Internet Engineering Task Force (IETF)
Request for Comments: 7191
Category: Standards Track
ISSN: 2070-1721

R. Housley
Vigil Security

April 2014

Cryptographic Message Syntax (CMS) Key Package Receipt and Error Content Types

Abstract

This document defines the syntax for two Cryptographic Message Syntax (CMS) content types: one for key package receipts and another for key package errors. The key package receipt content type is used to confirm receipt of an identified key package or collection of key packages. The key package error content type is used to indicate an error occurred during the processing of a key package. CMS can be used to digitally sign, digest, authenticate, or encrypt these content types.

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/rfc7191.

Table of Contents

1. Introduction

This document defines the syntax for two Cryptographic Message Syntax (CMS) [RFC5652] content types: one for key package receipts and another for key package errors. The key package receipt content type is used to confirm receipt of an identified key package or collection of key packages. The key package error content type is used to indicate an error occurred during the processing of a key package. CMS can be used to digitally sign, digest, authenticate, or encrypt these content types.

1.1. Requirements 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 content types defined herein use ASN.1 ([X.680], [X.681], [X.682], and [X.683]).

The CONTENT-TYPE definition was updated to the 2008 version of ASN.1 by [RFC6268]; however, none of the new 2008 ASN.1 tokens are used in this specification, which allows compilers that only support the 2002 version of ASN.1 to compile the module in Appendix A.

1.3. Processing Key Package Receipt Requests

The key package or collection of key packages [RFC4073] [RFC5958] [RFC6031] [RFC6032] for which the receipt is being generated MUST be signed, and the key package MUST include the key-package-identifier-and-receipt-request attribute specified in Section 3.

1.4. Processing Key Packages with Errors

The key package or collection of key packages [RFC4073] [RFC5958] [RFC6031] [RFC6032] for which the error is being generated might be signed. The key package can be identified by a key-package-identifier-and-receipt-request attribute specified in Section 3.

2. SIR Entity Name

Within a key distribution system, the source, intermediary, and receiver entities are identified by a Source Intermediary Recipient (SIR) entity name. The syntax for the SIR entity name does not impose any particular structure, and it accommodates straightforward registration of additional SIR entity name types.

The inclusion of the nameType object identifier ensures that two identifiers of different types that happen to contain the same values are not interpreted as equivalent. Additional SIR entity name types are expected to be registered that represent different granularities. For example, one SIR entity name type might represent the receiver organization, and at a finer granularity, another SIR entity name type might identify a specific device, perhaps using a manufacturer identifier and serial number. The use of an object identifier avoids the need for a central registry of SIR entity name types.

The nameValue is an OCTET STRING, which allows the canonical form of any name to be carried. Two names of the same type are considered equal if the octet strings are the same length and contain the same string of octets.

SIREntityNames and SIREntityName have the following syntax:

```
SIREntityNames ::= SEQUENCE SIZE (1..MAX) OF SIREntityName

SIR-ENTITY-NAME ::= CLASS {
    &sIRENType OBJECT IDENTIFIER UNIQUE,
    &SIRENValue
    } WITH SYNTAX {
    SYNTAX &SIRENValue IDENTIFIED BY &sIRENType }

SIREntityName ::= SEQUENCE {
    sirenType    SIR-ENTITY-NAME.&sIRENType({SIREntityNameTypes}),
    sirenValue   OCTET STRING (CONTAINING
                   SIR-ENTITY-NAME.&SIRENValue(
                     {SIREntityNameTypes}{@sirenType}) ) }
```

This document defines one SIR entity name type: the DN type. The DN type uses a nameType of id-dn and a nameValue of a Distinguished Name (DN). The nameValue OCTET STRING carries an ASN.1 encoded Name as specified in [RFC5280]. Note that other documents may define additional types.

```
SIREntityNameTypes SIR-ENTITY-NAME ::= {
    siren-dn,
    ... -- Expect additional SIR Entity Name types -- }

siren-dn SIR-ENTITY-NAME ::= {
    SYNTAX DistinguishedName
    IDENTIFIED BY id-dn }

id-dn OBJECT IDENTIFIER ::= {
    joint-iso-ccitt(2) country(16) us(840) organization(1)
    gov(101) dod(2) infosec(1) sir-name-types(16) 0 }
```

3. Key Package Identifier and Receipt Request Attribute

The key-package-identifier-and-receipt-request attribute, as its name implies, allows the originator to identify the key package and, optionally, request receipts. This attribute can appear as a signed, authenticated, and content attribute. Signed attributes are carried in the CMS Signed-data content type described in Section 5 of [RFC5652]. Authenticated attributes are carried in the CMS Authenticated-data content type described in Section 9 of [RFC5652] or in the CMS Authenticated-enveloped-data content type described in Section 2 of [RFC5083]. Content attributes are carried in the Content-with-attributes content type described in Section 3 of [RFC4073].

The key-package-identifier-and-receipt-request attribute has the following syntax:

```
  aa-keyPackageIdentifierAndReceiptRequest ATTRIBUTE ::= {
      TYPE KeyPkgIdentifierAndReceiptReq
      IDENTIFIED BY id-aa-KP-keyPkgIdAndReceiptReq }

  id-aa-KP-keyPkgIdAndReceiptReq OBJECT IDENTIFIER ::= {
      joint-iso-itu-t(2) country(16) us(840) organization(1)
      gov(101) dod(2) infosec(1) attributes(5) 65 }

  KeyPkgIdentifierAndReceiptReq ::= SEQUENCE {
      pkgID       KeyPkgID,
      receiptReq  KeyPkgReceiptReq OPTIONAL }

  KeyPkgID ::= OCTET STRING

  KeyPkgReceiptReq ::= SEQUENCE {
      encryptReceipt     BOOLEAN DEFAULT FALSE,
      receiptsFrom   [0] SIREntityNames OPTIONAL,
      receiptsTo         SIREntityNames }
```

Even though the ATTRIBUTE syntax is defined as a SET OF AttributeValue, a key-package-identifier-and-receipt-request attribute MUST have a single attribute value; zero or multiple instances of AttributeValue are not permitted.

The fields in the key-package-identifier-and-receipt-request attribute have the following semantics:

- pkgID contains an octet string, and this syntax does not impose any particular structure on the identifier.

- receiptReq is OPTIONAL, and when it is present, it includes an encryption receipt flag, an OPTIONAL indication of which receivers should generate receipts, and an indication of where the receipts are to be sent.

  * The encryption receipt flag indicates whether the key package originator wants the receipt to be encrypted. If the boolean is set, then the receipt SHOULD be encrypted.

  * The OPTIONAL ReceiptsFrom field provides an indication of which receivers SHOULD generate receipts. When the ReceiptsFrom field is absent, all receivers of the key package are expected to return receipts. When the ReceiptsFrom field is present, a list of SIR entity names indicates which receivers of the key package are requested to return receipts.

In this case, the receiver SHOULD return a receipt only if their SIR entity name appears on the list.

* The receipt request does not include any key management information; however, the list of SIR entity names in the receiptsTo field can be used to select symmetric or asymmetric keying material for the receipt receivers.

A receiver SHOULD ignore the nameValue associated with any unrecognized nameType in either the receiptsFrom field or the receiptsTo field.

When the key-package-identifier-and-receipt-request attribute appears in more than one location in the overall key package, each occurrence is evaluated independently. That is, the receiver may generate more than one receipt for a single key package. However, the time at which the receipts are sent will depend on policies that are beyond the scope of this document.

4. Key Package Receipt CMS Content Type

The key package receipt content type is used to confirm receipt of an identified key package or collection of key packages. This content type MUST be encoded using the Distinguished Encoding Rules (DER) [X.690].

The key package receipt content type has the following syntax:

```
ct-key-package-receipt CONTENT-TYPE ::= {
    TYPE KeyPackageReceipt
    IDENTIFIED BY id-ct-KP-keyPackageReceipt }

id-ct-KP-keyPackageReceipt 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) 3 }

KeyPackageReceipt ::= SEQUENCE {
    version          KeyPkgVersion DEFAULT v2,
    receiptOf        KeyPkgIdentifier,
    receivedBy       SIREntityName }

-- Revised definition of KeyPkgVersion from [RFC6031]
KeyPkgVersion ::= INTEGER  { v1(1), v2(2) } (1 .. 65535)

KeyPkgIdentifier ::= CHOICE {
    pkgID            KeyPkgID,
    attribute        SingleAttribute {{ KeyPkgIdentifiers }} }
```

```
KeyPkgID ::= OCTET STRING

KeyPkgIdentifiers ATTRIBUTE ::= { ... }
```

The KeyPackageReceipt fields are used as follows:

- version identifies version of the key package receipt content. For this version of the specification, the default value, v2, MUST be used. Note that v1 was defined in an earlier version, but the use of v1 is deprecated.

- receiptOf offers two alternatives for identifying the key package for which the receipt is being generated. The first alternative, pkgID, MUST be supported, and pkgID provides the key package identifier of the key package or collection of key packages for which this receipt is being generated. This key package identifier value MUST exactly match the key package identifier value of the key-package-identifier-and-receipt-request attribute in the received key package or collection. The key-package-identifier-and-receipt-request attribute is described Section 3. The second alternative allows alternate attributes to be used to define the identifier.

- receivedBy identifies the entity that received the key package. The entity is named by an SIR entity name as specified in Section 2.

Key package receipts MUST be encapsulated in a CMS SignedData content type to carry the signature of the entity that is confirming receipt of the identified key package or collection of key packages. Key package receipts MAY be encrypted by encapsulating them in the CMS EncryptedData content type, the CMS EnvelopedData content type, or the AuthEnvelopedData content type. When the key package receipt is signed and encrypted, it MUST be signed prior to being encrypted.

Note that delivery assurance is the responsibility of the protocol that is used to transport and track key packages. The key package receipt content type can be used in conjunction with that protocol as part of an overall delivery assurance solution.

Because the receipts are signed, all recipients that generate key package receipts MUST have a private signature key to sign the receipt as well as store their own certificate or have a means of obtaining the key identifier of their public key. If memory is a concern, the public key identifier can be computed from the public key.

If the receipt signer has access to a real-time clock, then the binary-signing-time [RFC6019] signed attribute SHOULD be included in the key package receipt to provide the date and time when it was generated.

5. Key Package Error CMS Content Type

The key package error content type provides an indication of the reason for rejection of a key package or collection of key packages. This content type MUST be encoded using the Distinguished Encoding Rules (DER) [X.690].

The key package error content type has the following syntax:

```
ct-key-package-error CONTENT-TYPE ::= {
    TYPE KeyPackageError IDENTIFIED BY id-ct-KP-keyPackageError }

id-ct-KP-keyPackageError 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) 6 }

KeyPackageError ::= SEQUENCE {
    version        KeyPkgVersion DEFAULT v2,
    errorOf    [0] KeyPkgIdentifier OPTIONAL,
    errorBy        SIREntityName,
    errorCode      ErrorCodeChoice }

KeyPkgVersion ::= INTEGER  { v1(1), v2(2) } (1 .. 65535)

KeyPkgIdentifier ::= CHOICE {
    pkgID            KeyPkgID,
    attribute        SingleAttribute {{ KeyPkgIdentifiers }} }

KeyPkgID ::= OCTET STRING

KeyPkgIdentifiers ATTRIBUTE ::= { ... }

ErrorCodeChoice ::= CHOICE {
    enum           EnumeratedErrorCode,
    oid            OBJECT IDENTIFIER }

EnumeratedErrorCode ::= ENUMERATED {
    decodeFailure                    (1),
    badContentInfo                   (2),
    badSignedData                    (3),
    badEncapContent                  (4),
    badCertificate                   (5),
```

```
        badSignerInfo                    (6),
        badSignedAttrs                   (7),
        badUnsignedAttrs                 (8),
        missingContent                   (9),
        noTrustAnchor                   (10),
        notAuthorized                   (11),
        badDigestAlgorithm              (12),
        badSignatureAlgorithm           (13),
        unsupportedKeySize              (14),
        unsupportedParameters           (15),
        signatureFailure                (16),
        insufficientMemory              (17),
        incorrectTarget                 (23),
        missingSignature                (29),
        resourcesBusy                   (30),
        versionNumberMismatch           (31),
        revokedCertificate              (33),

    --  Error codes with values <= 33 are aligned with [RFC5934]

        ambiguousDecrypt                (60),
        noDecryptKey                    (61),
        badEncryptedData                (62),
        badEnvelopedData                (63),
        badAuthenticatedData            (64),
        badAuthEnvelopedData            (65),
        badKeyAgreeRecipientInfo        (66),
        badKEKRecipientInfo             (67),
        badEncryptContent               (68),
        badEncryptAlgorithm             (69),
        missingCiphertext               (70),
        decryptFailure                  (71),
        badMACAlgorithm                 (72),
        badAuthAttrs                    (73),
        badUnauthAttrs                  (74),
        invalidMAC                      (75),
        mismatchedDigestAlg             (76),
        missingCertificate              (77),
        tooManySigners                  (78),
        missingSignedAttributes         (79),
        derEncodingNotUsed              (80),
        missingContentHints             (81),
        invalidAttributeLocation        (82),
        badMessageDigest                (83),
        badKeyPackage                   (84),
        badAttributes                   (85),
        attributeComparisonFailure      (86),
        unsupportedSymmetricKeyPackage  (87),
```

```
        unsupportedAsymmetricKeyPackage  (88),
        constraintViolation              (89),
        ambiguousDefaultValue            (90),
        noMatchingRecipientInfo          (91),
        unsupportedKeyWrapAlgorithm      (92),
        badKeyTransRecipientInfo         (93),
        other                           (127),
        ... -- Expect additional error codes  -- }
```

The KeyPackageError fields are used as follows:

- o version identifies version of the key package error content structure. For this version of the specification, the default value, v2, MUST be used. Note that v1 was defined in an earlier version, but the use of v1 is deprecated.

- o errorOf is OPTIONAL, and it provides the identifier of the keying material for which this error is being generated. This is omitted if the receiver or intermediary cannot parse the received data to determine the package identifier. Also, encryption may prevent an intermediary from obtaining any of the identifiers. Two alternatives for identifying the keying material are possible; see KeyPkgIdentifier as described in Section 4. The value MUST exactly match the value of the key-package-identifier-and-receipt-request attribute in the received key package or collection. The key-package-identifier-and-receipt-request attribute is described in Section 3.

- o errorBy identifies the entity that received the key package. The entity is named by an SIR entity name as specified in Section 2.

- o errorCode contains a code that indicates the reason for the error. It contains either an enumerated error code from the list below or an extended error code represented by an object identifier. The enumerated error code alternative MUST be supported. The object identifier error code MAY be supported.

  - * decodeFailure is used to indicate that the key package intermediary or receiver was unable to successfully decode the provided package. The specified content type and the provided content do not match.

  - * badContentInfo is used to indicate that the ContentInfo syntax is invalid or that the contentType carried within the ContentInfo is unknown or unsupported.

* badSignedData is used to indicate that the SignedData syntax is invalid, the version is unknown or unsupported, or more than one entry is present in digestAlgorithms.

* badEncapContent is used to indicate that the EncapsulatedContentInfo syntax is invalid within a SignedData or an AuthenticatedData or the EncryptedContentInfo syntax is invalid within an AuthEnvelopedData.

* badCertificate is used to indicate that the syntax for one or more certificates in CertificateSet or elsewhere is invalid or unsupported.

* badSignerInfo is used to indicate that the SignerInfo syntax is invalid or the version is unknown or unsupported.

* badSignedAttrs is used to indicate that the signedAttrs syntax within SignerInfo is invalid.

* badUnsignedAttrs is used to indicate that the unsignedAttrs within SignerInfo contains one or more attributes. Since unrecognized attributes are ignored, this error code is used when the object identifier for the attribute is recognized, but the value is malformed or internally inconsistent. In addition, this error code can be used when policy prohibits an implementation from supporting unsigned attributes.

* missingContent is used to indicate that the optional eContent is missing in EncapsulatedContentInfo, which is required when including an asymmetric key package, a symmetric key package, and an encrypted key package. This error can be generated due to problems located in SignedData or AuthenticatedData.

  Note that CMS EncapsulatedContentInfo eContent field is optional [RFC5652]; however, [RFC5958], [RFC6031], and [RFC6032] require that the eContent be present.

* noTrustAnchor is used to indicate that the subjectKeyIdentifier does not identify the public key of a trust anchor or a certification path that terminates with an installed trust anchor.

* notAuthorized is used to indicate that the sid within SignerInfo leads to an installed trust anchor, but that trust anchor is not an authorized signer for the received content type.

* badDigestAlgorithm is used to indicate that the digestAlgorithm in either SignerInfo, SignedData, or AuthenticatedData is unknown or unsupported.

* badSignatureAlgorithm is used to indicate that the signatureAlgorithm in SignerInfo is unknown or unsupported.

* unsupportedKeySize is used to indicate that the signatureAlgorithm in SignerInfo is known and supported, but the digital signature could not be validated because an unsupported key size was employed by the signer. Alternatively, the algorithm used in EnvelopedData, AuthenticatedData, or AuthEnvelopedData to generate the key-encryption key is known and supported, but an unsupported key size was employed by the originator.

* unsupportedParameters is used to indicate that the signatureAlgorithm in SignerInfo is known, but the digital signature could not be validated because unsupported parameters were employed by the signer. Alternatively, the algorithm used in EnvelopedData, AuthenticatedData, or AuthEnvelopedData to generate the key-encryption key is known and supported, but unsupported parameters were employed by the originator.

* signatureFailure is used to indicate that the signatureAlgorithm in SignerInfo is known and supported, but the digital signature in the signature field within SignerInfo could not be validated.

* insufficientMemory indicates that the key package could not be processed because the intermediary or receiver did not have sufficient memory to store the keying material.

* incorrectTarget indicates that a receiver is not the intended recipient.

* missingSignature indicates that the receiver requires the key package to be signed or authenticated with a Message Authentication Code (MAC), but the received key package was not signed or authenticated.

* resourcesBusy indicates that the resources necessary to process the key package are not available at the present time, but the resources might be available at some point in the future.

* versionNumberMismatch indicates that the version number in a received key package is not acceptable.

* revokedCertificate indicates that one or more of the certificates needed to properly process the key package has been revoked.

* ambiguousDecrypt indicates that the EncryptedData content type was used, and the key package receiver could not determine the appropriate keying material to perform the decryption.

* noDecryptKey indicates that the receiver does not have the key named in the content-decryption-key-identifier attribute (see [RFC6032]).

* badEncryptedData indicates that the EncryptedData syntax is invalid or the version is unknown or unsupported.

* badEnvelopedData indicates that the EnvelopedData syntax is invalid or the version is unknown or unsupported.

* badAuthenticatedData indicates that the AuthenticatedData syntax is invalid or the version is unknown or unsupported.

* badAuthEnvelopedData indicates that the AuthEnvelopedData syntax is invalid or the version is unknown or unsupported.

* badKeyAgreeRecipientInfo indicates that the KeyAgreeRecipientInfo syntax is invalid or the version is unknown or unsupported.

* badKEKRecipientInfo indicates that the KEKRecipientInfo syntax is invalid or the version is unknown or unsupported.

* badEncryptContent indicates that the EncryptedContentInfo syntax is invalid, or that the content type carried within the contentType is unknown or unsupported.

* badEncryptAlgorithm indicates that the encryption algorithm identified by contentEncryptionAlgorithm in EncryptedContentInfo is unknown or unsupported. This can result from EncryptedData, EnvelopedData, or AuthEnvelopedData.

* missingCiphertext indicates that the optional encryptedContent is missing in EncryptedContentInfo, which is required when including an asymmetric key package, a symmetric key package, and an encrypted key package.

* decryptFailure indicates that the encryptedContent in EncryptedContentInfo did not decrypt properly.

* badMACAlgorithm indicates that the MAC algorithm identified by MessageAuthenticationCodeAlgorithm in AuthenticatedData is unknown or unsupported.

* badAuthAttrs is used to indicate that the authAttrs syntax within AuthenticatedData or AuthEnvelopedData is invalid. Since unrecognized attributes are ignored, this error code is used when the object identifier for the attribute is recognized, but the value is malformed or internally inconsistent.

* badUnauthAttrs is used to indicate that the unauthAttrs syntax within AuthenticatedData or AuthEnvelopedData is invalid. Since unrecognized attributes are ignored, this error code is used when the object identifier for the attribute is recognized, but the value is malformed or internally inconsistent.

* invalidMAC is used to indicate that the message authentication code value within AuthenticatedData or AuthEnvelopedData did not validate properly.

* mismatchedDigestAlg is used to indicate that the digest algorithm in digestAlgorithms field within SignedData does not match the digest algorithm used in the signature algorithm.

* missingCertificate indicates that a signature could not be verified using a trust anchor or a certificate from the certificates field within SignedData. Similarly, this error code can indicate that a needed certificate is missing when processing EnvelopedData, AuthEnvelopedData, or AuthenticatedData.

* tooManySigners indicates that a SignedData content contained more than one SignerInfo for a content type that requires only one signer.

* missingSignedAttributes indicates that a SignedInfo within a SignedData content did not contain any signed attributes; at a minimum, the content-type and message-digest must be present, as per [RFC5652]. Similarly, this error code can indicate that required authenticated attributes are missing when processing AuthEnvelopedData or AuthenticatedData.

* derEncodingNotUsed indicates that the content contained BER encoding, or some other encoding, where DER encoding was required.

* missingContentHints indicates that a SignedData content encapsulates a content other than a key package or an encrypted key package; however, the content-hints attribute [RFC2634] is not included. Similarly, this error code can indicate that the content-hints attribute was missing when processing AuthEnvelopedData or AuthenticatedData.

* invalidAttributeLocation indicates that an attribute appeared in an unacceptable location.

* badMessageDigest indicates that the value of the message-digest attribute [RFC5652] did not match the calculated value.

* badKeyPackage indicates that the SymmetricKeyPackage [RFC6031] or AsymmetricKeyPackage [RFC5958] syntax is invalid or that the version is unknown.

* badAttributes indicates that an attribute collection either contained multiple instances of the same attribute type that allows only one instance or contained an attribute instance with multiple values in an attribute that allows only one value.

* attributeComparisonFailure indicates that multiple instances of an attribute failed the comparison rules for the type of attribute.

* unsupportedSymmetricKeyPackage indicates that the implementation does not support symmetric key packages [RFC6031].

* unsupportedAsymmetricKeyPackage indicates that the implementation does not support asymmetric key packages [RFC5958].

* constraintViolation indicates that one or more of the attributes has a value that is not in the authorized set of values for the signer [RFC6010]. That is, the value is in conflict with the constraints imposed on the signer.

* ambiguousDefaultValue indicates that one or more of the attributes that is part of the signer's constraints is omitted from the key package, and the constraint permits more than one value; therefore, the appropriate default value for that attribute or attribute cannot be determined.

* noMatchingRecipientInfo indicates that a recipientInfo could not be found for the recipient. This can result from a ktri or kari found in EncryptedData, EnvelopedData, or AuthEnvelopedData.

* unsupportedKeyWrapAlgorithm indicates that the key wrap algorithm is not supported.

* badKeyTransRecipientInfo indicates that the KeyTransRecipientInfo syntax is invalid or the version is unknown or unsupported.

* other indicates that the key package could not be processed, but the reason is not covered by any of the assigned status codes. Use of this status code SHOULD be avoided.

The key package error content type MUST be signed if the entity generating it is capable of signing it. For example, a device will be incapable of signing when it is in early stages of deployment and it has not been configured with a private signing key or a device has an internal error that prevents use of its private signing key. When it is signed, the key package error MUST be encapsulated in a CMS SignedData content type to carry the signature of the party that is indicating an error. When it is encrypted, the key package error MUST be encapsulated in a CMS EnvelopedData content type, a CMS EncryptedData content type, or a CMS AuthEnvelopedData content type. When a key package error is signed and encrypted, it MUST be signed prior to being encrypted.

All devices that generate signed key package error reports MUST store their own certificate or have a means of obtaining the key identifier of their public key. If memory is a concern, the public key identifier can be computed from the public key.

If the error report signer has access to a real-time clock, then the binary-signing-time attribute [RFC6019] SHOULD be included in the key package error to provide the date and time when it was generated.

6. Protecting the KeyPackageReceipt and KeyPackageError

CMS protecting content types, [RFC5652] and [RFC5083], can be used to provide security to the KeyPackageReceipt and KeyPackageError content types:

- SignedData can be used to apply a digital signature.
- EncryptedData can be used to encrypt the content type with simple symmetric encryption, where the sender and the receiver already share the necessary encryption key.
- EnvelopedData can be used to encrypt the content type with symmetric encryption, where the sender and the receiver do not already share the necessary encryption key.
- AuthenticatedData can be used to integrity protect the content type with message authentication algorithms that support authenticated encryption, where key management information is handled in a manner similar to EnvelopedData.
- AuthEnvelopedData can be used to protect the content types with algorithms that support authenticated encryption, where key management information is handled in a manner similar to EnvelopedData.

7. Using the application/cms Media Type

The media type and parameters for carrying a key package receipt or a key package error content type are specified in [RFC7193].

8. IANA Considerations

IANA has updated the reference for the following registration in the "SMI Security for S/MIME Module Identifier (1.2.840.113549.1.9.16.0)" registry:

63 id-mod-keyPkgReceiptAndErrV2 [RFC7191]

9. Security Considerations

The key package receipt and key package error contents are not necessarily protected. These content types can be combined with a security protocol to protect the contents of the package.

The KeyPkgReceiptReq structure includes a receiptsFrom list and a receiptsTo list. Both lists contain SIREntityNames. The syntax does not specify a limit on the number of SIREntityNames that may be

included in either of these lists. In addition, there is purposefully no requirement that the receiptTo entries have any relation to the sender of the key package. To avoid these features being used as part of a denial-of-service amplification, receipts should only be returned for key packages with a valid signature from a trusted signer.

If an implementation is willing to accept key packages from more than one source, then there is a possibility that the same key package identifier could be used by more than one source. As a result, there is the potential for a receipt for one key package to be confused with the receipt for another, potentially leading to confusion about the keying material that is available to the recipient. In environments with multiple key sources, a convention for assignment of key package identifiers can avoid this potential confusion altogether.

In some situations, returning very detailed error information can provide an attacker with insight into the security processing. Where this is a concern, the implementation should return the most generic error code that is appropriate. However, detailed error codes are very helpful during development, debugging, and interoperability testing. For this reason, implementations may want to have a way to configure the use of a generic error code or a detailed one.

10. Acknowledgements

Many thanks to Radia Perlman, Sean Turner, Jim Schaad, and Carl Wallace for their insightful review. Thanks to Robert Sparks for improved wording.

11. References

11.1. Normative References

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

[RFC2634] Hoffman, P., Ed., "Enhanced Security Services for S/MIME", RFC 2634, June 1999.

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

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

[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.

[RFC6019] Housley, R., "BinaryTime: An Alternate Format for Representing Date and Time in ASN.1", RFC 6019, September 2010.

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

[RFC6032] Turner, S. and R. Housley, "Cryptographic Message Syntax (CMS) Encrypted Key Package Content Type", RFC 6032, December 2010.

[RFC6268] Schaad, J. and S. Turner, "Additional New ASN.1 Modules for the Cryptographic Message Syntax (CMS) and the Public Key Infrastructure Using X.509 (PKIX)", RFC 6268, July 2011.

[RFC7193] Turner, S., Housley, R., and J. Schaad, "The application/cms Media Type", RFC 7193, April 2014.

[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.

[X.690] ITU-T Recommendation X.690 (2002) | ISO/IEC 8825- 1:2002. Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).

11.2. Informative References

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

[RFC5934] Housley, R., Ashmore, S., and C. Wallace, "Trust Anchor Management Protocol (TAMP)", RFC 5934, August 2010.

Appendix A. ASN.1 Module

This annex provides the normative ASN.1 definitions for the structures described in this specification using ASN.1 as defined in [X.680], [X.681], [X.682], and [X.683].

```
KeyPackageReceiptAndErrorModuleV2
  { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
    smime(16) modules(0) id-mod-keyPkgReceiptAndErrV2(63) }

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

-- EXPORTS ALL

IMPORTS

-- FROM New SMIME ASN.1 [RFC6268]

CONTENT-TYPE
  FROM CryptographicMessageSyntax-2010
    { iso(1) member-body(2) us(840) rsadsi(113549)
      pkcs(1) pkcs-9(9) smime(16) modules(0) id-mod-cms-2009(58) }

-- From New PKIX ASN.1 [RFC5912]

ATTRIBUTE, SingleAttribute {}
  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) }

DistinguishedName
  FROM PKIX1Explicit-2009
    { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-pkix1-explicit-02(51)}
;

---
--- Key Package Version Number (revised from [RFC6031])
---

KeyPkgVersion ::= INTEGER  { v1(1), v2(2) } (1 .. 65535)
```

```
   --
   -- SIR Entity Name
   --

   SIREntityNames ::= SEQUENCE SIZE (1..MAX) OF SIREntityName

   SIREntityNameTypes SIR-ENTITY-NAME ::= {
       siren-dn,
       ... -- Expect additional SIR Entity Name types -- }

   SIR-ENTITY-NAME ::= CLASS {
       &sIRENType OBJECT IDENTIFIER UNIQUE,
       &SIRENValue
       } WITH SYNTAX {
       SYNTAX &SIRENValue IDENTIFIED BY &sIRENType }

   SIREntityName ::= SEQUENCE {
       sirenType      SIR-ENTITY-NAME.&sIRENType({SIREntityNameTypes}),
       sirenValue     OCTET STRING (CONTAINING
                        SIR-ENTITY-NAME.&SIRENValue(
                          {SIREntityNameTypes}{@sirenType}) ) }

   siren-dn SIR-ENTITY-NAME ::= {
       SYNTAX DistinguishedName
       IDENTIFIED BY id-dn }

   id-dn OBJECT IDENTIFIER ::= {
       joint-iso-ccitt(2) country(16) us(840) organization(1)
       gov(101) dod(2) infosec(1) sir-name-types(16) 0 }

   --
   -- Attribute Definitions
   --

   aa-keyPackageIdentifierAndReceiptRequest ATTRIBUTE ::= {
       TYPE KeyPkgIdentifierAndReceiptReq
       IDENTIFIED BY id-aa-KP-keyPkgIdAndReceiptReq }
   id-aa-KP-keyPkgIdAndReceiptReq OBJECT IDENTIFIER ::= {
       joint-iso-itu-t(2) country(16) us(840) organization(1)
       gov(101) dod(2) infosec(1) attributes(5) 65 }

   KeyPkgIdentifierAndReceiptReq ::= SEQUENCE {
       pkgID       KeyPkgID,
       receiptReq  KeyPkgReceiptReq OPTIONAL }

   KeyPkgID ::= OCTET STRING
```

```
KeyPkgReceiptReq ::= SEQUENCE {
    encryptReceipt     BOOLEAN DEFAULT FALSE,
    receiptsFrom   [0] SIREntityNames OPTIONAL,
    receiptsTo         SIREntityNames }

--
-- Content Type Definitions
--

KeyPackageContentTypes CONTENT-TYPE ::= {
  ct-key-package-receipt |
  ct-key-package-error,
  ... -- Expect additional content types -- }

-- Key Package Receipt CMS Content Type

ct-key-package-receipt CONTENT-TYPE ::= {
    TYPE KeyPackageReceipt
    IDENTIFIED BY id-ct-KP-keyPackageReceipt }

id-ct-KP-keyPackageReceipt 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) 3 }

KeyPackageReceipt ::= SEQUENCE {
    version          KeyPkgVersion DEFAULT v2,
    receiptOf        KeyPkgIdentifier,
    receivedBy       SIREntityName }

KeyPkgIdentifier ::= CHOICE {
    pkgID            KeyPkgID,
    attribute        SingleAttribute {{ KeyPkgIdentifiers }} }

KeyPkgIdentifiers ATTRIBUTE ::= { ... }

-- Key Package Receipt CMS Content Type

ct-key-package-error CONTENT-TYPE ::= {
    TYPE KeyPackageError IDENTIFIED BY id-ct-KP-keyPackageError }

id-ct-KP-keyPackageError 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) 6 }
```

```
KeyPackageError ::= SEQUENCE {
    version        KeyPkgVersion DEFAULT v2,
    errorOf    [0] KeyPkgIdentifier OPTIONAL,
    errorBy        SIREntityName,
    errorCode      ErrorCodeChoice }

ErrorCodeChoice ::= CHOICE {
    enum           EnumeratedErrorCode,
    oid            OBJECT IDENTIFIER }

EnumeratedErrorCode ::= ENUMERATED {
    decodeFailure                     (1),
    badContentInfo                    (2),
    badSignedData                     (3),
    badEncapContent                   (4),
    badCertificate                    (5),
    badSignerInfo                     (6),
    badSignedAttrs                    (7),
    badUnsignedAttrs                  (8),
    missingContent                    (9),
    noTrustAnchor                    (10),
    notAuthorized                    (11),
    badDigestAlgorithm               (12),
    badSignatureAlgorithm            (13),
    unsupportedKeySize               (14),
    unsupportedParameters            (15),
    signatureFailure                 (16),
    insufficientMemory               (17),
    incorrectTarget                  (23),
    missingSignature                 (29),
    resourcesBusy                    (30),
    versionNumberMismatch            (31),
    revokedCertificate               (33),

--  Error codes with values <= 33 are aligned with [RFC5934]

    ambiguousDecrypt                 (60),
    noDecryptKey                     (61),
    badEncryptedData                 (62),
    badEnvelopedData                 (63),
    badAuthenticatedData             (64),
    badAuthEnvelopedData             (65),
    badKeyAgreeRecipientInfo         (66),
    badKEKRecipientInfo              (67),
    badEncryptContent                (68),
    badEncryptAlgorithm              (69),
    missingCiphertext                (70),
    decryptFailure                   (71),
```

```
        badMACAlgorithm                 (72),
        badAuthAttrs                    (73),
        badUnauthAttrs                  (74),
        invalidMAC                      (75),
        mismatchedDigestAlg             (76),
        missingCertificate              (77),
        tooManySigners                  (78),
        missingSignedAttributes         (79),
        derEncodingNotUsed              (80),
        missingContentHints             (81),
        invalidAttributeLocation        (82),
        badMessageDigest                (83),
        badKeyPackage                   (84),
        badAttributes                   (85),
        attributeComparisonFailure      (86),
        unsupportedSymmetricKeyPackage  (87),
        unsupportedAsymmetricKeyPackage (88),
        constraintViolation             (89),
        ambiguousDefaultValue           (90),
        noMatchingRecipientInfo         (91),
        unsupportedKeyWrapAlgorithm     (92),
        badKeyTransRecipientInfo        (93),
        other                          (127),
        ... -- Expect additional error codes  -- }

   END
```

Author's Address

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

EMail: housley@vigilsec.com