INTERNET-DRAFT   R. Housley
Intended Status: Proposed Standard   Vigil Security
Expires: 14 June 2019   14 December 2018

        Using Pre-Shared Key (PSK) in the Cryptographic Message Syntax (CMS)
                <draft-ietf-lamps-cms-mix-with-psk-02.txt>

Abstract

   The invention of a large-scale quantum computer would pose a serious
   challenge for the cryptographic algorithms that are widely deployed
   today. The Cryptographic Message Syntax (CMS) supports key transport
   and key agreement algorithms that could be broken by the invention of
   such a quantum computer. By storing communications that are
   protected with the CMS today, someone could decrypt them in the
   future when a large-scale quantum computer becomes available. Once
   quantum-secure key management algorithms are available, the CMS will
   be extended to support the new algorithms, if the existing syntax
   does not accommodate them. In the near-term, this document describes
   a mechanism to protect today's communication from the future
   invention of a large-scale quantum computer by mixing the output of
   key transport and key agreement algorithms with a pre-shared key.

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 http://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."

Table of Contents

[1](1). Introduction

   The invention of a large-scale quantum computer would pose a serious
   challenge for the cryptographic algorithms that are widely deployed
   today. It is an open question whether or not it is feasible to build
   a large-scale quantum computer, and if so, when that might happen.
   However, if such a quantum computer is invented, many of the
   cryptographic algorithms and the security protocols that use them
   would become vulnerable.

   The Cryptographic Message Syntax (CMS) [[RFC5652](RFC5652)][RFC5803] supports
   key transport and key agreement algorithms that could be broken by
   the invention of a large-scale quantum computer [[C2PQ](C2PQ)]. These

algorithms include RSA [RFC8017], Diffie-Hellman [RFC2631], and
Elliptic Curve Diffie-Hellman [RFC5753]. As a result, an adversary
that stores CMS-protected communications today, could decrypt those
communications in the future when a large-scale quantum computer
becomes available.

Once quantum-secure key management algorithms are available, the CMS
will be extended to support them, if the existing syntax does not
already accommodate the new algorithms.

In the near-term, this document describes a mechanism to protect
today's communication from the future invention of a large-scale
quantum computer by mixing the output of existing key transport and
key agreement algorithms with a pre-shared key (PSK). Secure
communication can be achieved today by mixing a strong PSK with the
output of an existing key transport algorithm, like RSA [RFC8017], or
an existing key agreement algorithm, like Diffie-Hellman [RFC2631] or
Elliptic Curve Diffie-Hellman [RFC5753]. A security solution that is
believed to be quantum resistant can be achieved by using a PSK with
sufficient entropy along with a quantum resistant key derivation
function (KDF), like HKDF [RFC5869], and a quantum resistant
encryption algorithm, like 256-bit AES [AES]. In this way, today's
CMS-protected communication can be invulnerable to an attacker with a
large-scale quantum computer.

In addition, there may be other reasons for including a strong PSK
besides protection against the future invention of a large-scale
quantum computer. For example, there is always the possibility of a
cryptoanalytic breakthrough on one or more of the classic public-key
algorithm, and there are longstanding concerns about undisclosed
trapdoors in Diffie-Hellamn parameters. Inclusion of a strong PSK as
part of the overall key management offer additional protection
against these concerns.

Note that the CMS also supports key management techniques based on
symmetric key-encryption keys and passwords, but they are not
discussed in this document because they are already quantum
resistant. The symmetric key-encryption key technique is quantum
resistant when used with an adequate key size. The password
technique is quantum resistant when used with a quantum-resistant key
derivation function and a sufficiently large password.

1.1. 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.

1.2. ASN.1

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

1.3. Version Numbers

The major data structures include a version number as the first item in the data structure. The version number is intended to avoid ASN.1 decode errors. Some implementations do not check the version number prior to attempting a decode, and then if a decode error occurs, the version number is checked as part of the error handling routine. This is a reasonable approach; it places error processing outside of the fast path. This approach is also forgiving when an incorrect version number is used by the sender.

Whenever the structure is updated, a higher version number will be assigned. However, to ensure maximum interoperability, the higher version number is only used when the new syntax feature is employed. That is, the lowest version number that supports the generated syntax is used.

2. Overview

The CMS enveloped-data content type [RFC5652] and the CMS authenticated-enveloped-data content type [RFC5083] support both key transport and key agreement public-key algorithms to establish the key used to encrypt the content. No restrictions are imposed on the

key transport or key agreement public-key algorithms, which means
that any key transport or key agreement algorithm can be used,
including algorithms that are specified in the future. In both
cases, the sender randomly generates the content-encryption key, and
then all recipients obtain that key. All recipients use the sender-
generated symmetric content-encryption key for decryption.

This specification defines two quantum-resistant ways to establish a
symmetric key-encryption key, which is used to encrypt the sender-
generated content-encryption key. In both cases, the PSK is used as
one of the inputs to a key-derivation function to create a quantum-
resistant key-encryption key. The PSK MUST be distributed to the
sender and all of the recipients by some out-of-band means that does
not make it vulnerable to the future invention of a large-scale
quantum computer, and an identifier MUST be assigned to the PSK.

   The content-encryption key or content-authenticated-encryption key is
   quantum-resistant, and the sender establishes it using these steps:

   When using a key transport algorithm:

   1. The content-encryption key or the content-authenticated-
      encryption key, called CEK, is generated at random.

   2. The key-derivation key, called KDK, is generated at random.

   3. For each recipient, the KDK is encrypted in the recipient's
      public key, then the key derivation function (KDF) is used to
      mix the pre-shared key (PSK) and the KDK to produce the key-
      encryption key, called KEK.

   4. The KEK is used to encrypt the CEK.

   When using a key agreement algorithm:

   1. The content-encryption key or the content-authenticated-
      encryption key, called CEK, is generated at random.

2. For each recipient, a pairwise key-encryption key, called KEK1,
        is established using the recipient's public key and the
        sender's private key.

     3. For each recipient, the key derivation function (KDF) is used
        to mix the pre-shared key (PSK) and the pairwise KEK1, and the
        result is called KEK2.

     4. For each recipient, the pairwise KEK2 is used to encrypt the
        CEK.

   As specified in Section 6.2.5 of [RFC5652], recipient information for
   additional key management techniques are represented in the
   OtherRecipientInfo type. Two key management techniques are specified
   in this document, and they are each identified by a unique ASN.1
   object identifier.

   The first key management technique, called keyTransPSK, see
   Section 3, uses a key transport algorithm to transfer the key-
   derivation key from the sender to the recipient, and then the key-
   derivation key is mixed with the PSK using a KDF. The output of the
   KDF is the key-encryption key, which is used for the encryption of
   the content-encryption key or content-authenticated-encryption key.

   The second key management technique, called keyAgreePSK, see
   Section 4, uses a key agreement algorithm to establish a pairwise

---

   key-encryption key, which is then mixed with the PSK using a KDF to
   produce a second pairwise key-encryption key, which is then used to
   encrypt the content-encryption key or content-authenticated-
   encryption key.

3. KeyTransPSKRecipientInfo

   Per-recipient information using keyTransPSK is represented in the
   KeyTransPSKRecipientInfo type, which is indicated by the id-ori-
   keyTransPSK object identifier. Each instance of
   KeyTransPSKRecipientInfo establishes the content-encryption key or
   content-authenticated-encryption key for one or more recipients that
   have access to the same PSK.

   The id-ori-keyTransPSK object identifier is:

```
         id-ori OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
           rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) TBD1 }

         id-ori-keyTransPSK OBJECT IDENTIFIER ::= { id-ori 1 }

      The KeyTransPSKRecipientInfo type is:

         KeyTransPSKRecipientInfo ::= SEQUENCE {
           version CMSVersion,  -- always set to 0
           pskid PreSharedKeyIdentifier,
           kdfAlgorithm KeyDerivationAlgorithmIdentifier,
           keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
           ktris KeyTransRecipientInfos,
           encryptedKey EncryptedKey }

         PreSharedKeyIdentifier ::= OCTET STRING

         KeyTransRecipientInfos ::= SEQUENCE OF KeyTransRecipientInfo

      The fields of the KeyTransPSKRecipientInfo type have the following
      meanings:

         version is the syntax version number.  The version MUST be 0.  The
         CMSVersion type is described in Section 10.2.5 of [RFC5652].

         pskid is the identifier of the PSK used by the sender.  The
         identifier is an OCTET STRING, and it need not be human readable.

         kdfAlgorithm identifies the key-derivation algorithm, and any
         associated parameters, used by the sender to mix the key-
         derivation key and the PSK to generate the key-encryption key.
         The KeyDerivationAlgorithmIdentifier is described in Section
```

```
         10.1.6 of [RFC5652].

         keyEncryptionAlgorithm identifies a key-encryption algorithm used
         to encrypt the content-encryption key.  The
         KeyEncryptionAlgorithmIdentifier is described in Section 10.1.3 of
            [RFC5652].

         ktris contains one KeyTransRecipientInfo type for each recipient;
```

it uses a key transport algorithm to establish the key-derivation
key. KeyTransRecipientInfo is described in Section 6.2.1 of
[RFC5652].

encryptedKey is the result of encrypting the content-encryption
key or the content-authenticated-encryption key with the key-
encryption key. EncryptedKey is an OCTET STRING.

4. KeyAgreePSKRecipientInfo

Per-recipient information using keyAgreePSK is represented in the
KeyAgreePSKRecipientInfo type, which is indicated by the id-ori-
keyAgreePSK object identifier. Each instance of
KeyAgreePSKRecipientInfo establishes the content-encryption key or
content-authenticated-encryption key for one or more recipients that
have access to the same PSK.

The id-ori-keyAgreePSK object identifier is:

    id-ori-keyAgreePSK OBJECT IDENTIFIER ::= { id-ori 2 }

The KeyAgreePSKRecipientInfo type is:

    KeyAgreePSKRecipientInfo ::= SEQUENCE {
      version CMSVersion, -- always set to 0
      pskid PreSharedKeyIdentifier,
      originator [0] EXPLICIT OriginatorIdentifierOrKey,
      ukm [1] EXPLICIT UserKeyingMaterial OPTIONAL,
      kdfAlgorithm KeyDerivationAlgorithmIdentifier,
      keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
      recipientEncryptedKeys RecipientEncryptedKeys }

The fields of the KeyAgreePSKRecipientInfo type have the following
meanings:

version is the syntax version number. The version MUST be 0. The
CMSVersion type is described in Section 10.2.5 of [RFC5652].

pskid is the identifier of the PSK used by the sender. The
identifier is an OCTET STRING, and it need not be human readable.

originator is a CHOICE with three alternatives specifying the

sender's key agreement public key. Implementations MUST support
all three alternatives for specifying the sender's public key.
The sender uses their own private key and the recipient's public
key to generate a pairwise key-encryption key. A key derivation
function (KDF) is used to mix the PSK and the pairwise key-
encryption key to produce a second key-encryption key. The
OriginatorIdentifierOrKey type is described in [Section 6.2.2 of
[RFC5652]](#).

ukm is optional. With some key agreement algorithms, the sender
provides a User Keying Material (UKM) to ensure that a different
key is generated each time the same two parties generate a
pairwise key. Implementations MUST accept a
KeyAgreePSKRecipientInfo SEQUENCE that includes a ukm field.
Implementations that do not support key agreement algorithms that
make use of UKMs MUST gracefully handle the presence of UKMs. The
UserKeyingMaterial type is described in [Section 10.2.6 of
[RFC5652]](#).

kdfAlgorithm identifies the key-derivation algorithm, and any
associated parameters, used by the sender to mix the pairwise key-
encryption key and the PSK to produce a second key-encryption key
of the same length as the first one. The
KeyDerivationAlgorithmIdentifier is described in [Section 10.1.6 of
[RFC5652]](#).

keyEncryptionAlgorithm identifies a key-encryption algorithm used
to encrypt the content-encryption key or the content-
authenticated-encryption key. The
KeyEncryptionAlgorithmIdentifier type is described in [Section
10.1.3 of [RFC5652]](#).

recipientEncryptedKeys includes a recipient identifier and
encrypted key for one or more recipients. The
KeyAgreeRecipientIdentifier is a CHOICE with two alternatives
specifying the recipient's certificate, and thereby the
recipient's public key, that was used by the sender to generate a
pairwise key-encryption key. The encryptedKey is the result of
encrypting the content-encryption key or the content-
authenticated-encryption key with the second pairwise key-
encryption key. EncryptedKey is an OCTET STRING. The
RecipientEncryptedKeys type is defined in [Section 6.2.2 of
[RFC5652]](#).

5. Key Derivation

   Many key derivation functions (KDFs) internally employ a one-way hash
   function. When this is the case, the hash function that is used is
   identified by the KeyDerivationAlgorithmIdentifier. HKDF [RFC5869]
   is one example of a KDF that make use fo a hash function.

   A KDF has several input values. This section describes the
   conventions for using the KDF to compute the key-encryption key for
   KeyTransPSKRecipientInfo and KeyAgreePSKRecipientInfo. For
   simplicity, the terminology used in the HKDF [RFC5869] specification
   is used here.

   The KDF inputs are:

      IKM is the input keying material; it is the symmetric secret input
      to the KDF. For KeyTransPSKRecipientInfo, it is the PSK
      concatenated with the key-derivation key. For
      KeyAgreePSKRecipientInfo, it is the PSK concatenated with the
      pairwise key-encryption key produced by the key agreement
      algorithm.

      salt is an optional non-secret random value. The salt is not
      used.

      L is the length of output keying material in octets; the value
      depends on the key-encryption algorithm that will be used. The
      algorithm is identified by the KeyEncryptionAlgorithmIdentifier.
      In addition, the OBJECT IDENTIFIER portion of the
      KeyEncryptionAlgorithmIdentifier is included in the next input
      value, called info.

      info is optional context and application specific information.
      The DER-encoding of CMSORIforPSKOtherInfo is used as the info
      value. Note that EXPLICIT tagging is used in the ASN.1 module
      that deines this structure. For KeyTransPSKRecipientInfo, the
      ENUMERATED value of 5 is used. For KeyAgreePSKRecipientInfo, the
      ENUMERATED value of 10 is used. CMSORIforPSKOtherInfo is defined
      by the following ASN.1 structure:

         CMSORIforPSKOtherInfo ::= SEQUENCE {
           keyMgmtAlgType   ENUMERATED {
             keyTrans   (5),
             keyAgree   (10) },
           keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
           pskLength   INTEGER (1..MAX),

```
          kdkLength                INTEGER (1..MAX) }
```

   The fields of type CMSORIforPSKOtherInfo have the following meanings:

      keyMgmtAlgType is either set to 5 or 10. For
      KeyTransPSKRecipientInfo, the ENUMERATED value of 5 is used. For
      KeyAgreePSKRecipientInfo, the ENUMERATED value of 10 is used.

      keyEncryptionAlgorithm is the KeyEncryptionAlgorithmIdentifier,
      which identifies the algorithm and provides algorithm parameters,
      if any.

      pskLength is a positive integer; it contains the length of the PSK
      in octets.

      kdkLength is a positive integer; it contains the length of the
      key-derivation key in octets. For KeyTransPSKRecipientInfo, the
      key-derivation key is generated by the sender. For
      KeyAgreePSKRecipientInfo, the key-derivation key is the pairwise
      key-encryption key produced by the key agreement algorithm.

   The KDF output is:

      OKM is the output keying material, which is exactly L octets. The
      OKM is the key-encryption key that is used to encrypt the content-
      encryption key or the content-authenticated-encryption key.

6. ASN.1 Module

   This section contains the ASN.1 module for the two key management
   techniques defined in this document. This module imports types from
   other ASN.1 modules that are defined in [RFC5911] and [RFC5912].

   CMSORIforPSK-2017
     { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
       smime(16) modules(0) id-mod-cms-ori-psk-2017(TBD0) }

   DEFINITIONS EXPLICIT TAGS ::=
   BEGIN

   -- EXPORTS All

```
   IMPORTS

   AlgorithmIdentifier{}, KEY-DERIVATION
     FROM AlgorithmInformation-2009  -- [RFC5912]
       { iso(1) identified-organization(3) dod(6) internet(1)
         security(5) mechanisms(5) pkix(7) id-mod(0)
         id-mod-algorithmInformation-02(58) }
```

```
   OTHER-RECIPIENT, OtherRecipientInfo, CMSVersion,
   KeyTransRecipientInfo, OriginatorIdentifierOrKey,
   UserKeyingMaterial, RecipientEncryptedKeys, EncryptedKey,
   KeyDerivationAlgorithmIdentifier, KeyEncryptionAlgorithmIdentifier
     FROM CryptographicMessageSyntax-2009  -- [RFC5911]
       { iso(1) member-body(2) us(840) rsadsi(113549)
         pkcs(1) pkcs-9(9) smime(16) modules(0)
         id-mod-cms-2004-02(41) } ;

   --
   -- OtherRecipientInfo Types (ori-)
   --

   SupportedOtherRecipInfo OTHER-RECIPIENT ::= {
     ori-keyTransPSK |
     ori-keyAgreePSK,
     ... }

   --
   -- Key Transport with Pre-Shared Key
   --

   ori-keyTransPSK OTHER-RECIPIENT ::= {
     KeyTransPSKRecipientInfo IDENTIFIED BY id-ori-keyTransPSK }

   id-ori OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
     rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) TBD1 }

   id-ori-keyTransPSK OBJECT IDENTIFIER ::= { id-ori 1 }

   KeyTransPSKRecipientInfo ::= SEQUENCE {
     version CMSVersion,  -- always set to 0
```

```
    pskid PreSharedKeyIdentifier,
    kdfAlgorithm KeyDerivationAlgorithmIdentifier,
    keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
    ktris KeyTransRecipientInfos,
    encryptedKey EncryptedKey }

  PreSharedKeyIdentifier ::= OCTET STRING

  KeyTransRecipientInfos ::= SEQUENCE OF KeyTransRecipientInfo

  --
  -- Key Agreement with Pre-Shared Key
  --

  ori-keyAgreePSK ORI-TYPE ::= {
    KeyAgreePSKRecipientInfo IDENTIFIED BY id-ori-keyAgreePSK }
```

```
  id-ori-keyAgreePSK OBJECT IDENTIFIER ::= { id-ori 2 }

  KeyAgreePSKRecipientInfo ::= SEQUENCE {
    version CMSVersion,  -- always set to 0
    pskid PreSharedKeyIdentifier,
    originator [0] EXPLICIT OriginatorIdentifierOrKey,
    ukm [1] EXPLICIT UserKeyingMaterial OPTIONAL,
    kdfAlgorithm KeyDerivationAlgorithmIdentifier,
    keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
    recipientEncryptedKeys RecipientEncryptedKeys }

  --
  -- Structure to provide 'info' input to the KDF
  --

  CMSORIforPSKOtherInfo ::= SEQUENCE {
    keyMgmtAlgType          ENUMERATED {
      keyTrans              (5),
      keyAgree              (10) },
    keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
    pskLength              INTEGER (1..MAX),
    kdkLength              INTEGER (1..MAX) }

  END
```

7. Security Considerations

   Implementations must protect the pre-shared key (PSK), key transport
   private key, the agreement private key, the key-derivation key, and
   the key-encryption key. Compromise of the PSK will make the
   encrypted content vulnerable to the future invention of a large-scale
   quantum computer. Compromise of the PSK and either the key transport
   private key or the agreement private key may result in the disclosure
   of all contents protected with that combination of keying material.
   Compromise of the PSK and the key-derivation key may result in
   disclosure of all contents protected with that combination of keying
   material. Compromise of the key-encryption key may result in the
   disclosure of all content-encryption keys or content-authenticated-
   encryption keys that were protected with that keying materail, which
   in turn may result in the disclosure of the content.

   A large-scale quantum computer will essentially negate the security
   provided by the key transport algorithm or the key agreement
   algorithm, which means that the attacker with a large-scale quantum
   computer can discover the key-derivation key. In addition a large-
   scale quantum computer effectively cuts the security provided by a
   symmetric key algorithm in half. Therefore, the PSK needs at least
   256 bits of entropy to provide 128 bits of security. To match that

   same level of security, the key derivation function needs to be
   quantum-resistant and produce a key-encryption key that is at least
   256 bits in length. Similarly, the content-encryption key or
   content-authenticated-encryption key needs to be at least 256 bits in
   length.

   When using a PSK with a key transport or a key agreement algorithm, a
   key-encryption key is produced to encrypt the content-encryption key
   or content-authenticated-encryption key. If the key-encryption
   algorithm is different than the algorithm used to protect the
   content, then the effective security is determined by the weaker of
   the two algorithms. If, for example, content is encrypted with
   256-bit AES, and the key is wrapped with 128-bit AES, then at most
   128 bits of protection is provided. Implementers must ensure that
   the key-encryption algorithm is as strong or stronger than the
   content-encryption algorithm or content-authenticated-encryption
   algorithm.

Implementers should not mix quantum-resistant key management
algorithms with their non-quantum-resistant counterparts. For
example, the same content should not be protected with
KeyTransRecipientInfo and KeyTransPSKRecipientInfo. Likewise, the
same content should not be protected with KeyAgreeRecipientInfo and
KeyAgreePSKRecipientInfo. Doing so would make the content vulnerable
to the future invention of a large-scale quantum computer.

Implementers should not send the same content in different messages,
one using a quantum-resistant key management algorithm and the other
using a non-quantum-resistant key management algorithm, even if the
content-encryption key is generated independently. Doing so may
allow an eavesdropper to correlate the messages, making the content
vulnerable to the future invention of a large-scale quantum computer.

Sound cryptographic key hygiene is to use a key for one and only one
purpose. Use of the recipient's public key for both the traditional
CMS and the PSK-mixing variation specified in this document would be
a violations of this principle; however, there is no known way for an
attacker to take advantage of this situation. However, an
application should enforce separation whenever possible. For
example, an purpose identifier for use in the X.509 extended key
usage certificate extension [RFC5280] could be identified in the
future to indicate that a public key should only be used in
conjunction with a PSK, or only without.

Implementations must randomly generate key-derivation keys as well as
the content-encryption keys or content-authenticated-encryption keys.
Also, the generation of public/private key pairs for the key
transport and key agreement algorithms rely on a random numbers. The

use of inadequate pseudo-random number generators (PRNGs) to generate
cryptographic keys can result in little or no security. An attacker
may find it much easier to reproduce the PRNG environment that
produced the keys, searching the resulting small set of
possibilities, rather than brute force searching the whole key space.
The generation of quality random numbers is difficult. [RFC4086]
offers important guidance in this area.

Implementers should be aware that cryptographic algorithms become
weaker with time. As new cryptoanalysis techniques are developed and
computing performance improves, the work factor to break a particular

cryptographic algorithm will be reduced. Therefore, cryptographic
algorithm implementations should be modular, allowing new algorithms
to be readily inserted. That is, implementors should be prepared for
the set of supported algorithms to change over time.

8. Privacy Considerations

An observer can see which parties are using each PSK simply by
watching the PSK key identifiers. However, the addition of these key
identifiers is not really making privacy worse. When key transport
is used, the RecipientIdentifier is always present, and it clearly
identifies each recipient to an observer. When key agreement is
used, either the IssuerAndSerialNumber or the RecipientKeyIdentifier
is always present, and these clearly identify each recipient.

9. IANA Considerations

One object identifier for the ASN.1 module in the Section 5 was
assigned in the SMI Security for S/MIME Module Identifiers
(1.2.840.113549.1.9.16.0) [IANA-MOD] registry:

   id-mod-cms-ori-psk-2017 OBJECT IDENTIFIER ::= {
      iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
      pkcs-9(9) smime(16) mod(0) TBD0 }

One object identifier for an arc to assign Other Recipient Info
Identifiers was assigned in the SMI Security for S/MIME Mail Security
(1.2.840.113549.1.9.16) [IANA-SMIME] registry:

   id-ori OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
     rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) TBD1 }

Housley   [Page 14]

This assignment created the new SMI Security for Other Recipient Info
Identifiers (1.2.840.113549.1.9.16.TBD1) [IANA-ORI] registry with the
following two entries with references to this document:

```
      id-ori-keyTransPSK OBJECT IDENTIFIER ::= {
         iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
         pkcs-9(9) smime(16) id-ori(TBD1) 1 }

      id-ori-keyAgreePSK OBJECT IDENTIFIER ::= {
         iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
         pkcs-9(9) smime(16) id-ori(TBD1) 2 }
```

10. References

10.1. Normative References

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
              Requirement Levels", BCP 14, RFC 2119, March 1997.

   [RFC5083] Housley, R., "Cryptographic Message Syntax (CMS)
              Authenticated-Enveloped-Data Content Type", RFC 5083,
              November 2007.

   [RFC5652]   Housley, R., "Cryptographic Message Syntax (CMS)", RFC
              5652, September 2009.

   [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
              2119 Key Words", BCP 14, RFC 8174, May 2017.

   [X680]   ITU-T, "Information technology -- Abstract Syntax Notation
              One (ASN.1): Specification of basic notation", ITU-T
              Recommendation X.680, 2015.

   [X690]   ITU-T, "Information technology -- ASN.1 encoding rules:
              Specification of Basic Encoding Rules (BER), Canonical
              Encoding Rules (CER) and Distinguished Encoding Rules
              (DER)", ITU-T Recommendation X.690, 2015.

10.2. Informative References

   [AES]   National Institute of Standards and Technology, FIPS Pub
              197: Advanced Encryption Standard (AES), 26 November 2001.

   [C2PQ]   Hoffman, P., "The Transition from Classical to Post-
              Quantum Cryptography", work-in-progress, draft-hoffman-
              c2pq-03, February 2018.

   [IANA-MOD] https://www.iana.org/assignments/smi-numbers/smi-
               numbers.xhtml#security-smime-0.

   [IANA-SMIME] https://www.iana.org/assignments/smi-numbers/smi-
               numbers.xhtml#security-smime.

   [IANA-ORI] https://www.iana.org/assignments/smi-numbers/smi-
               numbers.xhtml#security-smime-13.

   [RFC2631]   Rescorla, E., "Diffie-Hellman Key Agreement Method",
               RFC 2631, June 1999.

   [RFC3560]   Housley, R., "Use of the RSAES-OAEP Key Transport
               Algorithm in Cryptographic Message Syntax (CMS)",
               RFC 3560, July 2003.

   [RFC4086]   D. Eastlake 3rd, D., Schiller, J., and S. Crocker,
               "Randomness Requirements for Security", RFC 4086,
               June 2005.

   [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, May 2008.

   [RFC5753]   Turner, S., and D. Brown, "Use of Elliptic Curve
               Cryptography (ECC) Algorithms in Cryptographic Message
               Syntax (CMS)", RFC 5753, January 2010.

   [RFC5869]   Krawczyk, H., and P. Eronen, "HMAC-based Extract-and-
               Expand Key Derivation Function (HKDF)", RFC 5869,
               May 2010.

   [RFC5911]   Hoffman, P., and J. Schaad, "New ASN.1 Modules for
               Cryptographic Message Syntax (CMS) and S/MIME", RFC 5911,
               June 2010.

   [RFC5912]   Hoffman, P., and J. Schaad, "New ASN.1 Modules for the
               Public Key Infrastructure Using X.509 (PKIX)" RFC 5912,
               June 2010.

   [RFC8017]   Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch,
               "PKCS #1: RSA Cryptography Specifications Version 2.2",
               RFC 8017, November 2016.

Acknowledgements

   Many thanks to Burt Kaliski, Panos Kampanakis, Jim Schaad, Sean
   Turner, and Daniel Van Geest for their review and insightful
   comments. They have greatly improved the design, clarity, and
   implementation guidance.

Author's Address

   Russell Housley
   Vigil Security, LLC
   516 Dranesville Road
   Herndon, VA 20170
   USA
   EMail: housley@vigilsec.com