Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 18 September 2026

C. F. Myers, Ed.
R. S., Ed.
Freedom of the Press Foundation

17 March 2026

Authenticated Modes for HPKE
draft-ms-hpke-auth-modes-00

Abstract

The standards-track [I-D.ietf-hpke-hpke] supersedes the informational [RFC9180], omitting its authenticated modes mode_auth and mode_auth_psk. This document restores those two modes as a strict extension, requiring only the addition of AuthEncap()/AuthDecap() to the DHKEM, the definition of four setup functions, and a change in VerifyPSKInputs(). The extension does not alter the externally observable behavior of the existing HPKE modes standardized in [I-D.ietf-hpke-hpke].

This document also illustrates, informatively, how the restored modes can be used. One such application uses mode_auth_psk with a post-quantum KEM (PQ-KEM) shared secret as the PSK, providing hybrid PQ/T confidentiality. This material is provided to motivate the extension and may be developed as separate work.

About This Document

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

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

Source for this draft and an issue tracker can be found at https://github.com/cfm/draft-ms-hpke-auth-modes.

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/.

Copyright Notice

Table of Contents

1. Introduction

[I-D.ietf-hpke-hpke] is the standards-track successor to the informational [RFC9180] and omits the authenticated modes mode_auth and mode_auth_psk to simplify the standard. However, some protocols require an authenticated key-encapsulation mechanism (AKEM)---a KEM whose shared secret is implicitly bound to the sender's static private key---without requiring a full authenticated encryption context. The DHKEM construction in [I-D.ietf-hpke-hpke] already contains all the primitives needed: AuthEncap() computes a static-static DH value DH(skS, pkR) alongside the ephemeral-static value and mixes both into the key derivation, binding the output to the sender's key pair.

The normative portion of this document is small. It restores mode_auth and mode_auth_psk as a strict extension to [I-D.ietf-hpke-hpke], requiring only AuthEncap()/AuthDecap() on the DHKEM, four setup functions, and a modified VerifyPSKInputs(). The externally observable behavior of existing HPKE modes is unchanged.

Section 4 and Section 5 describe, informatively, how the restored modes can be used, including a construction that layers a post-quantum KEM shared secret as the PSK to achieve hybrid PQ/T confidentiality as defined in Section 5 of [RFC9794]. That material is provided to motivate the extension and may be developed as separate work.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Terms from [I-D.ietf-hpke-hpke] are used without redefinition. The following additional term is used herein:

* *AKEM:* a KEM whose encapsulation additionally takes the sender's static private key and implicitly binds the shared secret to it.

3. Authenticated Mode Extensions to [I-D.ietf-hpke-hpke]

This section specifies the additions to [I-D.ietf-hpke-hpke] required to restore mode_auth and mode_auth_psk. These extensions are defined so that the externally observable behavior of the existing HPKE modes is unchanged, although this document modifies the VerifyPSKInputs() procedure in [I-D.ietf-hpke-hpke].

3.1. Mode Identifiers

The reserved entries for values 0x02 and 0x03 in Table 1 of [I-D.ietf-hpke-hpke] are replaced with the following mode identifiers, as originally specified in Table 1 of [RFC9180]:

```
mode_auth     = 0x02
mode_auth_psk = 0x03
```

3.2. DHKEM Extension: AuthEncap() and AuthDecap()

The following two functions are added to the DHKEM, extending it to an AKEM. They are reproduced verbatim from Section 4.1 of [RFC9180]. All helper functions (GenerateKeyPair, DH, SerializePublicKey, DeserializePublicKey, ExtractAndExpand) are as defined in [I-D.ietf-hpke-hpke].

```
def AuthEncap(pkR, skS):
  skE, pkE = GenerateKeyPair()
  dh = concat(DH(skE, pkR), DH(skS, pkR))
  enc = SerializePublicKey(pkE)

  pkRm = SerializePublicKey(pkR)
  pkSm = SerializePublicKey(pk(skS))

  kem_context = concat(enc, pkRm, pkSm)
  shared_secret = ExtractAndExpand(dh, kem_context)

  return shared_secret, enc

def AuthDecap(enc, skR, pkS):
  pkE = DeserializePublicKey(enc)
  dh = concat(DH(skR, pkE), DH(skR, pkS))

  pkRm = SerializePublicKey(pk(skR))
  pkSm = SerializePublicKey(pkS)
  kem_context = concat(enc, pkRm, pkSm)

  shared_secret = ExtractAndExpand(dh, kem_context)
  return shared_secret
```

Note that the AuthEncap() and AuthDecap() functions are vulnerable to key-compromise impersonation (KCI): the assurance that the shared secret was generated by the holder of the private key skS does not hold if the recipient private key skR is compromised. See Section 6 for further discussion.

3.3. VerifyPSKInputs

The VerifyPSKInputs() function defined in Section 5.1 of [RFC9180] and [I-D.ietf-hpke-hpke] is extended to handle the two new modes. The updated function replaces the original:

```
def VerifyPSKInputs(mode, psk, psk_id):
  got_psk = (psk != default_psk)
  got_psk_id = (psk_id != default_psk_id)
  if got_psk != got_psk_id:
    raise Exception("Inconsistent PSK inputs")

  if got_psk and mode in [mode_base, mode_auth]:
    raise Exception("PSK input provided when not needed")
  if (not got_psk) and mode in [mode_psk, mode_auth_psk]:
    raise Exception("Missing required PSK input")
```

The only change from the original is that mode_base is replaced by [mode_base, mode_auth] and mode_psk is replaced by [mode_psk, mode_auth_psk] in the final two guards.

3.4. Setup Functions

The following four setup functions are reproduced verbatim from Sections 5.1.3 and 5.1.4 of [RFC9180]. KeyScheduleS()/KeyScheduleR() and AuthEncap()/AuthDecap() are as defined in [I-D.ietf-hpke-hpke] and Section 3.2 respectively.

```
def SetupAuthS(pkR, info, skS):
  shared_secret, enc = AuthEncap(pkR, skS)
  return enc, KeyScheduleS(mode_auth, shared_secret, info,
                           default_psk, default_psk_id)

def SetupAuthR(enc, skR, info, pkS):
  shared_secret = AuthDecap(enc, skR, pkS)
  return KeyScheduleR(mode_auth, shared_secret, info,
                      default_psk, default_psk_id)

def SetupAuthPSKS(pkR, info, psk, psk_id, skS):
  shared_secret, enc = AuthEncap(pkR, skS)
  return enc, KeyScheduleS(mode_auth_psk, shared_secret, info,
                           psk, psk_id)

def SetupAuthPSKR(enc, skR, info, psk, psk_id, pkS):
  shared_secret = AuthDecap(enc, skR, pkS)
  return KeyScheduleR(mode_auth_psk, shared_secret, info,
                      psk, psk_id)
```

3.5. Input Validation and Error Handling

In addition to the validation requirements in Section 7.1.4 of [I-D.ietf-hpke-hpke], the recipient MUST validate the sender's static public key pkS before use in AuthDecap(), applying the same validation rules as for other public key inputs. Validation failure MUST yield a ValidationError.

4. Example Applications (Informative)

This section is informative. It illustrates how the authenticated modes defined in Section 3 can be used. Any AEAD identifier from [I-D.ietf-hpke-hpke] may be used; the resulting context supports Seal(), Open(), and Export(). Key generation follows GenerateKeyPair() from [I-D.ietf-hpke-hpke].

The modes may be used directly via their setup functions. For mode_auth, the sender calls SetupAuthS(pkR, info, skS) and the receiver calls SetupAuthR(enc, skR, info, pkS). For mode_auth_psk, the sender calls SetupAuthPSKS() and the receiver calls SetupAuthPSKR(), with a shared PSK and PSK identifier, as defined in Section 3.4.

4.1. Hybrid PQ/T Profile (mode_auth_psk with PQ-KEM PSK)

The following terms are used in this section:

* *PQ-KEM:* a post-quantum KEM, e.g., ML-KEM [FIPS203] or an algorithm from [I-D.ietf-hpke-pq].

* PQKEM.Encap(pkR_pq): PQ-KEM encapsulation; returns (ss_pq, enc_pq).

* PQKEM.Decap(enc_pq, skR_pq): PQ-KEM decapsulation; returns ss_pq.

* Nenc_pq: the fixed ciphertext length of the chosen PQ-KEM, in bytes.

The sender encapsulates a PQ-KEM to the receiver's PQ public key pkR_pq, using the resulting shared secret ss_pq as the HPKE PSK and the PQ ciphertext enc_pq as the PSK identifier. The receiver's PQ public key pkR_pq is included in info to bind it to the key schedule. The combined encapsulation is concat(enc_dh, enc_pq); Nenc bytes are parsed as enc_dh and the remaining Nenc_pq bytes as enc_pq.

```
def HybridSetupS(pkR, pkR_pq, skS, info):
  ss_pq, enc_pq = PQKEM.Encap(pkR_pq)
  enc_dh, ctx = SetupAuthPSKS(pkR, concat(info, pkR_pq), ss_pq, enc_pq, skS)
  return concat(enc_dh, enc_pq), ctx

def HybridSetupR(enc, skR, skR_pq, pkR_pq, pkS, info):
  enc_dh, enc_pq = enc[:Nenc], enc[Nenc:]
  ss_pq = PQKEM.Decap(enc_pq, skR_pq)
  return SetupAuthPSKR(enc_dh, skR, concat(info, pkR_pq), ss_pq, enc_pq, pkS)
```

Implementations should verify len(enc) == Nenc + Nenc_pq and reject encapsulations of any other length. A fresh (ss_pq, enc_pq) pair should be generated for each encapsulation; reuse of a prior enc_pq is prohibited. The suite_id in the HPKE key schedule reflects only the classical ciphersuite (KEM_ID, KDF_ID, AEAD_ID); the PQ-KEM algorithm identity should be conveyed via application-layer framing or the info parameter when multiple PQ-KEM algorithms are supported.

Hybrid confidentiality. KeyScheduleS()/KeyScheduleR() delegate to CombineSecrets, for which [I-D.ietf-hpke-hpke] defines two variants. In CombineSecrets_TwoStage(), the combination is secret = LabeledExtract(dhkem_shared_secret, "secret", psk), equivalent to HKDF-Extract(salt = dhkem_shared_secret, IKM = ss_pq) [RFC5869]. In CombineSecrets_OneStage(), dhkem_shared_secret and psk are length-prefixed and concatenated before a single LabeledDerive() call. In both cases, dhkem_shared_secret and ss_pq enter the combination as independent inputs. The intended design property is that secret remains pseudorandom as long as at least one of the two inputs is--- meaning an adversary would need to attack both the classical DH-based component and the PQ-KEM to recover secret. Whether this property holds formally for a specific CombineSecrets variant depends on that variant’s security analysis, which is outside the scope of this document. Authentication remains entirely classical; a quantum adversary that breaks the DH assumption can also forge sender authentication, so post-quantum sender authentication would require an additional PQ signature. Note that [Alwen2023] describes a related hybrid construction in which a PQ _AKEM_ (rather than a plain KEM) is used to generate the PSK, which would additionally provide post-quantum sender authentication; that stronger construction is outside the scope of this document.

PSK freshness. The ML-KEM shared secret ss_pq satisfies the entropy requirement in Section 9.5 of [I-D.ietf-hpke-hpke] (32 bytes of uniform randomness). The prohibition on enc_pq reuse above ensures a fresh PSK per session.

5. HPKE-Auth Profiles (Informative)

This section is informative. The profiles below are suggested KEM and KDF parameter sets for the example applications in Section 4; they are not registered by this document. Because AEAD_ID is selected by the application, these are parameter sets, not fully determined ciphersuites. KEM_ID and KDF_ID are drawn from the registries in [I-D.ietf-hpke-hpke].

Profile	KEM_ID	KDF_ID	PQKEM	Nenc	Nenc_pq
HPKE-Auth-X25519-SHA256	0x0020	0x0001	---	32	---
HPKE-Auth-P256-SHA256	0x0010	0x0001	---	65	---
HPKE-Auth-X448-SHA512	0x0021	0x0003	---	56	---
HPKE-Auth-Hybrid-X25519-SHA256-MLKEM768	0x0020	0x0001	ML-KEM-768	32	1088
HPKE-Auth-Hybrid-X25519-SHA256-MLKEM1024	0x0020	0x0001	ML-KEM-1024	32	1568
HPKE-Auth-Hybrid-P256-SHA256-MLKEM768	0x0010	0x0001	ML-KEM-768	65	1088

Table 1

ML-KEM parameters are from [FIPS203]. HPKE-Auth-Hybrid-X25519-SHA256-MLKEM768 is the suggested default hybrid profile, yielding a combined encapsulation of 1120 bytes.

6. Security Considerations

The sender-authentication and key-compromise impersonation (KCI) properties of mode_auth and mode_auth_psk are as described in Sections 9.1 and 9.1.1 of [RFC9180], which apply without change to the functions defined in Section 3. Security properties specific to the hybrid PQ/T construction are discussed informatively in Section 4.1.

The formal security of the DHKEM authenticated modes under the Gap-DH assumption is established in [Alwen2021]. The security of mode_auth_psk---termed AuthPSK in [Alwen2023]---is analyzed there as the pskAPKE scheme.

7. IANA Considerations

This document requests no IANA actions; all identifiers are drawn from registries defined in [I-D.ietf-hpke-hpke].

8. References

8.1. Normative References

[I-D.ietf-hpke-hpke]
Barnes, R., Bhargavan, K., Lipp, B., and C. A. Wood, "Hybrid Public Key Encryption", Work in Progress, Internet-Draft, draft-ietf-hpke-hpke-03, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-hpke-hpke-03>.

[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>.

8.2. Informative References

[Alwen2021]
Alwen, J., Blanchet, B., Hauck, E., Kiltz, E., Lipp, B., and D. Riepel, "HPKE: Hybrid Public Key Encryption", Advances in Cryptology -- EUROCRYPT 2021, LNCS 12696, pp. 396-427, DOI 10.1007/978-3-030-77886-6_14, 2021, <https://doi.org/10.1007/978-3-030-77886-6_14>.

[Alwen2023]
Alwen, J., Janneck, J., Kiltz, E., and B. Lipp, "The Pre-Shared Key Modes of HPKE", Advances in Cryptology -- ASIACRYPT 2023, 2023, <https://eprint.iacr.org/2023/1480>.

[FIPS203] National Institute of Standards and Technology, "Module-Lattice-Based Key-Encapsulation Mechanism Standard", FIPS 203, August 2024, <https://doi.org/10.6028/NIST.FIPS.203>.

[I-D.ietf-hpke-pq]
Barnes, R. and D. Connolly, "Post-Quantum and Post-Quantum/Traditional Hybrid Algorithms for HPKE", Work in Progress, Internet-Draft, draft-ietf-hpke-pq-04, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-hpke-pq-04>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <https://www.rfc-editor.org/rfc/rfc5869>.

[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>.

[RFC9794] Driscoll, F., Parsons, M., and B. Hale, "Terminology for Post-Quantum Traditional Hybrid Schemes", RFC 9794, DOI 10.17487/RFC9794, June 2025, <https://www.rfc-editor.org/rfc/rfc9794>.

Acknowledgments

_TK_

Authors' Addresses

Cory Francis Myers (editor)
Freedom of the Press Foundation
Email: cfm@acm.org

Rowen S. (editor)
Freedom of the Press Foundation
Email: ro@freedom.press