Internet Engineering Task Force (IETF)
Request for Comments: 6032
Category: Standards Track
ISSN: 2070-1721

S. Turner
IECA
R. Housley
Vigil Security

December 2010

Cryptographic Message Syntax (CMS) Encrypted Key Package Content Type

Abstract

This document defines the Cryptographic Message Syntax (CMS) encrypted key package content type, which can be used to encrypt a content that includes a key package, such as a symmetric key package or an asymmetric key package. It is transport independent. CMS can be used to digitally sign, digest, authenticate, or further encrypt this content type. It is designed to be used with the CMS Content Constraints (CCC) extension, which does not constrain the EncryptedData, EnvelopedData, and AuthEnvelopedData.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6032.

1. Introduction

The Cryptographic Message Syntax (CMS) specification [RFC5652] defines mechanisms to digitally sign, digest, authenticate, or encrypt arbitrary message content. Many specifications define content types intended for use with CMS. [RFC6031] and [RFC5958] define symmetric key package and asymmetric key package content types that can be signed or encrypted using CMS. CMS allows the composition of complex messages with an arbitrary number of layers. CMS has been augmented by several specifications ([RFC3274], [RFC4073], and [RFC5083]) that define additional mechanisms to enable creation of messages of arbitrary depth and breadth using a variety of authentication, encryption, and compression techniques.

The CMS Content Constraints (CCC) certificate extension [RFC6010] defines an authorization mechanism that allows recipients to determine whether the originator of an authenticated CMS content type is authorized to produce messages of that type. CCC is used to authorize CMS-protected content. CCC cannot be used to constrain the following structures that are used to provide layers of protection: SignedData, EnvelopedData, EncryptedData, DigestData, CompressedData, AuthenticatedData, ContentCollection, ContentWithAttributes, or AuthEnvelopedData.

Using the existing CMS mechanisms, producers of authenticated plaintext key packages can be authorized by including a CCC extension containing the appropriate content type in the producer's certificate. However, these mechanisms cannot be used to authorize the producers of encrypted key material. In some key management systems, encrypted key packages are exchanged between entities that cannot decrypt the key package. The encrypted key package itself may

be authenticated and passed to another entity. In these cases, checking the authorization of the producer of the encrypted key package may be desired at the intermediate points.

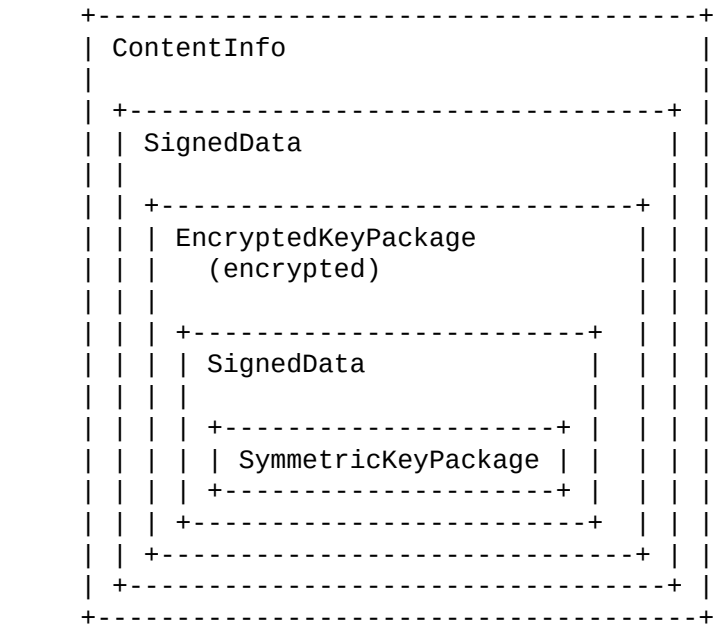
This document defines the encrypted key package content type, which can be used to encrypt a content that includes a key package, such as a symmetric key package [RFC6031] or an asymmetric key package [RFC5958]. It is transport independent. The Cryptographic Message Syntax (CMS) [RFC5652] can be used to digitally sign, digest, authenticate, or further encrypt this content type.

The encrypted key package content type is designed for use with [RFC6010]. To authorize an originator's public key to originate an encrypted key package, the object identifier associated with the encrypted key package content type is included in the originator's public key certificate CCC certificate extension. For CCC to function, originators encapsulate the encrypted key package in a SignedData, EnvelopedData, or AuthEnvelopedData; then, during certificate path validation, the recipient determines whether the originator is authorized to originate the encrypted key package.

In [RFC6010] terminology, the encrypted key package is a leaf node. Additional authorization checks may be required once the key package is decrypted. For example, the key package shown below consists of a SignedData layer that encapsulates an encrypted key package that encapsulates a SignedData layer containing a symmetric key package. A recipient capable of decrypting the key package would perform the following steps prior to accepting the encapsulated symmetric key material:

- Verify the signature on the outer SignedData layer per [RFC5652].
- Build and validate a certification path of the outer signer and confirm the outer signer is authorized to produce the encrypted key package per [RFC5280] and [RFC6010].
- Decrypt the encrypted key package.
- Verify the signature on the inner SignedData layer per [RFC5652].
- Build and validate a certification path to the signer of the inner SignedData and confirm the inner signer is authorized to produce the symmetric key package per [RFC5280] and [RFC6010]. As specified in [RFC6010], the validator may use the attributes and public keys returned from the second step as inputs for this CMS content constraints processing.

o Use the symmetric key material.



In the example, authorization of the SymmetricKeyPackage originator need not require an intermediate SignedData layer. For example, if the AuthEnvelopedData option within an EncryptedKeyPackage were used, the second authorization check would be performed beginning with the authEnveloped field.

This document also defines an unprotected attribute, Content Decryption Key Identifier, for use with EncryptedData.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.2. ASN.1 Syntax Notation

The key package is defined using the ASN.1 [X.680], [X.681], [X.682], and [X.683].

2. Encrypted Key Package

The encrypted key package content type is used to encrypt a content that includes a key package. This content type is usually used to provide encryption of a key package or a signed key package. This

content type makes use of the CMS EncryptedData content type [RFC5652], the CMS EnvelopedData content type [RFC5652], or the CMS AuthEnvelopedData content type [RFC5083] depending on the fields that are needed for key management. The difference between the encrypted key package content type and these three protecting content types is the object identifier and one tag; otherwise, the encrypted key package content type is the same as the selected protecting content type, which is either EncryptedData, EnvelopedData, or AuthEnvelopedData.

The encrypted key package content type has the following syntax:

```
ct-encrypted-key-package CONTENT-TYPE ::=
  { TYPE EncryptedKeyPackage
    IDENTIFIED BY id-ct-KP-encryptedKeyPkg }

id-ct-KP-encryptedKeyPkg OBJECT IDENTIFIER ::=
  { joint-iso-itu-t(2) country(16) us(840) organization(1)
    gov(101) dod(2) infosec(1) formats(2)
    key-package-content-types(78) 2 }

EncryptedKeyPackage ::= CHOICE {
  encrypted        EncryptedData,
  enveloped        [0] EnvelopedData,
  authEnveloped    [1] AuthEnvelopedData }
```

The EncryptedData structure is used for simple symmetric encryption, where the sender and the receiver already share the necessary encryption key. The EncryptedData structure carries an encryption algorithm identifier, and an unprotected attribute can be used to carry an encryption key identifier if one is needed (see Section 3). See [RFC5652] for further discussion of the EncryptedData fields.

The EnvelopedData structure is used for encryption, where transferred key management information enables decryption by the receiver. Encryption details depend on the key management algorithm used. In addition to the key management information, the EnvelopedData structure carries an encryption algorithm identifier. See [RFC5652] for further discussion of the EnvelopedData fields.

The AuthEnvelopedData structure is used for authenticated encryption, and it includes key management information in a manner similar to EnvelopedData. Encryption details depend on the key management algorithm used. In addition to the key management information, the AuthEnvelopedData structure carries a message authentication code that covers the content as well as authenticated attributes. See [RFC5083] for further discussion of the AuthEnvelopedData fields.

Implementations of this document MUST support the EnvelopedData choice, SHOULD support the EncryptedData choice, and MAY support the AuthEnvelopedData.

Implementations that support EnvelopedData and EncryptedData to encapsulate with this content type MUST support an EncryptedKeyPackage that encapsulates either a SignedData [RFC5652] that further encapsulates a SymmetricKeyPackage [RFC6031] or a SignedData that further encapsulates an AsymmetricKeyPackage [RFC5958]. Implementations that support AuthEnvelopedData to encapsulate with this content type MUST support an EncryptedKeyPackage that encapsulates either a SymmetricKeyPackage [RFC6031] or an AsymmetricKeyPackage [RFC5958]. It is OPTIONAL for implementations that support AuthEnvelopedData to encapsulate with this content type to support an EncryptedKeyPackage that encapsulates either a SignedData [RFC5652] that further encapsulates a SymmetricKeyPackage [RFC6031] or a SignedData that further encapsulates an AsymmetricKeyPackage [RFC5958]. Likewise, implementations that process this content type to decrypt the encapsulated data MUST support an EncryptedKeyPackage that encapsulates either a SignedData that further encapsulates a SymmetricKeyPackage or a SignedData that further encapsulates an AsymmetricKeyPackage. An EncryptedKeyPackage content type MUST contain at least one SymmetricKeyPackage or AsymmetricKeyPackage. Implementations MAY support additional encapsulating layers.

Note that interoperability between an originator and a recipient that do not support the same innermost content (e.g., originator supports AsymmetricKeyPackage while recipient supports SymmetricKeyPackage) is not a concern as originators should be aware of the recipient's capabilities; however, the mechanism for the exchange of the recipient's capabilities is beyond the scope of this document.

3. Content Decryption Key Identifier

The content-decryption-key-identifier attribute can be used to identify the symmetric keying material that is needed for decryption of the EncryptedData content if there is any ambiguity. The ATTRIBUTE definition is taken from [RFC5912]. There MUST be only one instance of the content-decryption-key-identifier attribute and there MUST be only one value for the content-decryption-key-identifier attribute.

The content-decryption-key-identifier attribute has the following syntax:

```
aa-content-decrypt-key-identifier ATTRIBUTE ::= {
  TYPE          ContentDecryptKeyID
  IDENTIFIED BY id-aa-KP-contentDecryptKeyID }

id-aa-KP-contentDecryptKeyID OBJECT IDENTIFIER ::= {
  joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
  dod(2) infosec(1) attributes(5) 66 }

ContentDecryptKeyID ::= OCTET STRING
```

The content decryption key identifier contains an OCTET STRING, and this syntax does not impose any particular structure on the identifier value.

Due to multiple layers of encryption, the content-decryption-key-identifier attribute can appear in more than one location in the overall key package. When there are multiple occurrences of the content-decryption-key-identifier attribute, each occurrence is evaluated independently. Each one is used to identify the needed keying material for that layer of encryption.

4. Security Considerations

Implementers of this protocol are strongly encouraged to consider generally accepted principles of secure key management when integrating this capability within an overall security architecture.

The security considerations from [RFC5083], [RFC5652], [RFC5911], [RFC5912], [RFC5958], and [RFC6031] apply. If the CCC extension is used as an authorization mechanism, then the security considerations from [RFC6010] also apply.

The encrypted key package content type might not provide proof of origin if the content encryption algorithm does not support authenticated key exchange. To provide proof of origin for this content, another security protocol needs to be used. This is the reason that support for encapsulating the SymmetricKeyPackage and AsymmetricKeyPackage with a SignedData content type from [RFC5652] is required for the EnvelopedData and EncryptedData choices.

When this content type is used the CMS SignedData [RFC5652] validation rules MUST be used. The PKCS #7 [RFC2315] validation rules MUST NOT be used.

5. IANA Considerations

This document makes use of object identifiers to identify a CMS content type, a CMS attribute, and the ASN.1 module; all found in Appendix A. All OIDs are registered in an arc delegated by RSADSI to the SMIME Working Group.

6. References

6.1. 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.

[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.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, September 2009.

[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.

[RFC5958] Turner, S., "Asymmetric Key Packages", RFC 5958, August 2010.

[RFC6010] Housley, R., Ashmore, S., and C. Wallace, "Cryptographic Message Syntax (CMS) Content Constraints Extension", RFC 6010, September 2010.

[RFC6031] Turner, S. and R. Housley, "Cryptographic Message Syntax (CMS) Symmetric Key Package Content Type", RFC 6031, December 2010.

[X.680] ITU-T Recommendation X.680 (2002) | ISO/IEC 8824-1:2002. Information Technology - Abstract Syntax Notation One.

[X.681] ITU-T Recommendation X.681 (2002) | ISO/IEC 8824-2:2002. Information Technology - Abstract Syntax Notation One: Information Object Specification.

[X.682] ITU-T Recommendation X.682 (2002) | ISO/IEC 8824-3:2002. Information Technology - Abstract Syntax Notation One: Constraint Specification.

[X.683] ITU-T Recommendation X.683 (2002) | ISO/IEC 8824-4:2002. Information Technology - Abstract Syntax Notation One: Parameterization of ASN.1 Specifications.

6.2. Informative References

[RFC2315] Kaliski, B., "PKCS #7: Cryptographic Message Syntax Version 1.5", RFC 2315, March 1998.

[RFC3274] Gutmann, P., "Compressed Data Content Type for Cryptographic Message Syntax (CMS)", RFC 3274, June 2002.

[RFC4073] Housley, R., "Protecting Multiple Contents with the Cryptographic Message Syntax (CMS)", RFC 4073, May 2005.

Appendix A. ASN.1 Module

This appendix provides the normative ASN.1 [X.680] definitions for the structures described in this specification using ASN.1, as defined in [X.680] through [X.683].

```
EncryptedKeyPackageModuleV1
  { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
    smime(16) modules(0) id-mod-encryptedKeyPkgV1(51) }

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

-- EXPORTS ALL --

IMPORTS

-- From New SMIME ASN.1 [RFC5911]

EncryptedData, EnvelopedData, CONTENT-TYPE
  FROM CryptographicMessageSyntax-2009
    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
      smime(16) modules(0) cms-2004-02(41) }

-- From New SMIME ASN.1 [RFC5911]

AuthEnvelopedData
  FROM CMS-AuthEnvelopedData-2009
     { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
       pkcs-9(9) smime(16) modules(0) cms-authEnvelopedData-02(43) }

-- From New PKIX ASN.1 [RFC5912]

ATTRIBUTE
  FROM PKIX-CommonTypes-2009
    { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-pkixCommon-02(57) }

;

ContentSet CONTENT-TYPE ::= {
  ct-encrypted-key-package,
  ... -- Expect additional content types --
}
```

```
   ct-encrypted-key-package CONTENT-TYPE ::=
       { TYPE EncryptedKeyPackage
         IDENTIFIED BY id-ct-KP-encryptedKeyPkg }

   id-ct-KP-encryptedKeyPkg OBJECT IDENTIFIER ::=
     { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
       dod(2) infosec(1) formats(2) key-package-content-types(78) 2 }

   EncryptedKeyPackage ::= CHOICE {
       encrypted        EncryptedData,
       enveloped        [0] EnvelopedData,
       authEnveloped    [1] AuthEnvelopedData }

   aa-content-decrypt-key-identifier ATTRIBUTE ::= {
       TYPE          ContentDecryptKeyID
       IDENTIFIED BY id-aa-KP-contentDecryptKeyID }

   id-aa-KP-contentDecryptKeyID OBJECT IDENTIFIER ::= {
     joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
     dod(2) infosec(1) attributes(5) 66 }

   ContentDecryptKeyID ::= OCTET STRING

   END
```

Authors' Addresses


```
   Sean Turner
   IECA, Inc.
   3057 Nutley Street, Suite 106
   Fairfax, VA 22031
   USA

   EMail: turners@ieca.com


   Russell Housley
   Vigil Security, LLC
   918 Spring Knoll Drive
   Herndon, VA 20170
   USA

   EMail: housley@vigilsec.com
```