Workgroup: COSE
Internet-Draft: draft-ietf-cose-hpke-04
Published: 13 March 2023
Intended Status: Standards Track
Expires: 14 September 2023
Authors: H. Tschofenig   B. Moran
Arm Limited

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.

This document defines the use of the HPKE base mode with COSE. Other modes are supported by HPKE but not by this specification.

Status of This Memo

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. 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 defines several features for use with public key encryption and a subset of those features is applied to COSE ([RFC9052], [RFC9053]). Since COSE provides constructs for authentication, those are not re-used from the HPKE specification. This specification uses the "base" mode, as it is called in HPKE specification language.

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 cases uses 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.

HPKE in Base mode requires little information to be provided by the sender, namely

* algorithm information (KEM, KDF, and AEAD identifiers),

* an encapsulated key generated by the sender, and

* an identifier of the static recipient key.

In the subsections below we explain how this information is carried inside the COSE_Encrypt0 and the COSE_Encrypt for the one layer and the two layer structure, respectively.

In both cases a new structure is used to convey information about the HPKE sender, namely the HPKE sender information structure (sender_info).

When the alg value is set to 'HPKE-v1-BASE', the sender_info structure MUST be present in the unprotected header parameter.

The CDDL grammar describing the sender_info structure is:

```
sender_info = [
    kem_id : uint,         ; kem identifier
    kdf_id : uint,         ; kdf identifier
    aead_id : uint,        ; aead identifier
    enc : bstr,            ; encapsulated key
]
```

The fields have the following meaning:

Name	CBOR Type	Value Registry	Description
kem_id	uint	HPKE KEM IDs Registry	Identifier for the KEM
kdf_id	uint	HPKE KDF IDs	Identifier for the KDF ID
aead_id	uint	HPKE AEAD IDs	Identifier for the AEAD ID
enc	bstr		Encapsulated key defined by HPKE

Figure 1: sender_info structure

kem_id: This parameter is used to identify the KEM. The registry for KEM ids has been established with RFC 9180.

kdf_id: This parameter contains the KDF identifier. The registry containing the KDF ids has been established with RFC 9180.

aead_id: This parameter contains the AEAD identifier. The registry containing the AEAD ids has been established with RFC 9180.

enc: This parameter contains the encapsulated key, which is output of the HPKE KEM.

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. The resulting ciphertext may be included in the COSE_Encrypt0 or may be 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 kid parameter and the sender_info structure into the unprotected header. The kid identifies the static recipient public key used by the sender. The recipient uses the kid to determine the appropriate private key.

Figure 2 shows the COSE_Encrypt0 CDDL structure.

```
COSE_Encrypt0_Tagged = #6.16(COSE_Encrypt0)

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

Figure 2: 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 sender_info structure as well as 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. For example, the content encrypted at layer 0 may be a firmware image. The same encrypted firmware image may need to be sent to many recipients; however, each recipient uses their own private key to obtain the CEK.

The COSE_recipient structure, shown in Figure 3, 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 3: CDDL for HPKE-based COSE_Encrypt Structure

The COSE_Encrypt MAY be tagged or untagged.

An example is shown in Section 5.2.

4. HPKE Encryption and Decryption

4.1. HPKE Encryption with SealBase

The SealBase(pkR, info, aad, pt) function is used to encrypt a plaintext pt to a recipient's public key (pkR).

Two cases of plaintext need to be distinguished:

* For use in COSE_Encrypt, the plaintext "pt" passed into SealBase is the CEK. The CEK is a random byte sequence of length

appropriate for the encryption algorithm selected in layer 0. For example, AES-128-GCM requires a 16 byte key and the CEK would therefore be 16 bytes long.

* In case of COSE_Encrypt0, the plaintext "pt" passed into SealBase is the content to be encrypted. Hence, there is no intermediate layer utilizing a CEK.

The "aad" and the "info" parameters are described in Section 4.3 and Section 4.4, respectively.

If SealBase() is successful, it will output a ciphertext "ct" and an encapsulated key "enc".

4.2. HPKE Decryption with OpenBase

The recipient will use the OpenBase(enc, skR, info, aad, ct) function with the "enc" and the "ct" parameters received from the sender. The "aad" and the "info" parameters are assumed to be constructed from the context and described in Section 4.3 and Section 4.4, respectively.

The OpenBase function will, if successful, decrypt "ct". When decrypted, the result will be either the CEK (when COSE_Encrypt is used), or the content (if COSE_Encrypt0 is used). The CEK is the symmetric key used to decrypt the ciphertext at layer 0.

4.3. AAD Parameter

HPKE requires an "aad" parameter to be provided to the SealBase and OpenBase functions. Note that there are three types of additional authenticated data used by this specification:

* AAD provided to HPKE for COSE_Encrypt0.

* AAD provided to HPKE for COSE_Encrypt at the recipient layer.

* AAD provided to the AEAD cipher used for content encryption at layer 0 by COSE_Encrypt.

We describe the three variants in the subsections below.

4.3.1. AAD provided to HPKE for COSE_Encrypt0

When COSE_Encrypt0 is used then there is no separate AEAD function at the content encryption layer provided by COSE natively and HPKE offers this functionality.

The "aad" parameter of provided to the SealBase and OpenBase functions is constructed as follows (again intentionally aligned with COSE by re-using the Enc_structure):

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

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 is populated with the API caller provided AAD information. If this field is not supplied, it defaults to a zero-length byte string.

4.3.2. AAD provided to HPKE for COSE_Encrypt at the Recipient Layer

The AAD used at the recipient layer re-uses Enc_structure from [RFC9052] and populates it with the following content:

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

The protected field in the Enc_structure contains the protected attributes from the COSE_recipient structure at layer 1, encoded in a bstr type.

The external_aad field in the Enc_structure is populated with the API caller provided AAD information. In the COSE_Encrypt case this AAD information is also input to the AAD at layer 0, if an AEAD cipher is used at layer 0. If this field is not supplied, it defaults to a zero-length byte string.

4.3.3. AAD provided to the AEAD cipher used for Content Encryption at Layer 0 by COSE_Encrypt

The construction of AAD is defined in Section 5.3 of [RFC9052] (see Enc_structure structure).

4.4. 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, when used with SealBase() and OpenBase(). 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 4.

```
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 4: COSE_KDF_Context Data Structure as 'info' Parameter for HPKE

5. Examples

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 5. Line breaks and comments have been inserted for better readability.

It uses the following algorithm combination: - KEM: DHKEM(P-256, HKDF-SHA256) - KDF: HKDF-SHA256 - AEAD: AES-128-GCM

```
// payload: "This is the content", aad: ""
//
16([
    h'a10120',  // alg = HPKE-v1-BASE
    {
        4: h'3031', // kid
        -4: [       // sender_info
            16,     // kem = DHKEM(P-256, HKDF-SHA256)
            1,      // kdf = HKDF-SHA256
            1,      // aead = AES-128-GCM
            h'048c6f75e463a773082f3cb0d3a701348a578c67
              80aba658646682a9af7291dfc277ec93c3d58707
              818286c1097825457338dc3dcaff367e2951342e
              9db30dc0e7',  // enc
        ],
    },
    / encrypted plaintext /
    h'ee22206308e478c279b94bb071f3a5fbbac412a6effe34195f7
      c4169d7d8e81666d8be13',
])
```

Figure 5: 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.

An example of the COSE_Encrypt structure using the HPKE scheme is shown in Figure 6. Line breaks and comments have been inserted for better readability.

It uses the following algorithm combination:

* At layer 0 AES-128-GCM is used for encryption of the detached plaintext "This is the content.".

* At the recipient structure at layer 1, DHKEM(P-256, HKDF-SHA256) (as the KEM), with AES-128-GCM (as the AEAD) and HKDF-SHA256 (as the KDF) is used.

The algorithm selection is based on the registry of the values offered by the alg parameters (see Section 7).

```
// plaintext: "This is the content.", aad: ""
96_0([
    h'a10101',  // alg = AES-128-GCM (1)
    {5: h'67303696a1cc2b6a64867096'},  // iv
    h'',        // detached ciphertext
    [
        [
            h'a10120',  // alg = HPKE-v1-BASE (-1 #TBD)
            {
                4: h'3031', // kid
                -4: [       // sender_info
                    16,     // kem = DHKEM(P-256, HKDF-SHA256)
                    1,      // kdf = HKDF-SHA256
                    1,      // aead = AES-128-GCM
                    / enc output /
                    h'0421ccd1b00dd958d77e10399c
                         97530fcbb91a1dc71cb3bf41d9
                         9fd39f22918505c973816ecbca
                         6de507c4073d05cceff73e0d35
                         f60e2373e09a9433be9e95e53c',
                ],
            },
            // ciphertext containing encrypted CEK
            h'bb2f1433546c55fb38d6f23f5cd95e1d72eb4
              c129b99a165cd5a28bd75859c10939b7e4d',
        ],

        [
            h'a10120',  // alg = HPKE-v1-BASE (-1 #TBD)
            {
                4: h'313233', // kid
                -4: [       // sender_info
                    16,     // kem = DHKEM(P-256, HKDF-SHA256)
                    1,      // kdf = HKDF-SHA256
                    1,      // aead = AES-128-GCM
                    / enc output /
                       h'6de507c4073d05cceff73e0d35
                         f60e2373e09a9433be9e95e53c
                         9fd39f22918505c973816ecbca
                         6de507c4073d05cceff73e0d35
                         f60e2373e09a9433be9e95e53c',
                ],
            },
            // ciphertext containing encrypted CEK
            h'c4169d7d8e81666d8be13bb2f1433546c55fb
              c129b99a165cd5a28bd75859c10939b7e4d',
        ]
    ],
])
```

Figure 6: COSE_Encrypt Example for HPKE

To offer authentication of the sender the payload in Figure 6 is signed with a COSE_Sign1 wrapper, which is shown in Figure 7. The payload in Figure 7 corresponds to the content shown in Figure 6.

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

Figure 7: COSE_Encrypt Example for HPKE

6. Security Considerations

This specification is based on HPKE and the security considerations of HPKE [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.

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 the it MUST be ensured that the guidelines for random number generations are followed.

The COSE_Encrypt structure MUST be authenticated using COSE constructs like COSE_Sign, COSE_Sign1, COSE_MAC, or COSE_MAC0.

When 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 Algorithm Parameters' registries in the 'Standards Action With Expert Review category.

7.1. COSE Algorithms Registry

*Name: HPKE-v1-BASE

*Value: TBD1 (Assumed: -1)

*Description: HPKE in version 1 in base mode for use with COSE

*Capabilities: [kty]

*Change Controller: IESG

*Reference: [[TBD: This RFC]]

*Recommended: Yes

7.2. COSE Header Algorithm Parameters

*Name: sender_info

*Label: TBD2 (Assumed: -4)

*Value type: sender_info

*Value Registry: N/A

*Description: HPKE Sender Information structure for the Base mode.

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

[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.

- Daisuke Ajitomi

- Richard Barnes

- Ilari Liusvaara

Finally, we would like to thank Russ Housley for his contributions to the draft as a co-author of initial versions.

Appendix B. Acknowledgements

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

Authors' Addresses

Hannes Tschofenig

Email: hannes.tschofenig@gmx.net

Brendan Moran
Arm Limited

Email: Brendan.Moran@arm.com