Workgroup: LAMPS
Internet-Draft: draft-ietf-lamps-cms-kyber-00
Published: 13 March 2023
Intended Status: Standards Track
Expires: 14 September 2023
Authors: J. Prat, CryptoNext Security
M. Ounsworth, Entrust Limited

Use of KYBER in the Cryptographic Message Syntax (CMS)

Abstract

This document describes the conventions for using a Key Encapsulation Mechanism algorithm (KEM) within the Cryptographic Message Syntax (CMS). The CMS specifies the envelopped-data content type, which consists of an encrypted content and encrypted content-encryption keys for one or more recipients. The mechanism proposed here can rely on either post-quantum KEMs, hybrid KEMs or classical KEMs.

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/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 14 September 2023.

Copyright Notice

Copyright (c) 2023 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in

Section 4.e of the Trust Legal Provisions and are provided without
warranty as described in the Revised BSD License.

Table of Contents

1. Changes in this version

The following changes were made in this version:

*Use of KEMRecipientInfo to communicate algorithm info;

*Editorial changes.

2. Introduction

In recent years, there has been a substantial amount of research on
quantum computers -- machines that exploit quantum mechanical

phenomena to solve mathematical problems that are difficult or intractable for conventional computers. If large-scale quantum computers are ever built, they will be able to break many of the public-key cryptosystems currently in use. This would seriously compromise the confidentiality and integrity of digital communications on the Internet and elsewhere. Under such a threat model, the current key encapsulation mechanisms would be vulnerable.

Post-quantum key encapsulation mechanisms (PQ-KEM) are being developed in order to provide secure key establishment against an adversary with access to a quantum computer.

As the National Institute of Standards and Technology (NIST) is still in the process of selecting the new post-quantum cryptographic algorithms that are secure against both quantum and classical computers, the purpose of this document is to propose a generic "algorithm-agnostic" solution to protect in confidentiality the CMS envelopped-data content against the quantum threat : the KEM-TRANS mechanism.

Although this mechanism could thus be used with any key encapsulation mechanism, including post-quantum KEMs or hybrid KEMs.

This RFC nonetheless specifically specifies the case where the algorithm PQ-KEM algorithm is Kyber.

3. 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.

The following terms are used in this document:

BER: Basic Encoding Rules (BER) as defined in [X.690].

DER: Distinguished Encoding Rules as defined in [X.690].

4. Design Rationales

The Cryptographic Message Syntax (CMS) [RFC5652] defines two levels of encryptions in the Envelopped-Data Content section:

* the Content-encryption process which protects the data using a symmetric algorithm used with a content encryption key (CEK);

* the Key-encryption process which protects this CEK using a key transport mechanism.

One of the typical use case of the CMS Envelopped-Data Content is to randomly generate a CEK, encrypt the data with a symmetric algorithm using this CEK and individually send the CEK to one or more recipients protected by asymmetric cryptography in a RecipientInfo object.

To achieve this scenario with KEM primitives, it is necessary to define a new key transport mechanism that will fulfil the following requirements:

* the Key Transport Mechanism SHALL be secure against quantum computers.

* the Key Transport Mechanism SHALL take the Content-Encryption Key (CEK) as input.

According to NIST, a KEM generates a random secret and a ciphertext from which the recipient can extract the shared secret, meaning that a KEM can not be used straightforwardly as a key transport mechanism in the CMS "multi-recipients" context. The KEM-TRANS mechanism defined in this document aims to turn a KEM into a key transport scheme allowing the sender to distribute a randomly generated key to several recipients. The KEM-TRANS Key transport mechanism described in the following section fulfils the requirements listed above and is an adaptation of the RSA-KEM algorithm previously specified in [RFC5990]. The solution is also aligned with the hybrid public key encryption scheme described in [RFC9180].

5. KEM Key Transport Mechanism (KEM-TRANS)

The KEM Key Transport Mechanism (KEM-TRANS) is a one-pass (store-and-forward) mechanism for transporting keying data to a recipient.

With this type of mechanism, a sender cryptographically encapsulates the keying data using the recipient's public key to obtain encrypted keying data. The recipient can then decapsulate the encrypted keying data using his private key to recover the plaintext keying data.

5.1. Underlying Components

The KEM-TRANS requires use of the following underlying components, which are provided to KEM-TRANS as algorithm parameters.

* KEM, a Key Encapsulation Mechanism;

* KDF, a Key Derivation Function, which derives keying data of a specified length from a shared secret value;

* WRAP, a symmetric key-wrapping scheme, which encrypts keying Data using a key-encrypting key (KEK).

5.1.1. KEM

A KEM is a cryptographic algorithm consisting of three functions :

* a key generation function **KEM.keygen** taking as input a security level and returning a key pair (private key and the associated public key) for this security level.

* an encapsulation function **KEM.encaps** taking a public key as input and returning a random session key and a ciphertext that is an encapsulation of the session key.

* a decaspulation function **KEM.decaps** taking as input a private key and a ciphertext and returning a session key.

5.1.2. KDF

A key derivation function (KDF) is a cryptographic function that deterministically derives one or more secret keys from a secret value using a pseudorandom function. KDFs can be used to stretch keys into longer keys or to obtain keys of a required format.

If the session key obtained from the KEM algorithm is long enough to fit into the WRAP algorithm, then the KDF could be equal to the identity function.

5.1.3. WRAP

A wrapping algorithm is a symmetric algorithm protecting data in confidentiality and integrity. It is especially designed to transport key material. the WRAP algorithm consists of two functions :

* a wrapping function **Wrap** taking a wrapping key and a plaintext key as input and returning a wrapped key.

* a decaspulation function **Unwrap** taking as input a wrapping key and a wraped key and returning the plaintext key.

In the following, *kekLen* denotes the length in bytes of the wrapping key for the underlying symmetric key-wrapping scheme.

In this scheme, the length of the keying data to be transported MUST be among the lengths supported by the underlying symmetric key-wrapping scheme.

5.2. Recipient's Key Generation and Distribution

The KEM-TRANS described in the next sections assumes that the recipient has previously generated a key pair (*recipPrivKey* and *recipPubKey*) and has distributed this public key to the sender.

The protocols and mechanisms by which the key pair is securely generated and the public key is securely distributed are out of the scope of this document.

5.3. Sender's Operations

This process assumes that the following algorithm parameters have been selected:

- *KEM*: a key encapsulation mechanism, as defined above.

- *KDF*: a key derivation function, as defined above.

- *Wrap*: a symmetric key-wrapping algorithm, as defined above.

- *kekLen*: the length in bits of the key required for the Wrap algorithm.

This process assumes that the following input data has been provided:

- *recipPubKey*: the recipient's public key.

- *K*: the keying data to be transported, assumed to be a length that is compatible with the chosen Wrap algorithm.

This process outputs:

- *EK*: the encrypted keying data, from which the recipient will be able to retrieve *K*.

The sender performs the following operations:

1. Generate a shared secret *SS* and the associated ciphertext *CT* using the KEM encaspulation function and the recipient's public key *recipPubKey*:

   ```
   (SS, CT) = KEM.encaps(recipPubKey)
   ```

2. Derive a key-encrypting key *KEK* of length *kekLen* bytes from the shared secret *SS* using the underlying key derivation function:

   ```
   KEK = KDF(SS, kekLen)
   ```

3. Wrap the keying data *K* with the key-encrypting key *KEK* using the underlying key-wrapping scheme to obtain wrapped keying data *WK* of length *wrappedKekLen*:

   ```
   WK = Wrap(KEK, K)
   ```

4. Concatenate the wrapped keying data *WK* of length *wrappedKekLen* and the ciphertext *CT* to obtain the encrypted keying data *EK*:

   ```
   EK = (WK || CT)
   ```

5. Output the encrypted keying data *EK*.

5.4. Recipient's Operations

This process assumes that the following algorithm parameters have been communicated from the sender:

* *KEM*: a key encapsulation mechanism, as defined above.

* *KDF*: a key derivation function, as defined above.

* *Wrap*: a symmetric key-wrapping algorithm, as defined above.

* *kekLen*: the length in bits of the key required for the Wrap algorithm.

This process assumes that the following input data has been provided:

* *recipPrivKey*: the recipient's private key.

* *EK*: the encrypted keying data.

This process outputs:

* *K*: the keying data to be transported.

The recipient performs the following operations:

1. Separate the encrypted keying data *EK* into wrapped keying data *WK* of length *wrappedKekLen* and a ciphertext *CT* :

   ```
   (WK || CT) = EK
   ```

2. Decapsulate the ciphertext *CT* using the KEM decaspulation function and the recipient's private key to retrieve the shared secret *SS*:

   ```
   SS = KEM.decaps(recipPrivKey, CT)
   ```

If the decapsulation operation outputs an error, output "decryption error", and stop.

3. Derive a key-encrypting key *KEK* of length *kekLen* bytes from the shared secret *SS* using the underlying key derivation function:

```
KEK = KDF(SS, kekLen)
```

4. Unwrap the wrapped keying data *WK* with the key-encrypting key *KEK* using the underlying key-wrapping scheme to recover the keying data *K*:

```
K = Unwrap(KEK, WK)
```

If the unwrapping operation outputs an error, output "decryption error", and stop.

5. Output the keying data *K*.

6. Use in CMS

The KEM Key Transport Mechanism MAY be employed for one or more recipients in the CMS envelopped-data content type (Section 6 of [RFC5652]), where the keying data *K* processed by the mechanism is the CMS content-encryption key (*CEK*).

6.1. RecipientInfo Conventions

When the KEM Key Transport Mechanism is employed for a recipient, the RecipientInfo alternative for that recipient MUST be KEMRecipientInfo as defined in draft-ietf-lamps-cms-kemri.

6.2. Certificate Conventions

The conventions specified in this section augment [RFC5280].

6.2.1. Key Usage Extension

The intended application for the key MAY be indicated in the key usage certificate extension (see [RFC5280], Section 4.2.1.3). If the keyUsage extension is present in a certificate that conveys a public key with the id-kem object identifier as discussed above, then the key usage extension MUST contain only the value *keyEncipherment*

digitalSignature, *nonRepudiation*, *dataEncipherment*, *keyAgreement*, *keyCertSign*, *cRLSign*, *encipherOnly* and *decipherOnly* SHOULD NOT be present.

A key intended to be employed only with the KEM-TRANS SHOULD NOT also be employed for data encryption. Good cryptographic practice employs a given 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.

6.2.2. Subject Public Key Info

EDITOR'S NOTE' - TODO
Update this section according to the future evolution of draft-ietf-lamp

If the recipient wishes only to employ the KEM-TRANS with a given public key, the recipient MUST identify the public key in the certificate using the *id-kem* object identifier.

EDITOR'S NOTE' - TODO
Clarify that id-kem refers to the KEM algo (kyber512, kyber768 or kyber1
while KEM-TRANS refers to the KEM Transport mechanism

6.3. SMIME Capabilities Attribute Conventions

[RFC8551], Section 2.5.2 defines the SMIMECapabilities signed attribute (defined as a SEQUENCE of SMIMECapability SEQUENCEs) to be used to specify a partial list of algorithms that the software announcing the SMIMECapabilities can support. When constructing a signedData object, compliant software MAY include the SMIMECapabilities signed attribute announcing that it supports the KEM Key Transport Mechanism.

The SMIMECapability SEQUENCE representing the KEM Key Transport Mechanism MUST include the id-kem-trans object identifier in the capabilityID field and MUST include a GenericKemTransParameters value in the parameters field identifying the components with which the mechanism is to be employed.

The DER encoding of a SMIMECapability SEQUENCE is the same as the DER encoding of an AlgorithmIdentifier. Example DER encodings for typical sets of components are given in Appendix A.

7. Security Considerations

EDITOR'S NOTE' - TODO
section to be completed

8. IANA Considerations

Within the CMS, algorithms are identified by object identifiers (OIDs). With one exception, all of the OIDs used in this document were assigned in other IETF documents, in ISO/IEC standards documents, by the National Institute of Standards and Technology

(NIST). The two exceptions are the ASN.1 module's identifier and id-kem-transport that are both assigned in this document.

9. Acknowledgements

This document incorporates contributions and comments from a large group of experts. The Editors would especially like to acknowledge the expertise and tireless dedication of the following people, who attended many long meetings and generated millions of bytes of electronic mail and VOIP traffic over the past year in pursuit of this document:

EDITOR'S NOTE' - TODO
section to be completed

We are grateful to all, including any contributors who may have been inadvertently omitted from this list.

This document borrows text from similar documents, including those referenced below. Thanks go to the authors of those documents. "Copying always makes things easier and less error prone" - [RFC8411].

10. Annex A : ASN.1 Syntax

The syntax for the scheme is given in Appendix A.1.

The syntax for selected underlying components including those mentioned above is given in Appendix A.2.

The following object identifier prefixes are used in the definitions below:

```
nistAlgorithm OID ::= {
   joint-iso-itu-t(2) country(16) us(840) organization(1)
   gov(101) csor(3) nistAlgorithm(4)
}

smimeAlgorithm OID ::= { iso(1) member-body(2)
   us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) alg(3)
}
```

10.1. Annex A1 : KEM-TRANS Key Transport Mechanism

The object identifier for the KEM Key Transport Mechanism is id-kem-trans, which is defined in this document as:

```
id-kem-trans OID ::= { TBD }
```

When id-kem-trans is used in an AlgorithmIdentifier, the parameters MUST employ the GenericKemTransParameters syntax. The syntax for GenericKemTransParameters is as follows:

```
GenericKemTransParameters ::= {
    kem  KeyEncapsulationMechanism,
    kdf  KeyDerivationFunction,
    wrap KeyWrappingMechanism
}
```

The fields of type GenericKemTransParameters have the following meanings:

* kem identifies the underlying key encapsulation mechanism (KEM). This can be Kyber.

* kdf identifies the underlying key derivation function (KDF). This can be any KDF from [SP-800-56C-r2]. kdf can be equal to *null* if the key encaspulation mechanism outputs a shared secret *SS* of size *kekLen*.

* wrap identifies the underlying key wrapping mechanism (WRAP). This can be any wrapping mechanism from [RFC5649].

10.2. Annex A2 : Underlying Components

10.2.1. Key Encapsulation Mechanisms

KEM-TRANS can support any NIST KEM, including the post-quantum KEM Kyber. This RFC only specifies the use of Kyber.

The object identifier for KEM depends on the security level (128 bits, 192 bits or 256 bits)

```
id-kyber512 OID ::= { nistAlgorithm TBD }
id-kyber768 OID ::= { nistAlgorithm TBD }
id-kyber1024 OID ::= { nistAlgorithm TBD }
```

These object identifiers have no associated parameters.

```
kyber512 ALGORITHM ::= { OID id-kyber512 }
kyber768 ALGORITHM ::= { OID id-kyber768 }
kyber1024 ALGORITHM ::= { OID id-kyber1024 }
```

10.2.2. Key Derivation Functions

KEM-TRANS can support any KDF from [RFC5869]. This RFC only specifies the use of HKDF. The KDF can be bypassed if the key encaspulation mechanism outputs a shared secret *SS* of size *kekLen*. kdf is then equal to *null*.

The object identifier for KDF depends on the security level (128 bits, 192 bits or 256 bits)

```
  id-alg-hkdf-with-sha256 OID ::= {
           smimeAlgorithm id-alg-hkdf-with-sha256(28)
      }

  id-alg-hkdf-with-sha384 OID ::= {
           smimeAlgorithm id-alg-hkdf-with-sha384(29)
      }

  id-alg-hkdf-with-sha512 OID ::= {
           smimeAlgorithm id-alg-hkdf-with-sha512(30)
      }
```

EDITOR'S NOTE' - TO BE DISCUSSED :
Is there a RFC defining KDF with SHA3?
Should we extend the compatible KDFs to [SP-800-56C-r2]?

10.2.3. Key Wrapping Schemes

KEM-TRANS can support any wrapping mechanism from [RFC5649]. This RFC only specifies the use of aes256-Wrap.

The object identifiers for the AES Key Wrap depend on the size of the key-encrypting key. There are three object identifiers (see [RFC5649]):

```
  id-aes128-Wrap OID ::= { nistAlgorithm aes(1) aes128-Wrap(5) }
  id-aes192-Wrap OID ::= { nistAlgorithm aes(1) aes192-Wrap(25) }
  id-aes256-Wrap OID ::= { nistAlgorithm aes(1) aes256-Wrap(45) }
```

These object identifiers have no associated parameters.

```
  aes128-Wrap ALGORITHM ::= { OID id-aes128-Wrap }
  aes192-Wrap ALGORITHM ::= { OID id-aes192-Wrap }
  aes256-Wrap ALGORITHM ::= { OID id-aes256-Wrap }
```

10.2.4. Appendix A : Examples

EDITOR'S NOTE' - TODO
section to be completed

11. References

11.1. 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/info/rfc2119>.

[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/info/rfc5280>.

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

[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/info/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/info/rfc8551>.

[SP-800-56C-r2] NIST, "Recommendation for Key-Derivation Methods in Key-Establishment Schemes", 2020.

[X.690] ASC, "Information technology - ASN.1 encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", 2007.

[X9.44] ASC, "American National Standard X9.44: Public Key Cryptography for the Financial Services Industry -- Key Establishment Using Integer Factorization Cryptography", 2007.

11.2. Informative References

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, DOI 10.17487/RFC2986, November 2000, <https://www.rfc-editor.org/info/rfc2986>.

[RFC5649] Housley, R. and M. Dworkin, "Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm", RFC 5649, DOI 10.17487/RFC5649, September 2009, <https://www.rfc-editor.org/info/rfc5649>.

[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/info/rfc5869>.

[RFC5990] Randall, J., Kaliski, B., Brainard, J., and S. Turner, "Use of the RSA-KEM Key Transport Algorithm in the Cryptographic Message Syntax (CMS)", RFC 5990, DOI 10.17487/RFC5990, September 2010, <https://www.rfc-editor.org/info/rfc5990>.

[RFC8411] Schaad, J. and R. Andrews, "IANA Registration for the Cryptographic Algorithm Object Identifier Range", RFC 8411, DOI 10.17487/RFC8411, August 2018, <https://www.rfc-editor.org/info/rfc8411>.

[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/info/rfc9180>.

[SP-800-108] NIST, "Recommendation for Key Derivation Using Pseudorandom Functions", 2009.

Authors' Addresses

Julien Prat
CryptoNext Security

Email: julien.prat@cryptonext-security.com

Mike Ounsworth
Entrust Limited

Email: mike.ounsworth@entrust.com