LAMPS
Internet-Draft
Intended status: Standards Track
Expires: 7 November 2026

D. Van Geest
CryptoNext Security
M. Ounsworth
J. Gray
Entrust
J. Klaussner
Bundesdruckerei GmbH

6 May 2026

Composite ML-KEM for use in Cryptographic Message Syntax (CMS)
draft-ietf-lamps-cms-composite-kem-01

Abstract

Composite ML-KEM defines combinations of ML-KEM with RSA-OAEP, ECDH, X25519, and X448. This document specifies the conventions for using Composite ML-KEM algorithms with the Cryptographic Message Syntax (CMS) using the KEMRecipientInfo structure defined in “Using Key Encapsulation Mechanism (KEM) Algorithms in the Cryptographic Message Syntax (CMS)” (RFC 9629).

About This Document

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

The latest revision of this draft can be found at https://lamps-wg.github.io/draft-composite-kem/draft-ietf-lamps-pq-composite-kem.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-lamps-cms-composite-kem/.

Discussion of this document takes place on the LAMPS Working Group mailing list (mailto:spams@ietf.org), which is archived at https://datatracker.ietf.org/wg/lamps/about/. Subscribe at https://www.ietf.org/mailman/listinfo/spams/.

Source for this draft and an issue tracker can be found at https://github.com/lamps-wg/draft-composite-kem.

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/.

This Internet-Draft will expire on 7 November 2026.

Copyright Notice

Table of Contents

1. Introduction

[I-D.ietf-lamps-pq-composite-kem] defines a collection of Key Encapsulation Mechanism (KEM) algorithms, referred to as Composite ML-KEM, which combine ML-KEM [FIPS203] with traditional algorithms RSA-OAEP, ECDH, X25519, and X448. [RFC9629] defines the KEMRecipientInfo structure for the use of KEM algorithms for the Cryptographic Message Syntax (CMS) [RFC5652] enveloped-data content type, the CMS authenticated-data content type, and the CMS authenticated-enveloped-data content type. This document acts as a companion to [I-D.ietf-lamps-pq-composite-kem] by providing conventions for using Composite ML-KEM algorithms with the KEMRecipientInfo structure within the CMS.

1.1. ASN.1

CMS values are generated using ASN.1 [X680], using the Basic Encoding Rules (BER) and the Distinguished Encoding Rules (DER) [X690].

1.2. 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. These words may also appear in this document in lower case as plain English words, absent their normative meanings.

1.3. Composite ML-KEM

ML-KEM is a lattice-based KEM using Module Learning with Errors as its underlying primitive. It was standardized with three parameter sets: ML-KEM-512, ML-KEM-768, and ML-KEM-1024. Composite ML-KEM pairs ML-KEM-768 or ML-KEM-1024 with RSA-OAEP, ECDH, X25519, or X448 at similar security levels such that the shared secret key from each component algorithm is combined into a single shared secret key. Composite ML-KEM does not provide pairings based on ML-KEM-512.

All KEM algorithms provide three functions: KeyGen(), Encapsulate(), and Decapsulate().

The following summarizes these three functions for Composite ML-KEM:

KeyGen() -> (ek, dk): Generate the public encapsulation key (ek) and a private decapsulation key (dk). Section 3.1 of [I-D.ietf-lamps-pq-composite-kem] specifies the key generation algorithm for Composite ML-KEM.

Encapsulate(ek) -> (c, ss): Given the recipient's public key (ek), produce both a ciphertext (c) to be passed to the recipient and a shared secret (ss) for use by the originator. Section 3.2 of [I-D.ietf-lamps-pq-composite-kem] specifies the encapsulation algorithm for Composite ML-KEM.

Decapsulate(dk, c) -> ss: Given the private key (dk) and the ciphertext (c), produce the shared secret (ss) for the recipient. Section 3.3 of [I-D.ietf-lamps-pq-composite-kem] specifies the decapsulation algorithm for Composite ML-KEM.

2. Use of Composite ML-KEM in the CMS

Composite ML-KEM algorithms MAY be employed for one or more recipients in the CMS enveloped-data content type [RFC5652], the CMS authenticated-data content type [RFC5652], or the CMS authenticated-enveloped-data content type [RFC5083]. In each case, the KEMRecipientInfo [RFC9629] type is used with the Composite ML-KEM algorithm to securely transfer the content-encryption key from the originator to the recipient.

Processing a Composite ML-KEM algorithm with KEMRecipientInfo follows the same steps as Section 2 of [RFC9629]. To support the Composite ML-KEM algorithm, a CMS originator MUST implement the Encapsulate() function and a CMS recipient MUST implement the Decapsulate() function.

2.1. RecipientInfo Conventions

When a Composite ML-KEM algorithm is employed for a recipient, the RecipientInfo alternative for that recipient MUST be OtherRecipientInfo using the KEMRecipientInfo structure as defined in [RFC9629].

The fields of the KEMRecipientInfo have the following meanings:

version
: The syntax version number; it MUST be 0.

rid
: Identifies the recipient's certificate or public key.

kem
: Identifies the KEM algorithm; it MUST contain one of the Composite ML-KEM OIDs in Section 3.

kemct
: The ciphertext produced for this recipient.

kdf
: Identifies the key derivation algorithm. Note that the Key Derivation Function (KDF) used for CMS RecipientInfo process MAY be different than the KDF used within the Composite ML-KEM algorithm. Implementations MUST support the HMAC-based Key Derivation Function (HKDF) [RFC5869] with SHA-256 [FIPS180], using the id-alg-hkdf-with-sha256 KDF object identifier (OID) [RFC8619]. As specified in [RFC8619], the parameter field MUST be absent when this OID appears within the ASN.1 type AlgorithmIdentifier. Implementations MAY support other KDFs as well.

kekLength
: The size of the key-encryption key in octets.

ukm
: Optional input to the KDF. The secure use of Composite ML-KEM in CMS does not depend on the use of a ukm value, so this document does not place any requirements on this value. See Section 3 of [RFC9629] for more information about the ukm parameter.

wrap
: Identifies a key-encryption algorithm used to encrypt the content-encryption key. Implementations MUST support the AES-Wrap-256 [RFC3394] key-encryption algorithm using the id-aes256-wrap key-encryption algorithm OID [RFC3565]. Implementations MAY support other key-encryption algorithms as well.

Appendix B contains an example of establishing a content-encryption key using Composite ML-KEM in the KEMRecipientInfo type.

2.2. Underlying Components

When Composite ML-KEM is employed in the CMS, the underlying components used within the KEMRecipientInfo structure SHOULD be consistent with a minimum desired security level. Several security levels have been identified [SP.800-57pt1r5].

If underlying components other than those specified in Section 2.1 are used, then the following table gives the minimum requirements on the components used with Composite ML-KEM in the KEMRecipientInfo type in order to satisfy the KDF and key wrapping algorithm requirements from Section 7 of [RFC9629]. The components are chosen based on the ML-KEM variant used within the Composite ML-KEM algorithm.

Security Strength	ML-KEM Variant	KDF Preimage Strength	Symmetric Key-Encryption Strength
192-bit	ML-KEM-768	192-bit	192-bit (*)
256-bit	ML-KEM-1024	256-bit	256-bit

Table 1: Composite ML-KEM KEMRecipientInfo Component Security Levels

(*) In the case of AES Key Wrap, a 256-bit key is typically used because AES-192 is not as commonly deployed.

2.2.1. Use of the HKDF-Based Key Derivation Function

The HKDF function is a composition of the HKDF-Extract and HKDF-Expand functions.

```
HKDF(salt, IKM, info, L)
  = HKDF-Expand(HKDF-Extract(salt, IKM), info, L)
```

When used with KEMRecipientInfo, the salt parameter is unused; that is, it is the zero-length string "". The IKM, info, and L parameters correspond to the same KDF inputs from Section 5 of [RFC9629]. The info parameter is independently generated by the originator and recipient. Implementations MUST confirm that L is consistent with the key size of the key-encryption algorithm.

2.3. Certificate Conventions

[RFC5280] specifies the profile for using X.509 certificates in Internet applications. A recipient static public key is needed for Composite ML-KEM and the originator obtains that public key from the recipient’s certificate. The conventions for carrying Composite ML-KEM public keys are specified in [I-D.ietf-lamps-pq-composite-kem].

2.4. SMIME Capabilities Attribute Conventions

Section 2.5.2 of [RFC8551] defines the SMIMECapabilities attribute to announce a partial list of algorithms that an S/MIME implementation can support. When constructing a CMS signed-data content type [RFC5652], a compliant implementation MAY include the SMIMECapabilities attribute that announces support for one or more of the Composite ML-KEM algorithm identifiers.

The SMIMECapability SEQUENCE representing the Composite ML-KEM algorithm MUST include one of the Composite ML-KEM OIDs in the capabilityID field. When one of the Composite ML-KEM OIDs appears in the capabilityID field, the parameters MUST NOT be present.

3. Identifiers

All identifiers used to indicate Composite ML-KEM within the CMS are defined in [I-D.ietf-lamps-pq-composite-kem], [RFC8619], and [RFC3565]; they are reproduced here for convenience:

```
-- Composite ML-KEM OIDs

id-MLKEM768-RSA2048-SHA3-256 OBJECT IDENTIFIER ::= {
  iso(1) identified-organization(3) dod(6) internet(1) security(5)
  mechanisms(5) pkix(7) alg(6) 55 }

id-MLKEM768-RSA3072-SHA3-256 OBJECT IDENTIFIER ::= {
  iso(1) identified-organization(3) dod(6) internet(1) security(5)
  mechanisms(5) pkix(7) alg(6) 56 }

id-MLKEM768-RSA4096-SHA3-256 OBJECT IDENTIFIER ::= {
  iso(1) identified-organization(3) dod(6) internet(1) security(5)
  mechanisms(5) pkix(7) alg(6) 57 }

id-MLKEM768-X25519-SHA3-256 OBJECT IDENTIFIER ::= {
  iso(1) identified-organization(3) dod(6) internet(1) security(5)
  mechanisms(5) pkix(7) alg(6) 58 }

id-MLKEM768-ECDH-P256-SHA3-256 OBJECT IDENTIFIER ::= {
  iso(1) identified-organization(3) dod(6) internet(1) security(5)
  mechanisms(5) pkix(7) alg(6) 59 }

id-MLKEM768-ECDH-P384-SHA3-256 OBJECT IDENTIFIER ::= {
  iso(1) identified-organization(3) dod(6) internet(1) security(5)
  mechanisms(5) pkix(7) alg(6) 60 }

id-MLKEM768-ECDH-brainpoolP256r1-SHA3-256 OBJECT IDENTIFIER ::= {
  iso(1) identified-organization(3) dod(6) internet(1) security(5)
  mechanisms(5) pkix(7) alg(6) 61 }

id-MLKEM1024-RSA3072-SHA3-256 OBJECT IDENTIFIER ::= {
  iso(1) identified-organization(3) dod(6) internet(1) security(5)
  mechanisms(5) pkix(7) alg(6) 62 }

id-MLKEM1024-ECDH-P384-SHA3-256 OBJECT IDENTIFIER ::= {
  iso(1) identified-organization(3) dod(6) internet(1) security(5)
  mechanisms(5) pkix(7) alg(6) 63 }
```

```
id-MLKEM1024-ECDH-brainpoolP384r1-SHA3-256 OBJECT IDENTIFIER ::= {
  iso(1) identified-organization(3) dod(6) internet(1) security(5)
  mechanisms(5) pkix(7) alg(6) 64 }

id-MLKEM1024-X448-SHA3-256 OBJECT IDENTIFIER ::= {
  iso(1) identified-organization(3) dod(6) internet(1) security(5)
  mechanisms(5) pkix(7) alg(6) 65 }

id-MLKEM1024-ECDH-P521-SHA3-256 OBJECT IDENTIFIER ::= {
  iso(1) identified-organization(3) dod(6) internet(1) security(5)
  mechanisms(5) pkix(7) alg(6) 66 }

-- KEMRecipientInfo.kdf OIDs

id-alg-hkdf-with-sha256 OBJECT IDENTIFIER ::= { iso(1)
    member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
    smime(16) alg(3) 28 }

-- KEMRecipientInfo.wrap OIDs

aes OBJECT IDENTIFIER ::= { joint-iso-itu-t(2) country(16) us(840)
    organization(1) gov(101) csor(3) nistAlgorithms(4) 1 }

id-aes256-wrap OBJECT IDENTIFIER ::= { aes 45 }
```

4. Security Considerations

The Security Considerations sections of [I-D.ietf-lamps-pq-composite-kem] and [RFC9629] apply to this specification as well.

Implementations MUST protect the Composite ML-KEM private key, the key-encryption key, the content-encryption key, message-authentication key, and the content-authenticated-encryption key. Of these keys, all but the private key are ephemeral and MUST be wiped after use. Disclosure of the Composite ML-KEM private key could result in the compromise of all messages protected with that key. Disclosure of the key-encryption key, the content-encryption key, or the content-authenticated-encryption key could result in the compromise of the associated encrypted content. Disclosure of the key-encryption key, the message-authentication key, or the content-authenticated-encryption key could allow modification of the associated authenticated content.

Additional considerations related to key management may be found in [SP.800-57pt1r5].

The generation of private keys relies on random numbers, as does the encapsulation function of Composite ML-KEM. The use of inadequate pseudorandom number generators (PRNGs) to generate these values can result in little or no security. If the random value is weakly chosen, then an attacker may find it much easier to reproduce the PRNG environment that produced the keys or ciphertext, searching the resulting small set of possibilities for a matching public key or ciphertext value, rather than performing a more complex algorithmic attack against Composite ML-KEM.

Composite ML-KEM encapsulation and decapsulation only outputs a shared secret and ciphertext. Implementations MUST NOT use intermediate values directly for any purpose.

Implementations SHOULD NOT reveal information about intermediate values or calculations, whether by timing or other "side channels"; otherwise an opponent may be able to determine information about the keying data and/or the recipient’s private key. Although not all intermediate information may be useful to an opponent, it is preferable to conceal as much information as is practical, unless analysis specifically indicates that the information would not be useful to an opponent.

Generally, good cryptographic practice employs a given Composite ML-KEM key pair in only one scheme. This practice avoids the risk that vulnerability in one scheme may compromise the security of the other and may be essential to maintain provable security.

5. IANA Considerations

IANA is requested to allocate a value from the "SMI Security for S/MIME Module Identifier (1.2.840.113549.1.9.16.0)" registry for the included ASN.1 module.

* Decimal: IANA Assigned - *Replace TBDMOD*

* Description: Composite-MLKEM-CMS-2026 - id-mod-composite-mlkem-cms-2026

* References: This Document

  | RFC EDITOR: Please replace TBDCompositeMOD in the ASN.1 module
  | with with module number assigned to id-mod-composite-mlkem-2025
  | in [I-D.ietf-lamps-pq-composite-kem].

6. References

6.1. Normative References

[FIPS180] "Secure hash standard", National Institute of Standards and Technology (U.S.), DOI 10.6028/nist.fips.180-4, 2015, <https://doi.org/10.6028/nist.fips.180-4>.

[I-D.ietf-lamps-pq-composite-kem] Ounsworth, M., Gray, J., Pala, M., Klauner, J., and S. Fluhrer, "Composite ML-KEM for use in X.509 Public Key Infrastructure", Work in Progress, Internet-Draft, draft-ietf-lamps-pq-composite-kem-14, 27 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-pq-composite-kem-14>.

[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>.

[RFC3394] Schaad, J. and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", RFC 3394, DOI 10.17487/RFC3394, September 2002, <https://www.rfc-editor.org/rfc/rfc3394>.

[RFC3565] Schaad, J., "Use of the Advanced Encryption Standard (AES) Encryption Algorithm in Cryptographic Message Syntax (CMS)", RFC 3565, DOI 10.17487/RFC3565, July 2003, <https://www.rfc-editor.org/rfc/rfc3565>.

[RFC5083] Housley, R., "Cryptographic Message Syntax (CMS) Authenticated-Enveloped-Data Content Type", RFC 5083, DOI 10.17487/RFC5083, November 2007, <https://www.rfc-editor.org/rfc/rfc5083>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/rfc/rfc5280>.

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

[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>.

[RFC5911] Hoffman, P. and J. Schaad, "New ASN.1 Modules for Cryptographic Message Syntax (CMS) and S/MIME", RFC 5911, DOI 10.17487/RFC5911, June 2010, <https://www.rfc-editor.org/rfc/rfc5911>.

[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>.

[RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/ Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <https://www.rfc-editor.org/rfc/rfc8551>.

[RFC8619] Housley, R., "Algorithm Identifiers for the HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 8619, DOI 10.17487/RFC8619, June 2019, <https://www.rfc-editor.org/rfc/rfc8619>.

[RFC9629] Housley, R., Gray, J., and T. Okubo, "Using Key Encapsulation Mechanism (KEM) Algorithms in the Cryptographic Message Syntax (CMS)", RFC 9629, DOI 10.17487/RFC9629, August 2024, <https://www.rfc-editor.org/rfc/rfc9629>.

[X680] ITU-T, "Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, ISO/IEC 8824-1:2021, February 2021, <https://www.itu.int/rec/T-REC-X.680>.

[X690] ITU-T, "Information technology - Abstract Syntax Notation One (ASN.1): ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, ISO/IEC 8825-1:2021, February 2021, <https://www.itu.int/rec/T-REC-X.690>.

6.2. Informative References

[FIPS203] "Module-lattice-based key-encapsulation mechanism standard", National Institute of Standards and Technology (U.S.), DOI 10.6028/nist.fips.203, August 2024, <https://doi.org/10.6028/nist.fips.203>.

[RFC9690] Housley, R. and S. Turner, "Use of the RSA-KEM Algorithm in the Cryptographic Message Syntax (CMS)", RFC 9690, DOI 10.17487/RFC9690, February 2025, <https://www.rfc-editor.org/rfc/rfc9690>.

[RFC9936] Prat, J., Ounsworth, M., and D. Van Geest, "Use of ML-KEM in the Cryptographic Message Syntax (CMS)", RFC 9936, DOI 10.17487/RFC9936, March 2026, <https://www.rfc-editor.org/rfc/rfc9936>.

[SP.800-57pt1r5] National Institute of Standards and Technology (NIST), "Recommendation for Key Management: Part 1 General", May 2020, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf>.

Appendix A. ASN.1 Module

This appendix includes the ASN.1 module [X680] for Composite ML-KEM. This module imports objects from [RFC5911], [RFC9629], [RFC8619], [I-D.ietf-lamps-pq-composite-kem].

```
<CODE BEGINS>
Composite-MLKEM-CMS-2026
  { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
        id-smime(16) id-mod(0)
        id-mod-composite-mlkem-cms-2026(TBDMOD) }

DEFINITIONS IMPLICIT TAGS ::= BEGIN

EXPORTS ALL;

IMPORTS

SMIME-CAPS
  FROM AlgorithmInformation-2009  -- [RFC5911]
    { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-algorithmInformation-02(58) }

KEM-ALGORITHM
  FROM KEMAlgorithmInformation-2023  -- [RFC9629]
      { iso(1) identified-organization(3) dod(6) internet(1)
        security(5) mechanisms(5) pkix(7) id-mod(0)
        id-mod-kemAlgorithmInformation-2023(109) }

kda-hkdf-with-sha256
  FROM HKDF-OID-2019  -- [RFC8619]
      { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
        pkcs-9(9) smime(16) modules(0) id-mod-hkdf-oid-2019(68) }

kwa-aes256-wrap
```

```
    FROM CMSAesRsaesOaep-2009  -- [RFC5911]
        { iso(1) member-body(2) us(840) rsadsi(113549)
        pkcs(1) pkcs-9(9) smime(16) modules(0)
        id-mod-cms-aes-02(38) }

  kema-MLKEM768-RSA2048-SHA3-256, kema-MLKEM768-RSA3072-SHA3-256,
  kema-MLKEM768-RSA4096-SHA3-256, kema-MLKEM768-X25519-SHA3-256,
  kema-MLKEM768-ECDH-P256-SHA3-256, kema-MLKEM768-ECDH-P384-SHA3-256,
  kema-MLKEM768-ECDH-brainpoolP256r1-SHA3-256,
  kema-MLKEM1024-RSA3072-SHA3-256, kema-MLKEM1024-ECDH-P384-SHA3-256,
  kema-MLKEM1024-ECDH-brainpoolP384r1-SHA3-256, kema-MLKEM1024-X448,
  kema-MLKEM1024-ECDH-P521-SHA3-256
    FROM Composite-MLKEM-2025  -- [I-D.ietf-lamps-pq-composite-kem]
        { iso(1) identified-organization(3) dod(6) internet(1)
        security(5) mechanisms(5) pkix(7) id-mod(0)
        id-mod-composite-mlkem-2025(TBDCompositeMOD) }
  ;

  -- Updates for the SMIME-CAPS Set from RFC 5911

  SMimeCapsSet SMIME-CAPS ::=
     { kema-MLKEM768-RSA2048-SHA3-256.&smimeCaps |
       kema-MLKEM768-RSA3072-SHA3-256.&smimeCaps |
       kema-MLKEM768-RSA4096-SHA3-256.&smimeCaps |
       kema-MLKEM768-X25519-SHA3-256.&smimeCaps |
       kema-MLKEM768-ECDH-P256-SHA3-256.&smimeCaps |
       kema-MLKEM768-ECDH-P384-SHA3-256.&smimeCaps |
       kema-MLKEM768-ECDH-brainpoolP256r1-SHA3-256.&smimeCaps |
       kema-MLKEM1024-RSA3072-SHA3-256.&smimeCaps |
       kema-MLKEM1024-ECDH-P384-SHA3-256.&smimeCaps |
       kema-MLKEM1024-ECDH-brainpoolP384r1-SHA3-256.&smimeCaps |
       kema-MLKEM1024-X448.&smimeCaps |
       kema-MLKEM1024-ECDH-P521-SHA3-256.&smimeCaps |
       kda-hkdf-with-sha256.&smimeCaps |
       kwa-aes256-wrap.&smimeCaps,
       ... }

  END
  <CODE ENDS>
```

Appendix B. Composite ML-KEM CMS Authenticated-Enveloped-Data Example

This example shows the establishment of an AES-256 content-encryption key using:

* id-MLKEM768-ECDH-P256-SHA3-256;

* KEMRecipientInfo key derivation using HKDF with SHA-256; and

* KEMRecipientInfo key wrap using AES-256-KEYWRAP.

In real-world use, the originator would encrypt the content-encryption key in a manner that would allow decryption with their own private key as well as the recipient's private key. This is omitted in an attempt to simplify the example.

B.1. Originator CMS Processing

Alice obtains Bob's id-MLKEM768-ECDH-P256-SHA3-256 public key:


```
-----BEGIN PUBLIC KEY-----
MIIE8jAKBggrBgEFBQcGOwOCBOIAjssyhemrdzqTE5MNKmCruyoBancRo6y8A5Aw
uDSZ2AwdyWZCWnG6R2sP0vlXCNglq7BvHQTF6BObo+tT+ZmJSnsy1YaSKuyf8yw3
vxyr1ScErsY0r2s1gfprFhYLUQNNoQd+YViNTAedvTBeQGJLgJOozEIFKiwHbvcu
ddc2qnVGqNW0SSjF5QoE/lcwgvpYpZosHRePTYhm87hCdQJldwB6PZmgxZqfyeh0
vXsB+KGw+XLJaOgb2vtl+Nmbm5cyUvOgHItyxQIadhO8iHEIubYzKdlOcDGiP1qE
lCiMNslC+TGcl4Ij2SBQgyfLGpw2jpd3xJxyqjFZeeGrgidfRdVRtUyqUbCGHsLH
ITFJz/h9vqe+S5N6vJmHMFCTHtu0CSiz5pK/RepjWKZiWhYdz1k9u8GWo4q3h8SH
MnQKoSJq3Fx84Kl/ghUNxXB//2w+a/ekHuUzK8QbyWIqm2RSHkOj91hApCMG0tMn
I0xoEqm/ewo3xZA5MJUZEdIkmoN+d5fLwVlUOHS0TWVrrCWAvttq4HySFMBU3vGd
qGQPv2ZLd+NgjzmRMwc3unQh4KIXBspNK+lJgyU57QaDWtVq/yoXjhw5YQlYmckb
NuIeTjpNDqfNxFSuziVQwZQ6VPm/63tjcxy9uMtxw5wTyqfAkyZVKdTLUdcMsJVF
8XE0r6xtCQrNsmWzqhK1kvOUQcITlZEonyN4uXdFweFwXImg/NSK6QUcjKi6iQeI
Gmiy2bKbUIgrIbhyUOZU1ad/vrF+dkBzrPei5BanHBLF5hYVoJcQ+hmg3cVsR1Ke
XaduFdh0UXPNPMiTrClLz/FMdfcbzbe2xdSXHqCsdIBKxGagT3p0H/uTW1WwheFr
DJKBDYBaZxVOwRqP16M7QIOTo8AqeXTOQMSxAdCAx7NR4SttZbQDrjwj1qCj5LQ9
0TW4vUmNsWkaTeQP0LC6f/J5vlEQeIu0vUuggkqmDdaKPwx35SOixjwEwiweXgi1
3bdn68py4/VVYPwXHZaDhakUurZcNEU91nwkrdG03DmbyBlIrwwIClgMwhmSbSGj
Mql2lCPPE/zBJAvL9lfDP3mpoEMB0iCZm6t2IfAN7NLOwFeBOUomqxcvs0OkdxFG
1Gk0hVEis4LHcje/1rBAX8i2Ige6H0AqY5e9X2OHHjVFNzoM8Mavo5trYEksanbG
aJmpADcX8wxjDPhq+VBethWo5Ik0aLW16qGY/jAX6ROrpeRobOiSYNnDLFWI38Mf
cZJuHbBajBe2gwQbTdadu0elr7fMuHd7OOB35vgH7qq0lgOg7LcB8rVgXBeHLsJp
AGS2XTYlKFRNieIo5NSRdkXNkIoHyFZWU9dNUwbKD0CYtxxlj3cKp4AFFHdzCTot
bFIfhoQQxIsZhbPOFCZKjtEQ+UVnjQN1y0E9uym1N7ZjW9OkjeCs0YaooyooNpu6
TOMKu3GjVsqmKvgxk3cIztUouTC2+BC0iYANMUxRbZIiqLGWTsa6rBulSqm5Hpu8
R/LBY0gljpYLFDQJiuYRNvOOIUYEO3ubd8fKY8swLGLNi/LoNwtb+HoJ8szpDsXh
TIalQ0MEsdePvv0LCZ+0N55b1MaFJZieRuCMVPkbeJGWGzb4WVohn4jrsslaNC6y
eZxiXOnPS7EDyzLv2PL/zhedCEbC4A==
-----END PUBLIC KEY-----
```


Bob's id-MLKEM768-ECDH-P256-SHA3-256 public key has the following key identifier:

14F1D8FED21F6103676C752C97D0949537A96AB2

Alice generates a shared secret and ciphertext using Bob's id-MLKEM768-ECDH-P256-SHA3-256 public key:

Shared secret:

9bd51d44ba390ff68176b763cee113007f112aac3c2ddbdde916e713762a389c

Ciphertext:

54b75c945ff83194fbf312214749b114bf6838878b49403f5235be774ae6719d
90531da9ab013a0f8a81dbd6510592f5fbcbe3b1bcf393793f517e758bbdd104
67c1a72b14165233e386ea6b027e482824d33244cc7f810bf7d21cd28e007c84
19087698be0dd7934653f35c8b0164c218dcbcbe28302a813e47a334c9565349
0cf57602c798d63d393e9e6ebc219265aa6562632a618d4c3e4052b20c89f2f0
4ea45554673d1be8911f83ca4a20ce86b59cf0a8c964e5b04707875d56011d48
9291abf2370838dc608fe2e26e617d0ec2c2c643e51caf6716e64c2cc5a185c7
2770ec4185487589156bb47e414180c6cedeacd8dad07ffa9091441fdaced553
c482bd0c41d5b77649f0ed803d07d29682dc2c41d2047d18fddab96529b9f8a2
f0ddd5e4b302fffd8ccbec03d9917ffa6b8d249117146c58fd89a135a96ece67
0c58a559628a68012d4c9a43e5fffb427f0df30da745bb0e4eb334716bb82175
10467b8b58e1be54b30c857522654cbb5787813cb1b2fb957a725f51612187b5
8bb83353f8e56aa6af89c94edd656d92bf66913022fce8f85730617d8a166bbb
24515559915cc472435950bce8c25219fef507a6527a6a3920c2deada7732902
e2d2ae8c60bda518a65da66f93581b7e37042a10464675ab326f603be14e5b55
25634adeb956512a84fdb809af2f37d20a9fd23f67eca7741b49d5b7d74f947a
8b6dac68888b3e3802994c98cccb68c2a0e7927f15e5e7be1400983e22a4c9ef
2babdf244525adde3b4d91e778e21e2bd32c5221f5433f28390128eabf3b181b
85c2c1efd7fae946399bb8fe21de2aed0a3e72cab34d305509547df33a45d17d
013a3ac08f0db69dbff0bae5a4b9c15ee8bdc4e2525b49e6865aeb752051935f
8e2fb5daea912d118671102b8fb4675c3937b389a7c7068cb697b3798fffe9d6
e64156493bb808d067a3eac418a7bf5819e25a740c8349914e5008340c381a07
d870b14b9be9939d306bc702d46a58216df932d2bcb45a3f181bd84f4b9ba992
f3a5f12a5db615ae0598a9c432cf5c0095187a849a93ca0d2d7deedb2db1ced5
3c0d4ae5d2b37eb25e07992452e0188e2d72682f46f0167f0f6a680243eff2c4
b5fcf0235852f99defa4d535a479176fa3a22587511581b3649fe410e302ec1b
9061cb535c492ae57ac126cb49ea47de12a9097c5f8a869d84d4aae903f0bc6f
53a60514cebe02fdc9a204a6be2a66567d589fcbaca36565a3e02d10b16920ea
26b05c50be805d06ece3c9ee7db7eb207c33c1d4ac92ca294f1a0f8fac839fea
0389fae43384a2daa7a4c1f5e92d622f8eb37ac0c55eb9df8d9713de03a0ef8b
2d9306c8530f607d898ba88adb6ad982becd3b05cac32b7dfe767c826e5e8bdf
40120f1a49e84cd11c4c09b07c27591a6032b372a7bd468b09ece07ee6eca6e9
4559f14843dd9a97931e0c06d105aa2237e67a3c128d33ab61dc47986fdafb79
7e60cb44453b4dfbcf7afdfe02f686c720fd65d5381827be62f3322acbc37213
0443d6690e3ae3b17f263f9345998ca26317f757b86d0ad41531b114f5d57fa8
2e5023e5176227d087f765e1421cef31be9c315866838017bb2aa58955bf52ce
6e

Alice encodes the CMSORIforKEMOtherInfo:

3010300b060960864801650304012d020120

Alice derives the key-encryption key from the shared secret and CMSORIforKEMOtherInfo using HKDF with SHA-256:

0555324e4703e672404e2272c555438bd7d04feac1321e05ebae6c0a2336d529

Alice randomly generates a 128-bit content-encryption key:

3424cef0cbc2b67f58351abd1e87b3507a11c90adef7e1cdab8d20b0331b8105

Alice uses AES-256-KEYWRAP to encrypt the content-encryption key with the key-encryption key:

5d13ae00d914a29158a8ef32b23ae5f27b8c293387e3be65fc3dfc19b99d8fdc
552527fd42154b37

Alice encrypts the padded content using AES-256-GCM with the content-encryption key and encodes the AuthEnvelopedData (using KEMRecipientInfo) and ContentInfo, and then sends the result to Bob.

The Base64-encoded result is:

```
-----BEGIN CMS-----
MIIFcgYLKoZIhvcNAQkQAReggguVhMIIFXQIBADGCBQikggUEBgsqhkiG9w0BCRAN
AzCCBPMCAQCAFBTx2P7SH2EDZ2x1LJfQlJU3qWqyMAoGCCsGAQUFBwY7BIIEgVS3
XJRf+DGU+/MSIUdJsRS/aDiHi0lAP1I1vndK5nGdkFMdqasBOg+KgdvWUQWS9fvL
47G885N5P1F+dYu90QRnwacrFBZSM+OG6msCfkgoJNMyRMx/gQv30hzSjgB8hBkI
dpi+DdeTRlPzXIsBZMIY3Ly+KDAqgT5HozTJVlNJDPV2AseY1j05Pp5uvCGSZapl
YmMqYY1MPkBSsgyJ8vBOpFVUZz0b6JEfg8pKIM6GtZzwqMlk5bBHB4ddVgEdSJKR
q/I3CDjcYI/i4m5hfQ7CwsZD5RyvZxbmTCzFoYXHJ3DsQYVIdYkVa7R+QUGAxs7e
rNja0H/6kJFEH9rO1VPEgr0MQdW3dknw7YA9B9KWgtwsQdIEfRj92rllKbn4ovDd
1eSzAv/9jMvsA9mRf/prjSSRFxRsWP2JoTWpbs5nDFilWWKKaAEtTJpD5f/7Qn8N
8w2nRbsOTrM0cWu4IXUQRnuLWOG+VLMMhXUiZUy7V4eBPLGy+5V6cl9RYSGHtYu4
M1P45Wqmr4nJTt1lbZK/ZpEwIvzo+FcwYX2KFmu7JFFVWZFcxHJDWVC86MJSGf71
B6ZSemo5IMLeradzKQLi0q6MYL2lGKZdpm+TWBt+NwQqEEZGdasyb2A74U5bVSVj
St65VlEqhP24Ca8vN9IKn9I/Z+yndBtJ1bfXT5R6i22saIiLPjgCmUyYzMtowqDn
kn8V5ee+FACYPiKkye8rq98kRSWt3jtNked44h4r0yxSIfVDPyg5ASjqvzsYG4XC
we/X+ulGOZu4/iHeKu0KPnLKs00wVQlUffM6RdF9ATo6wI8Ntp2/8LrlpLnBXui9
xOJSW0nmhlrrdSBRk1+OL7Xa6pEtEYZxECuPtGdcOTeziafHBoy2l7N5j//p1uZB
Vkk7uAjQZ6PqxBinv1gZ4lp0DINJkU5QCDQMOBoH2HCxS5vpk50wa8cC1GpYIW35
MtK8tFo/GBvYT0ubqZLzpfEqXbYVrgWYqcQyz1wAlRh6hJqTyg0tfe7bLbHO1TwN
SuXSs36yXgeZJFLgGI4tcmgvRvAWfw9qaAJD7/LEtfzwI1hS+Z3vpNU1pHkXb6Oi
JYdRFYGzZJ/kEOMC7BuQYctTXEkq5XrBJstJ6kfeEqkJfF+Khp2E1KrpA/C8b1Om
BRTOvgL9yaIEpr4qZlZ9WJ/LrKNlZaPgLRCxaSDqJrBcUL6AXQbs48nufbfrIHwz
wdSsksopTxoPj6yDn+oDifrkM4Si2qekwfXpLWIvjrN6wMVeud+NlxPeA6Dviy2T
BshTD2B9iYuoittq2YK+zTsFysMrff52fIJuXovfQBIPGknoTNEcTAmwfCdZGmAy
s3KnvUaLCezgfubspulFWfFIQ92al5MeDAbRBaoiN+Z6PBKNM6th3EeYb9r7eX5g
y0RFO037z3r9/gL2hscg/WXVOBgnvmLzMirLw3ITBEPWaQ4647F/Jj+TRZmMomMX
91e4bQrUFTGxFPXVf6guUCPlF2In0If3ZeFCHO8xvpwxWGaDgBe7KqWJVb9Szm4w
DQYLKoZIhvcNAQkQAxwCASAwCwYJYIZIAWUDBAEtBChdE64A2RSikVio7zKyOuXy
e4wpM4fjvmX8PfwZuZ2P3FUlJ/1CFUs3MDoGCSqGSIb3DQEHATAeBglghkgBZQME
AS4wEQQMhrwH+8PcSlmdkPPoAgEQgA0g+U8GH3sF+VsyV//ABBBfKmBjryzwX+Hi
l6KLp6sQ
-----END CMS-----
```

This result decodes to:

```
   0 1394: SEQUENCE {
   4   11:  OBJECT IDENTIFIER
         :   authEnvelopedData (1 2 840 113549 1 9 16 1 23)
  17 1377:  [0] {
  21 1373:   SEQUENCE {
  25    1:    INTEGER 0
  28 1288:    SET {
  32 1284:     [4] {
  36   11:      OBJECT IDENTIFIER '1 2 840 113549 1 9 16 13 3'
  49 1267:      SEQUENCE {
  53    1:       INTEGER 0
  56   20:       [0]
         :   14 F1 D8 FE D2 1F 61 03 67 6C 75 2C 97 D0 94 95
```

```
        :   37 A9 6A B2
    78   10:       SEQUENCE {
    80    8:         OBJECT IDENTIFIER '1 3 6 1 5 5 7 6 59'
        :         }
    90 1153:       OCTET STRING
        :   54 B7 5C 94 5F F8 31 94 FB F3 12 21 47 49 B1 14
        :   BF 68 38 87 8B 49 40 3F 52 35 BE 77 4A E6 71 9D
        :   90 53 1D A9 AB 01 3A 0F 8A 81 DB D6 51 05 92 F5
        :   FB CB E3 B1 BC F3 93 79 3F 51 7E 75 8B BD D1 04
        :   67 C1 A7 2B 14 16 52 33 E3 86 EA 6B 02 7E 48 28
        :   24 D3 32 44 CC 7F 81 0B F7 D2 1C D2 8E 00 7C 84
        :   19 08 76 98 BE 0D D7 93 46 53 F3 5C 8B 01 64 C2
        :   18 DC BC BE 28 30 2A 81 3E 47 A3 34 C9 56 53 49
        :   0C F5 76 02 C7 98 D6 3D 39 3E 9E 6E BC 21 92 65
        :   AA 65 62 63 2A 61 8D 4C 3E 40 52 B2 0C 89 F2 F0
        :   4E A4 55 54 67 3D 1B E8 91 1F 83 CA 4A 20 CE 86
        :   B5 9C F0 A8 C9 64 E5 B0 47 07 87 5D 56 01 1D 48
        :   92 91 AB F2 37 08 38 DC 60 8F E2 E2 6E 61 7D 0E
        :   C2 C2 C6 43 E5 1C AF 67 16 E6 4C 2C C5 A1 85 C7
        :   27 70 EC 41 85 48 75 89 15 6B B4 7E 41 41 80 C6
        :   CE DE AC D8 DA D0 7F FA 90 91 44 1F DA CE D5 53
        :   C4 82 BD 0C 41 D5 B7 76 49 F0 ED 80 3D 07 D2 96
        :   82 DC 2C 41 D2 04 7D 18 FD DA B9 65 29 B9 F8 A2
        :   F0 DD D5 E4 B3 02 FF FD 8C CB EC 03 D9 91 7F FA
        :   6B 8D 24 91 17 14 6C 58 FD 89 A1 35 A9 6E CE 67
        :   0C 58 A5 59 62 8A 68 01 2D 4C 9A 43 E5 FF FB 42
        :   7F 0D F3 0D A7 45 BB 0E 4E B3 34 71 6B B8 21 75
        :   10 46 7B 8B 58 E1 BE 54 B3 0C 85 75 22 65 4C BB
        :   57 87 81 3C B1 B2 FB 95 7A 72 5F 51 61 21 87 B5
        :   8B B8 33 53 F8 E5 6A A6 AF 89 C9 4E DD 65 6D 92
        :   BF 66 91 30 22 FC E8 F8 57 30 61 7D 8A 16 6B BB
        :   24 51 55 59 91 5C C4 72 43 59 50 BC E8 C2 52 19
        :   FE F5 07 A6 52 7A 6A 39 20 C2 DE AD A7 73 29 02
        :   E2 D2 AE 8C 60 BD A5 18 A6 5D A6 6F 93 58 1B 7E
        :   37 04 2A 10 46 46 75 AB 32 6F 60 3B E1 4E 5B 55
        :   25 63 4A DE B9 56 51 2A 84 FD B8 09 AF 2F 37 D2
        :   0A 9F D2 3F 67 EC A7 74 1B 49 D5 B7 D7 4F 94 7A
        :   8B 6D AC 68 88 8B 3E 38 02 99 4C 98 CC CB 68 C2
        :   A0 E7 92 7F 15 E5 E7 BE 14 00 98 3E 22 A4 C9 EF
        :   2B AB DF 24 45 25 AD DE 3B 4D 91 E7 78 E2 1E 2B
        :   D3 2C 52 21 F5 43 3F 28 39 01 28 EA BF 3B 18 1B
        :   85 C2 C1 EF D7 FA E9 46 39 9B B8 FE 21 DE 2A ED
        :   0A 3E 72 CA B3 4D 30 55 09 54 7D F3 3A 45 D1 7D
        :   01 3A 3A C0 8F 0D B6 9D BF F0 BA E5 A4 B9 C1 5E
        :   E8 BD C4 E2 52 5B 49 E6 86 5A EB 75 20 51 93 5F
        :   8E 2F B5 DA EA 91 2D 11 86 71 10 2B 8F B4 67 5C
        :   39 37 B3 89 A7 C7 06 8C B6 97 B3 79 8F FF E9 D6
        :   E6 41 56 49 3B B8 08 D0 67 A3 EA C4 18 A7 BF 58
```

```
          :   19 E2 5A 74 0C 83 49 91 4E 50 08 34 0C 38 1A 07
          :   D8 70 B1 4B 9B E9 93 9D 30 6B C7 02 D4 6A 58 21
          :   6D F9 32 D2 BC B4 5A 3F 18 1B D8 4F 4B 9B A9 92
          :   F3 A5 F1 2A 5D B6 15 AE 05 98 A9 C4 32 CF 5C 00
          :   95 18 7A 84 9A 93 CA 0D 2D 7D EE DB 2D B1 CE D5
          :   3C 0D 4A E5 D2 B3 7E B2 5E 07 99 24 52 E0 18 8E
          :   2D 72 68 2F 46 F0 16 7F 0F 6A 68 02 43 EF F2 C4
          :   B5 FC F0 23 58 52 F9 9D EF A4 D5 35 A4 79 17 6F
          :   A3 A2 25 87 51 15 81 B3 64 9F E4 10 E3 02 EC 1B
          :   90 61 CB 53 5C 49 2A E5 7A C1 26 CB 49 EA 47 DE
          :   12 A9 09 7C 5F 8A 86 9D 84 D4 AA E9 03 F0 BC 6F
          :   53 A6 05 14 CE BE 02 FD C9 A2 04 A6 BE 2A 66 56
          :   7D 58 9F CB AC A3 65 65 A3 E0 2D 10 B1 69 20 EA
          :   26 B0 5C 50 BE 80 5D 06 EC E3 C9 EE 7D B7 EB 20
          :   7C 33 C1 D4 AC 92 CA 29 4F 1A 0F 8F AC 83 9F EA
          :   03 89 FA E4 33 84 A2 DA A7 A4 C1 F5 E9 2D 62 2F
          :   8E B3 7A C0 C5 5E B9 DF 8D 97 13 DE 03 A0 EF 8B
          :   2D 93 06 C8 53 0F 60 7D 89 8B A8 8A DB 6A D9 82
          :   BE CD 3B 05 CA C3 2B 7D FE 76 7C 82 6E 5E 8B DF
          :   40 12 0F 1A 49 E8 4C D1 1C 4C 09 B0 7C 27 59 1A
          :   60 32 B3 72 A7 BD 46 8B 09 EC E0 7E E6 EC A6 E9
          :   45 59 F1 48 43 DD 9A 97 93 1E 0C 06 D1 05 AA 22
          :   37 E6 7A 3C 12 8D 33 AB 61 DC 47 98 6F DA FB 79
          :   7E 60 CB 44 45 3B 4D FB CF 7A FD FE 02 F6 86 C7
          :   20 FD 65 D5 38 18 27 BE 62 F3 32 2A CB C3 72 13
          :   04 43 D6 69 0E 3A E3 B1 7F 26 3F 93 45 99 8C A2
          :   63 17 F7 57 B8 6D 0A D4 15 31 B1 14 F5 D5 7F A8
          :   2E 50 23 E5 17 62 27 D0 87 F7 65 E1 42 1C EF 31
          :   BE 9C 31 58 66 83 80 17 BB 2A A5 89 55 BF 52 CE
          :   6E
  1247   13:       SEQUENCE {
  1249   11:        OBJECT IDENTIFIER
          :         hkdfWithSha256 (1 2 840 113549 1 9 16 3 28)
          :         }
  1262    1:       INTEGER 32
  1265   11:       SEQUENCE {
  1267    9:        OBJECT IDENTIFIER
          :         aes256-wrap (2 16 840 1 101 3 4 1 45)
          :         }
  1278   40:       OCTET STRING
          :   5D 13 AE 00 D9 14 A2 91 58 A8 EF 32 B2 3A E5 F2
          :   7B 8C 29 33 87 E3 BE 65 FC 3D FC 19 B9 9D 8F DC
          :   55 25 27 FD 42 15 4B 37
          :        }
          :       }
          :      }
  1320   58:    SEQUENCE {
  1322    9:     OBJECT IDENTIFIER data (1 2 840 113549 1 7 1)
```

```
1333   30:     SEQUENCE {
1335    9:      OBJECT IDENTIFIER
          :       aes256-GCM (2 16 840 1 101 3 4 1 46)
1346   17:      SEQUENCE {
1348   12:       OCTET STRING 86 BC 07 FB C3 DC 4A 59 9D 90 F3 E8
1362    1:       INTEGER 16
          :        }
          :       }
1365   13:     [0] 20 F9 4F 06 1F 7B 05 F9 5B 32 57 FF C0
          :      }
1380   16:    OCTET STRING
          :   5F 2A 60 63 AF 2C F0 5F E1 E2 97 A2 8B A7 AB 10
          :     }
          :    }
          :   }
```

B.2. Recipient CMS Processing

Bob’s id-MLKEM768-ECDH-P256-SHA3-256 private key:

```
-----BEGIN PRIVATE KEY-----
MIGEAgEAMAoGCCsGAQUFBwY7BHOImiTrgVkiRNnS3EmMdxHUrh+EHeflRSSQaMrG
7NAvf0DDSVt58hvzJz/RuCOE4/8REOs/DZVr5gWO2jOyldClMDECAQEEIM/ctKyh
CrQyBNhBbpwS5ZkEO1mklf14j5NRtwj3obB+oAoGCCqGSM49AwEH
-----END PRIVATE KEY-----
```

Bob decapsulates the ciphertext in the KEMRecipientInfo to get the MLKEM768-ECDH-P256-SHA3-256 shared secret, encodes the CMSORIforKEMOtherInfo, derives the key-encryption key from the shared secret and the DER-encoded CMSORIforKEMOtherInfo using HKDF with SHA-256, uses AES-256-KEYWRAP to decrypt the content-encryption key with the key-encryption key, and decrypts the encrypted contents with the content-encryption key, revealing the plaintext content:

(Artwork only available as TEST-VECTORS: see https://www.ietf.org/archive/id/draft-ietf-lamps-cms-composite-kem-01.html)

Appendix C. Acknowledgments

This document borrows heavily from [RFC9690] and [RFC9936]. Thanks go to the authors of those documents. "Copying always makes things easier and less error prone" - RFC8411.

Authors’ Addresses

Daniel Van Geest
CryptoNext Security
16, Boulevard Saint-Germain
75007 Paris
France
Email: daniel.vangeest@cryptonext-security.com

Mike Ounsworth
Entrust Limited
2500 Solandt Road Suite 100
Ottawa, Ontario K2K 3G5
Canada
Email: mike.ounsworth@entrust.com

John Gray
Entrust Limited
2500 Solandt Road Suite 100
Ottawa, Ontario K2K 3G5
Canada
Email: john.gray@entrust.com

Jan Klaussner
Bundesdruckerei GmbH
Kommandantenstr. 18
10969 Berlin
Germany
Email: jan.klaussner@bdr.de