

**Using Pre-Shared Key (PSK) in the Cryptographic Message Syntax (CMS)**  
**<[draft-housley-cms-mix-with-psk-00.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 them, if current syntax the does not accommodated 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."

**Copyright Notice**

Copyright (c) 2017 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 (<http://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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">2</a>
<a href="#">1.1.</a>	Terminology . . . . .	<a href="#">3</a>
<a href="#">1.2.</a>	ASN.1 . . . . .	<a href="#">3</a>
<a href="#">1.3.</a>	Version Numbers . . . . .	<a href="#">3</a>
<a href="#">2.</a>	Overview . . . . .	<a href="#">4</a>
<a href="#">3.</a>	KeyTransPSKRecipientInfo . . . . .	<a href="#">5</a>
<a href="#">4.</a>	KeyAgreePSKRecipientInfo . . . . .	<a href="#">6</a>
<a href="#">5.</a>	ASN.1 Module . . . . .	<a href="#">6</a>
<a href="#">6.</a>	Security Considerations . . . . .	<a href="#">8</a>
<a href="#">7.</a>	IANA Considerations . . . . .	<a href="#">9</a>
<a href="#">8.</a>	Normative References . . . . .	<a href="#">10</a>
<a href="#">9.</a>	Informative References . . . . .	<a href="#">10</a>
	Author's Address . . . . .	<a href="#">11</a>

## [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](#)][[RFC5803](#)] supports key transport and key agreement algorithms that could be broken by the invention of a large-scale quantum computer [[C2PQ](#)]. These algorithms include RSA [[RFC4055](#)], Diffie-Hellman [[RFC2631](#)], and Elliptic Curve Diffie-Hellman. 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 current syntax the does not accommodated 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 existing key transport and key agreement algorithms with a pre-shared key (PSK). Secure communication can be achieved today by mixing with a strong PSK with the output of an existing key transport algorithm, like RSA, 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.

### **[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. For enveloped-data, a content-encryption key is established. For authenticated-enveloped-data, a content-authenticated-encryption key is established. All recipients use this same key for decryption.

This specification defines two quantum-resistant ways to establish these keys. In both cases, a 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 authenticated-enveloped-data key is established by following these steps:

1. The key-derivation key is generated at random.
2. The key-derivation key is encrypted for each recipient. The details of this encryption depend on the key management algorithm used:

key transport: the key-derivation key is encrypted in the recipient's public key; or

key agreement: the recipient's public key and the sender's private key are used to generate a pairwise symmetric key, then the key-derivation key is encrypted in the pairwise symmetric key.

3. The key derivation function (KDF) is used to mix the key-derivation key and PSK. The output of the KDF is the content-encryption key or authenticated-enveloped-data key.

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. Each of these is identified by a unique ASN.1 object identifier.

The first key management technique, called keyTransPSK, see [Section 3](#), uses a key transport algorithm to establish the key-derivation key, and then it is mixed with the PSK using a KDF.

The second key management technique, called keyAgreePSK, see [Section](#)



4, uses a key agreement algorithm to establish the key-derivation key, and then it is mixed with the PSK using a KDF.

### 3. KeyTransPSKRecipientInfo

Per-recipient information using keyTransPSK is represented in the KeyTransPSKRecipientInfo type, and its use is indicated by the id-ori-keyTransPSK object identifier. Each instance of KeyTransPSKRecipientInfo establishes the content-encryption key or authenticated-enveloped-data key to one recipient.

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
    ktri KeyTransRecipientInfo,
    pskid PreSharedKeyIdentifier,
    kdfAlgorithm KeyDerivationAlgorithmIdentifier }
```

```
PreSharedKeyIdentifier ::= OCTET STRING
```

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 \[RFC5652\]](#).

ktri is the KeyTransRecipientInfo type described in [Section 6.2.1 of \[RFC5652\]](#); it uses a key transport algorithm to establish the key-derivation key.

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. The KeyDerivationAlgorithmIdentifier is described in [Section 10.1.6 of \[RFC5652\]](#).





#### 4. KeyAgreePSKRecipientInfo

Per-recipient information using keyAgreePSK is represented in the KeyAgreePSKRecipientInfo type, and its use is indicated by the id-ori-keyAgreePSK object identifier. Each instance of KeyAgreePSKRecipientInfo establishes the content-encryption key or authenticated-enveloped-data key to one recipient.

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  
    kari KeyAgreeRecipientInfo,  
    pskid PreSharedKeyIdentifier,  
    kdfAlgorithm KeyDerivationAlgorithmIdentifier }
```

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 \[RFC5652\]](#).

kari is the KeyAgreeRecipientInfo type described in [Section 6.2.2 of \[RFC5652\]](#); it uses a key agreement algorithm to establish the key-derivation key.

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. The KeyDerivationAlgorithmIdentifier is described in [Section 10.1.6 of \[RFC5652\]](#).

#### 5. 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

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

DEFINITIONS IMPLICIT 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, KeyAgreeRecipientInfo,  
KeyDerivationAlgorithmIdentifier

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
    ktri KeyTransRecipientInfo,
    pskid PreSharedKeyIdentifier,
    kdfAlgorithm KeyDerivationAlgorithmIdentifier }

PreSharedKeyIdentifier ::= OCTET STRING

--
-- 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
    kari KeyAgreeRecipientInfo,
    pskid PreSharedKeyIdentifier,
    kdfAlgorithm KeyDerivationAlgorithmIdentifier }

END
```

## 6. Security Considerations

Implementations must protect the pre-shared key (PSK). Compromise of the PSK will make the the encrypted content vulnerable to the future invention of a large-scale quantum computer.

Implementations must protect the key transport private key, the agreement private key, and the key-derivation key. Compromise of the private keying material may result in the disclosure of all contents protected with that key. Similarly, compromise of the key-derivation key that is established with the private keying material may result in disclosure of the associated encrypted content.

Implementations must protect the content-encryption key and the content-authenticated-encryption key. Compromise of the content-encryption key or content-authenticated-encryption key may result in disclosure of the associated encrypted content.

Implementations must randomly generate key-derivation key. 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.

When using 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 be aware that cryptographic algorithms become weaker with time. As new cryptanalysis 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.

## **7. 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-ecdh-alg-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 }
```





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 }
```

## 8. 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.

## 9. 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-02](#), August 2017.
- [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.
- [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.

#### Author's Address

Russell Housley  
Vigil Security, LLC  
918 Spring Knoll Drive  
Herndon, VA 20170  
USA  
EMail: [housley@vigilsec.com](mailto:housley@vigilsec.com)

