alg = HPKE-0 (35), kid = 'alice' /
               h'A20118230445616C696365',
               {
                   / ek /
                   -4: h'\
040506BE8D9C2AFE42D3330676A3F616BAE02F6779D962449F26759B8D1E8F4DF10C\
     9F344627DEB063EE1DDB4858A5E7605BD09ECEB409B037E6E61F44D1E946C1'
               },
               / ciphertext containing encrypted CEK /
               h'\
   B11361397A19E9C155C3E0E8117B5E88155600E550DDE03DC834A46A182DE6F1'
           ]
       ]
   ])
```

Figure 3: COSE_Encrypt Example for HPKE

To offer authentication of the sender the payload in Figure 3 is signed with a COSE_Sign1 wrapper, which is outlined in Figure 4. The payload in Figure 4 is meant to contain the content of Figure 3.

Bob uses the following signature key to sign the COSE_Encrypt payload without any additional data.

Private Key:

```
0xd9, 0xb5, 0xe7, 0x1f, 0x77, 0x28, 0xbf, 0xe5,
0x63, 0xa9, 0xdc, 0x93, 0x75, 0x62, 0x27, 0x7e,
0x32, 0x7d, 0x98, 0xd9, 0x94, 0x80, 0xf3, 0xdc,
0x92, 0x41, 0xe5, 0x74, 0x2a, 0xc4, 0x58, 0x89
```

The output of the message is as follows:

```
   =============== NOTE: '\' line wrapping per RFC 8792 ================

   18([
          / alg = ES256 (-7) /
          h'A10126',
          {
             / kid = 'bob' /
             4: h'626F62'
          },
          / payload / h'\
   D8608443A10101A1055033739C468ACB8EEC693C563EAEA12DD058231F3EE9966D5C\
   EE016E49365CF366FD608F271FC3B5ABDD5253844EE38EE6ABB7F5559A81834BA201\
   18230445616C696365A1235841040506BE8D9C2AFE42D3330676A3F616BAE02F6779\
   D962449F26759B8D1E8F4DF10C9F344627DEB063EE1DDB4858A5E7605BD09ECEB409\
   B037E6E61F44D1E946C15820B11361397A19E9C155C3E0E8117B5E88155600E550DD\
                                                   E03DC834A46A182DE6F1',
          / Signature /
          h'\
   7F9A83D1753E6FA8475A1250A786DA3E680265949A0AEE1984895A406E41AE8A2966\
           38CA64AE270C5317829BD3968EF76C42DF1566DADC9A68B06BA6ED376B8A'
     ])
```

Figure 4: COSE_Sign1 Example

5.3. Key Representation

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

5.3.1. KEM 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 5: Key Representation Example for HPKE-0

5.3.2. KEM 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 6: 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 7: 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: 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

7.1.2. HPKE-1

* Name: HPKE-1

* 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

7.1.3. HPKE-2

* Name: HPKE-2

* 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

7.1.4. HPKE-3

* Name: HPKE-3

* 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

7.1.5. HPKE-4

* Name: HPKE-4

* 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

7.1.6. HPKE-5

* Name: HPKE-5

* 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

7.1.7. HPKE-6

* Name: HPKE-6
* 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

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>.

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 Michael B. Jones, John Mattsson, Mike Prorock, Michael Richardson, Thomas Fossati, and G旦ran Selander for their review feedback.

Authors' Addresses

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

Orie Steele (editor)
Transmute
United States
Email: orie@transmute.industries

Daisuke Ajitomi
bibital
Japan
Email: dajiaji@gmail.com

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