NSA's Cryptographic Message Syntax (CMS) Key Management Attributes
draft-turner-km-attributes-07.txt

Abstract

   This document defines key management attributes used by the National
   Security Agency (NSA). The attributes can appear in asymmetric
   and/or symmetric key packages as well as the Cryptographic Message
   Syntax (CMS) content types that subsequently envelope the key
   packages. Key packages described in RFC 5958 and RFC 6031 are
   examples where these attributes can be used.

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."

described in the Simplified BSD License.

Table of Contents

1. Introduction

   This document defines key management attributes used by the National
   Security Agency (NSA). The attributes can appear in asymmetric
   and/or symmetric key packages as well as the Cryptographic Message
   Syntax (CMS) content types that subsequently envelope the key
   packages.

   This document contains definitions for new attributes as well as
   previously defined attributes. References are provided to the
   previously defined attributes however their definitions are included
   herein for convenience.

   CMS allows for arbitrary nesting of content types. Attributes are
   also supported in various locations in content types and key
   packages, which are themselves content types (see Section 1.1). An
   implementation that supports all of the possibilities would be
   extremely complex. Instead of implementing the full flexibility
   supported by this document, some devices may choose to support one or
   more templates, which is a profile for a combination of CMS content
   type(s), key package, and attribute(s) (see Section 19).

1.1. Attribute Locations

   There are a number of CMS content types that support attributes
   SignedData [RFC5652], EnvelopedData [RFC5652], EncryptedData
   [RFC5652], AuthenticatedData [RFC5652], and AuthEnvelopedData
   [RFC5083] as well as ContentWithAttributes [RFC4073]. There are also
   a number of other content types defined with CONTENT-TYPE [RFC6268]
   that support attributes including AsymmetricKeyPackage [RFC5958] and
   SymmetricKeyPackage [RFC6031].

   CMS defines a number of "protecting content types", SignedData
   [RFC5652], EnvelopedData [RFC5652], EncryptedData [RFC5652],
   AuthenticatedData [RFC5652], and AuthEnvelopedData [RFC5083], that
   provide some type of security service. There are also other CMS
   content types, Data [RFC5652], ContentWithAttributes [RFC4073], and
   ContentCollection [RFC4073] that provide no security service.

   There are also different kinds of attributes in these content types:

     o SignedData supports two kinds of attributes: signed and unsigned
       attributes in the signedAttrs and unsignedAttrs fields,
       respectively.

      o EnvelopedData and EncryptedData each support one kind of
        attribute: unprotected attributes in the unprotectedAttrs field.

      o AuthEnvelopedData supports two kinds of attributes: authenticated
        and unauthenticated attributes in the authAttrs and unauthAttrs
        fields, respectively. Both of these attributes are also
        unprotected (i.e., they are not encrypted); therefore, when
        referring to AuthEnvelopedData attributes they are
        authenticated/unprotected and unauthenticated/unprotected. For
        this specification, unauthenticated attributes MUST NOT be
        included.

        o AuthenticatedData supports two kinds of attributes:
        authenticated and unauthenticated attributes in the authAttrs and
        unauthAttrs fields, respectively. For this specification,
        unauthenticated attributes MUST NOT be included.

      o ContentWithAttributes supports one kind of attribute: content
        attributes in the attrs field.

      o AsymmetricKeyPackage supports one kind of attribute: asymmetric
        key attributes in the attributes field. If an attribute appears
        as part of an asymmetric key package, it SHOULD appear in the
        attributes field of the AsymmetricKeyPackage.

      o SymmetricKeyPackage supports two kinds of attributes: symmetric
        key and symmetric key package attributes in the sKeyAttrs and
        sKeyPkgAttrs fields, respectively. Note that [RFC6031] prohibits
        the same attribute from appearing in both locations in the same
        SymmetricKeyPackage.

   Note that this specification updates the following information object
   sets SignedAttributesSet, UnsignedAttributes,
   UnprotectedEnvAttributes, UnprotectedEncAttributes, AuthAttributeSet,
   UnauthAttributeSet, AuthEnvDataAttributeSet,
   UnauthEnvDataAttributeSet, and ContentAttributeSet from [RFC6268] as
   well as OneAsymmetricKeyAttributes from [RFC5958], SKeyPkgAttributes
   from [RFC6031], SKeyAttributes from [RFC6031] to constrain the
   permissible locations for attributes. See Appendix A for the ASN.1
   for the information object sets.

1.2. ASN.1 Notation

   The attributes defined in this document use 2002 ASN.1
   [X.680][X.681][X.682][X.683]. The attributes MUST be DER [X.690]
   encoded.

   Each of the attributes has a single attribute value instance in the

values set. Even though the syntax is defined as a set, there MUST
be exactly one instance of AttributeValue present. Further, the
SignedAttributes, UnsignedAttributes, UnprotectedAttributes,
AuthAttributes, and UnauthAttributes are also defined as a set, and
this set MUST include only one instance of any particular type of
attribute. That is, any object identifier appearing in AttributeType
MUST only appear one time in the set of attributes.

SignedData, EnvelopedData, EncryptedData, AuthenticatedData,
AuthEnvelopedData, and ContentWithAttributes were originally defined
using the 1988 version of ASN.1. These definitions were updated to
the 2008 version of ASN.1 by [RFC6268]. None of the new 2008 ASN.1
tokens are used, which allows 2002 compilers to compile 2008 ASN.1.
AsymmetricKeyPackage and SymmetricKeyPackage are defined using the
2002 ASN.1.

[RFC5652] and [RFC2634] define generally useful attributes for CMS
using the 1988 version of ASN.1. These definitions were updated to
the 2008 version of ASN.1 by [RFC6268] and the 2002 version of ASN.1
by [RFC5911], respectively. [RFC4108] and [RFC6019] also defined
attributes using the 1988 version of ASN.1, which this document uses.
 Both were updated by [RFC5911] to the 2002 ASN.1. Refer to
[RFC2634], [RFC4108], [RFC5652], and [RFC6019] for the attribute's
semantics but refer to [RFC5911] or [RFC6268] for the attribute's
ASN.1 syntax.

1.3. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and
"OPTIONAL" in this document are to be interpreted as described in RFC
2119 [RFC2119].

Attribute Scope: The scope of an attribute is the compilation of
keying material to which the attribute value is assigned. The scope
of each attribute is determined by its placement within the key
package or content collection. See Section 31.

SIR: Sender-Intermediary-Receiver is a model with three entities:

  o A sender initiates the delivery of a key to one or more
    receivers. It may wrap or encrypt the key for delivery. This is
    expected to be the common case, since cleartext key is vulnerable
    to exposure and compromise. If the sender is to encrypt the key
    for delivery, it must know how to encrypt the key so that the
    receiver(s) can decrypt it. A sender may also carry out any of
    the functions of an intermediary.

* The original key package creators are sometimes referred to as key source authorities. These entities create the symmetric and/or asymmetric key package and apply the initial CMS protecting layer, which is normally a SignedData but sometimes an AuthenticatedData. This initial CMS protecting layer is maintained through any intermediary for the receivers of the key package to ensure that receivers can validate the key source authority.

o An intermediary does not have access to cleartext key. An intermediary may perform source authentication on key packages, and may append or remove management information related to the package. It may encapsulate the encrypted key packages in larger packages that contain other user data destined for later intermediaries or receivers.

o A receiver has access to cleartext key. If the received key package is encrypted, it can unwrap or decrypt the encrypted key to obtain the cleartext key. A receiver may be the final destination of the cryptographic product. An element that acts as a receiver and is not the final destination of the key package may also act as a sender or as an intermediary. After receiving a key, a receiver may encrypt the received key for local storage.

NOTE: As noted in Section 1, a receiver can be tailored to support a particular combination of CMS content type(s), key package, and attribute(s) resulting in less complex implementations. All of these tailored receivers can be supported by a common key management infrastructure that uses this specification, which also can yield efficiencies in generation and provisioning. Senders and intermediaries that have to understand multiple tailored receivers get the efficiency of a common specification language and modular implementation, as opposed to needing stove-piped processing for each different receiver.

2. CMS-Defined Attributes

The following attributes are defined for [RFC5652]:

o content-type [RFC5652][RFC5911][RFC6268] uniquely specifies the CMS content type. This attribute MUST be included as a signed, authenticated, or authenticated/unprotected attribute.

o message-digest [RFC5652][RFC5911][RFC6268] is the message digest of the encapsulated content calculated using the signer's message digest algorithm. As specified in [RFC5652], it must be included as a signed attribute and an authenticated attribute; as specified in [RFC5652], it must not be an unsigned attribute,

unauthenticated attribute, or unprotected attribute; as specified
in [RFC5083], it should not be included as an
authenticated/unprotected attribute in AuthEnvelopedData. This
attribute MUST NOT be included elsewhere.

o content-hints [RFC2634][RFC5911][RFC6268] identifies the
innermost content when multiple layers of encapsulation have been
applied. Every instance of SignedData, AuthenticatedData, and
AuthEnvelopedData that does not directly encapsulate a
SymmetricKeyPackage, an AsymmetricKeyPackage, or an
EncryptedKeyPackage [RFC6032] MUST include this attribute.

3. Community Identifiers

The community-identifiers attribute, defined in [RFC4108][RFC5911],
lists the communities that are authorized recipients of the signed
content. It can appear as a signed, authenticated,
authenticated/unprotected, or content attribute. This attribute MUST
be supported.

The 2002 ASN.1 syntax for the community-identifier attribute is
included for convenience:

```
aa-communityIdentifiers ATTRIBUTE ::= {
  TYPE CommunityIdentifiers
  IDENTIFIED BY id-aa-communityIdentifiers }

id-aa-communityIdentifiers OBJECT IDENTIFIER ::= {
  iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
  smime(16) aa(2) 40 }

CommunityIdentifiers ::= SEQUENCE OF CommunityIdentifier

CommunityIdentifier ::= CHOICE {
  communityOID  OBJECT IDENTIFIER,
  hwModuleList  HardwareModules }

HardwareModules ::= SEQUENCE {
  hwType          OBJECT IDENTIFIER,
  hwSerialEntries  SEQUENCE OF HardwareSerialEntry }

HardwareSerialEntry ::= CHOICE {
  all    NULL,
  single OCTET STRING,
  block  SEQUENCE {
          low OCTET STRING,
          high OCTET STRING } }
```

Consult [RFC4108] for the attribute's semantics.

4. Key Province Attribute

The key-province-v2 attribute identifies the scope, range, or
jurisdiction in which the key is to be used. The key-province-v2
attribute MUST be present as a signed attribute or an authenticated
attribute in the innermost CMS protection content type that provides
authentication (i.e., SignedData, AuthEnvelopedData, or
AuthenticatedData) and encapsulates a symmetric key package or an
asymmetric key package.

The key-province attribute has the following syntax:

```
aa-keyProvince-v2 ATTRIBUTE ::= {
  TYPE KeyProvinceV2
  IDENTIFIED BY id-aa-KP-keyProvinceV2 }

id-aa-KP-keyProvinceV2 OBJECT IDENTIFIER ::=
  { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
    dod(2) infosec(1) attributes(5) 71 }

KeyProvinceV2 ::= OBJECT IDENTIFIER
```

5. Binary Signing Time

The binary-signing-time attribute, defined in [RFC6019][RFC6268],
specifies the time at which the signature or the message
authentication code was applied to the encapsulated content. It can
appear as a signed, authenticated, or authenticated/unprotected
attribute.

The 2002 ASN.1 syntax is included for convenience:

```
aa-binarySigningTime ATTRIBUTE ::= {
  TYPE BinarySigningTime
  IDENTIFIED BY id-aa-binarySigningTime }

id-aa-binarySigningTime OBJECT IDENTIFIER ::= {
  iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
  smime(16) aa(2) 46 }

BinarySigningTime ::= BinaryTime

BinaryTime ::= INTEGER (0..MAX)
```

Consult [RFC6019] for the binary-signing-time attribute's semantics.

6. **Manifest**

   The manifest attribute lists the short titles of all the Transmission
   Security Nomenclature (TSEC-Nomenclature) attributes from inner key
   packages. It MUST only appear as an outer-most signed,
   authenticated, or authenticated/unprotected attribute. If a short
   title is repeated in inner packages, it need only appear once in the
   manifest attribute. The manifest attribute MUST NOT appear in the
   same level as the TSEC-Nomenclature from the Section 10.

   The manifest attribute has the following syntax:

      aa-manifest ATTRIBUTE ::= {
        TYPE Manifest
        IDENTIFIED BY id-aa-KP-manifest }

      id-aa-KP-manifest OBJECT IDENTIFIER ::= {
        joint-iso-itu-t(2) country(16) us(840) organization(1)
        gov(101) dod(2) infosec(1) attributes(5) 72 }

      Manifest ::= SEQUENCE SIZE (1..MAX) OF ShortTitle

7. **Key Algorithm**

   The key-algorithm attribute indirectly specifies the size and format
   of the keying material in the skey field of a symmetric key package,
   which is defined in [RFC6031]. It can appear as a symmetric key,
   symmetric key package, signed, authenticated,
   authenticated/unprotected, or content attribute. If this attribute
   appears as a signed attribute, then all of the keying material within
   the SignedData content MUST be associated with the same algorithm.
   If this attribute appears as an authenticated or
   authenticated/unprotected attribute, then all of the keying material
   within the AuthenticatedData or AuthEnvelopedData content type MUST
   be associated with the same algorithm. If this attribute appears as
   a content attribute, then all of the keying material within the
   collection MUST be associated with the same algorithm. If both the
   key-algorithm and key-wrap-algorithm attributes apply from the
   Section 24 to an sKey, then the key-algorithm attribute refers to the
   decrypted value of sKey rather than to the content of sKey itself.
   This attribute MUST be supported.

   The key-algorithm attribute has the following syntax:

      aa-keyAlgorithm ATTRIBUTE ::= {
        TYPE KeyAlgorithm
        IDENTIFIED BY id-kma-keyAlgorithm }

```
   id-kma-keyAlgorithm OBJECT IDENTIFIER ::= {
     joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
     dod(2) infosec(1) keying-material-attributes(13) 1 }

   KeyAlgorithm ::= SEQUENCE {
     keyAlg             OBJECT IDENTIFIER,
     checkWordAlg  [1] OBJECT IDENTIFIER OPTIONAL,
     crcAlg        [2] OBJECT IDENTIFIER OPTIONAL }
```

The fields in the key-algorithm attribute have the following
semantics:

  o keyAlg specifies the size and format of the keying material.

  o If the particular key format supports more than one check word
    algorithm, then the OPTIONAL checkWordAlg identifier indicates
    which check word algorithm was used to generate the check word
    that is present. If the check word algorithm is implied by the
    key algorithm, then the checkWordAlg field SHOULD be omitted.

  o If the particular key format supports more than one Cyclic
    Redundancy Check (CRC) algorithm, then the OPTIONAL crcAlg
    identifier indicates which CRC algorithm was used to generate the
    value that is present. If the CRC algorithm is implied by the
    key algorithm, then the crcAlg field SHOULD be omitted.

The keyAlg identifier, the checkWordAlg identifier, and the crcAlg
identifier are object identifiers. The use of an object identifier
accommodates any algorithm from any registry.

The format of the keying material in the skey field of a symmetric
key package will not match this attribute if the keying material is
split (see section 18 for a discussion on the split-identifier
attribute). In this situation, this attribute identifies the format
of the keying material once the two splits are combined.

Due to multiple layers of encapsulation or the use of content
collections, the key-algorithm attribute can appear in more than one
location in the overall key package. When there are multiple
occurrences of the key-algorithm attribute within the same scope, the
keyAlg field MUST match in all instances. The OPTIONAL checkWordAlg
and crcAlg fields can be omitted in the key-algorithm attribute when
it appears as a signed, authenticated, authenticated/unprotected, or
content attribute. However, if these optional fields are present,
they MUST also match the other occurrences within the same scope.
Receivers MUST reject any key package that fails these consistency
checks.

8. User Certificate

The user-certificate attribute specifies the type, format, and value
of an X.509 certificate and is used in asymmetric key package's
attributes field. This attribute can appear as an asymmetric key
attribute. This attribute MUST NOT appear in an asymmetric key
package attributes field that includes the other-certificate-formats
attribute. Symmetric key packages do not contain any certificates,
so the user-certificate attribute MUST NOT appear in a symmetric key
package. The user-certificate attribute MUST NOT appear as a signed,
authenticated, authenticated/unprotected, or content attribute. This
attribute MUST be supported.

The syntax is taken from [X.509] but redefined using the ATTRIBUTE
CLASS from [RFC5911]. The user-certificate attribute has the
following syntax:

```
aa-userCertificate ATTRIBUTE ::= {
  TYPE Certificate
  EQUALITY MATCHING RULE certificateExactMatch
  IDENTIFIED BY id-at-userCertificate }

id-at-userCertificate OBJECT IDENTIFIER ::= {
  joint-iso-itu-t(2) ds(5) attributes(4) 36 }
```

Since the user-certificate attribute MUST NOT appear as a signed,
authenticated, authenticated/unprotected, or content attribute, an
asymmetric key package cannot include multiple occurrences of the
user-certificate attribute within the same scope. Receivers MUST
reject any asymmetric key package in which the user-certificate
attribute appears as a signed, authenticated,
authenticated/unprotected, or content attribute.

9. Key Package Receivers

The key-package-receivers-v2 attribute indicates the intended
audience for the key package. The key-package-receivers-v2 attribute
is not intended for access control decisions, rather intermediate
systems may use this attribute to make routing and relaying
decisions. If the receiver is not listed, it will not be able to
decrypt the package, therefore the receiver SHOULD reject the key
package if the key-package-receivers-v2 attribute is present and they
are not listed as an intended receiver. The key-package-receivers-v2
attribute can be used as a signed, authenticated,
authenticated/unprotected, or content attribute. If key-package-
receivers-v2 attribute is associated with a collection, then the
named receivers MUST be able to receive all of the key packages
within the collection. This attribute MUST be supported.

The key-package-receivers-v2 attribute has the following syntax:

```
aa-keyPackageReceivers-v2 ATTRIBUTE ::= {
  TYPE KeyPkgReceiversV2
  IDENTIFIED BY id-kma-keyPkgReceiversV2 }

id-kma-keyPkgReceiversV2 OBJECT IDENTIFIER ::= {
  joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
  dod(2) infosec(1) keying-material-attributes(13) 16 }

KeyPkgReceiversV2 ::= SEQUENCE SIZE (1..MAX) OF KeyPkgReceiver

KeyPkgReceiver ::= CHOICE {
  sirEntity  [0] SIREntityName,
  community  [1] CommunityIdentifier }
```

The key-package-receivers-v2 attribute contains a list of receiver
identifiers. The receiver identifier is either a SIREntityName
[RFC7191] or a CommunityIdentifier (see Section 3). The
SIREntityName syntax does not impose any particular structure on the
receiver identifier, but it does require registration of receiver
identifier types. The nameType ensures that two receiver identifiers
of different types that contain the same values are not interpreted
as equivalent. Name types are expected to be defined that represent
several different granularities. For example, one name type will
represent the receiver organization. At a finer granularity, the
name type will identify a specific cryptographic device, perhaps
using a manufacturer identifier and serial number.

If a receiver does not recognize a particular nameType or a community
identifier, then keying material within the scope of the unrecognized
nameType or community identifier MUST NOT be used in any manner.
However, the receiver need not discard the associated key package.
Since many cryptographic devices are programmable, a different
firmware load may recognize the nameType. Likewise, a change in the
configuration may lead to the recognition of a previously
unrecognized community identifier. Therefore, the receiver may
retain the key package, but refuse to use it for anything with a
firmware load that does not recognize the nameType or a configuration
that does not recognize the community identifier.

Whenever a key package is saved for later processing due to an
unrecognized nameType or community identifier, subsequent processing
MUST NOT rely on any checks that were made the first time the key
package processing was attempted. That is, the subsequent processing
MUST include the full complement of checks. Further, a receipt for
the packages MUST NOT be generated unless all of these checks are
successfully completed.

Due to multiple layers of encapsulation or the use of content
collections, the key-package-receivers-v2 attribute can appear in
more than one location in the overall key package. When there are
multiple occurrences of the key-package-receivers-v2 attribute, each
occurrence is evaluated independently.

In a content collection, each member of the collection might contain
its own signed, authenticated, authenticated/unprotected, or content
attribute that includes a key-package-receivers-v2 attribute. In
this situation, each member of the collection is evaluated
separately, and any member that includes an acceptable receiver
SHOULD be retained. Other members can be rejected or retained for
later processing with a different firmware load.

10. TSEC Nomenclature

The Telecommunications Security Nomenclature (TSEC-Nomenclature)
attribute provides the name for a piece of keying material, which
always includes a printable string called a "short title" (see
below). The TSEC-Nomenclature attribute also contains other
identifiers when the shortTitle is insufficient to uniquely name a
particular piece of keying material. This attribute can appear as a
symmetric key, symmetric key package, asymmetric key, signed,
authenticated, authenticated/unprotected, or content attribute. If
this attribute appears in the sKeyAttrs field, the editionID,
registerID, and segmentID attribute fields MUST NOT be ranges. If
this attribute appears as a signed, authenticated,
authenticated/unprotected, or content attribute, all of the keying
material within the associated content MUST have the same shortTitle,
and the attribute value MUST contain only a shortTitle. That is,
when this attribute appears as a signed, authenticated,
authenticated/unprotected, or content attribute, all of the optional
fields MUST be absent. If this attribute is associated with a
collection, all of the keying material within the collection MUST
have the same shortTitle; however, the editionID, registerID, and
segmentID will be different for each key package in the collection.
This attribute MUST be supported.

The TSEC-Nomenclature attribute has the following syntax:

```
aa-tsecNomenclature ATTRIBUTE ::= {
  TYPE TSECNomenclature
  IDENTIFIED BY id-kma-TSECNomenclature }

id-kma-TSECNomenclature OBJECT IDENTIFIER ::= {
  joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
  dod(2) infosec(1) keying-material-attributes(13) 3 }
```

```
     TSECNomenclature ::= SEQUENCE {
       shortTitle  ShortTitle,
       editionID   EditionID OPTIONAL,
       registerID  RegisterID OPTIONAL,
       segmentID   SegmentID OPTIONAL }

     ShortTitle ::= PrintableString

     EditionID ::= CHOICE {
       char CHOICE {
           charEdition       [1] CharEdition,
           charEditionRange  [2] CharEditionRange }
       num CHOICE {
           numEdition        [3] NumEdition,
           numEditionRange    [4] NumEditionRange } }

     CharEdition ::= PrintableString

     CharEditionRange ::= SEQUENCE {
       firstCharEdition  CharEdition,
       lastCharEdition   CharEdition }

     NumEdition ::= INTEGER (0..308915776)

     NumEditionRange ::= SEQUENCE {
       firstNumEdition  NumEdition,
       lastNumEdition   NumEdition }

     RegisterID ::= CHOICE {
       register       [5] Register,
       registerRange  [6] RegisterRange }

     Register ::= INTEGER (0..2147483647)

     RegisterRange ::= SEQUENCE {
       firstRegister  Register,
       lastRegister   Register }

     SegmentID ::= CHOICE {
       segmentNumber [7] SegmentNumber,
       segmentRange   [8] SegmentRange }

     SegmentNumber ::= INTEGER (1..127)

     SegmentRange ::= SEQUENCE {
       firstSegment  SegmentNumber,
       lastSegment   SegmentNumber }
```

The fields in the TSEC-Nomenclature attribute have the following
semantics:

   o The shortTitle consists of up to 32 alphanumeric characters.
     shortTitle processing always uses the value in its entirety.

   o The editionID is OPTIONAL, and the editionIdentifier is used to
     distinguish accountable items. The editionID consists either of
     six alphanumeric characters or an integer. When present, the
     editionID is either a single value or a range. The integer
     encoding should be used when it is important to keep key package
     size to a minimum.

   o The registerID is OPTIONAL. For electronic keying material, the
     registerID is usually omitted. The registerID is an accounting
     number assigned to identify COMSEC material. The registerID is
     either a single value or a range.

   o The segmentID is OPTIONAL, and it distinguishes the individual
     symmetric keys delivered in one edition. A unique segmentNumber
     is assigned to each key in an edition. The segmentNumber is set
     to one for the first item in each edition, and it is incremented
     by one for each additional item within that edition. The
     segmentID is either a single value or a range.

The order that the keying material will appear in the key package is
illustrated by the following example: a cryptographic device may
require fresh keying material every day, an edition represents the
keying material for a single month, and the segments represent the
keying material for a day within that month. Consider a key package
that contains the keying material for July and August; it will
contain keying material for 62 days. The keying material will appear
in the following order: Edition 1, Segment 1; Edition 1, Segment 2;
Edition 1, Segment 3; ...; Edition 1, Segment 31; Edition 2, Segment
1; Edition 2, Segment 2; Edition 2, Segment 3; ...; Edition 2,
Segment 31.

Due to multiple layers of encapsulation or the use of content
collections, the TSEC-Nomenclature attribute can appear in more than
one location in the overall key package. When there are multiple
occurrences of the TSEC-Nomenclature attribute within the same scope,
the shortTitle field MUST match in all instances. Receivers MUST
reject any key package that fails these consistency checks.

When the manifest attribute from Section 6 is included in an outer
layer, the ShortTitle field values present in TSEC-Nomenclature
attributes MUST be one of the values in the manifest attribute.
Receivers MUST reject any key package that fails this consistency

   checks.

11. Key Purpose

   The key-purpose attribute specifies the intended purpose of the key
   material. It can appear as a symmetric key, symmetric key package,
   asymmetric key, signed, authenticated, authenticated/unprotected, or
   content attribute. If the key-purpose attribute appears as a signed,
   authenticated, authenticated/unprotected, or content attribute, then
   all of the keying material within the associated content MUST have
   the same key purpose value.

   The key-purpose attribute has the following syntax:

```
     aa-keyPurpose ATTRIBUTE ::= {
       TYPE KeyPurpose
       IDENTIFIED BY id-kma-keyPurpose }

     id-kma-keyPurpose OBJECT IDENTIFIER ::= {
       joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
       dod(2) infosec(1) keying-material-attributes(13) 13 }

     KeyPurpose ::= ENUMERATED {
       n-a    (0),    -- Not Applicable
       A      (65),   -- Operational
       B      (66),   -- Compatible Multiple Key
       L      (76),   -- Logistics Combinations
       M      (77),   -- Maintenance
       R      (82),   -- Reference
       S      (83),   -- Sample
       T      (84),   -- Training
       V      (86),   -- Developmental
       X      (88),   -- Exercise
       Z      (90),   -- "On the Air" Testing
       ... -- Expect additional key purpose values -- }
```

   Due to multiple layers of encapsulation or the use of content
   collections, the key-purpose attribute can appear in more than one
   location in the overall key package. When there are multiple
   occurrences of the key-purpose attribute within the same scope, all
   fields within the attribute MUST contain exactly the same values.
   Receivers MUST reject any key package that fails these consistency
   checks.

12. Key Use

   The key-use attribute specifies the intended use of the key material.
    It can appear as a symmetric key, symmetric key package, asymmetric,

      signed, authenticated, authenticated/unprotected, or content
      attribute. If the key-use attribute appears as a signed,
      authenticated, authenticated/unprotected, or content attribute, then
      all of the keying material within the associated content MUST have
      the same key use value.

   The key-use attribute has the following syntax:

      aa-key-Use ATTRIBUTE ::= {
        TYPE KeyUse
        IDENTIFIED BY id-kma-keyUse }

      id-kma-keyUse OBJECT IDENTIFIER ::= {
        joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
        dod(2) infosec(1) keying-material-attributes(13) 14 }

      KeyUse ::= ENUMERATED {
        n-a   (0),   -- Not applicable
        ffk   (1),   -- FIREFLY/CROSSTALK Key (Basic Format)
        kek   (2),   -- Key Encryption Key
        kpk   (3),   -- Key Production Key
        msk   (4),   -- Message Signature Key
        qkek   (5),   -- QUADRANT Key Encryption Key
        tek   (6),   -- Traffic Encryption Key
        tsk   (7),   -- Transmission Security Key
        trkek (8),   -- Transfer Key Encryption Key
        nfk   (9),   -- Netted FIREFLY Key
        effk   (10),   -- FIREFLY Key (Enhanced Format)
        ebfk   (11),   -- FIREFLY Key (Enhanceable Basic Format)
        aek   (12),   -- Algorithm Encryption Key
        wod   (13),   -- Word of Day
        kesk (246),   -- Key Establishment Key
        eik   (247),   -- Entity Identification Key
        ask   (248),   -- Authority Signature Key
        kmk   (249),   -- Key Modifier Key
        rsk   (250),   -- Revocation Signature Key
        csk   (251),   -- Certificate Signature Key
        sak   (252),   -- Symmetric Authentication Key
        rgk   (253),   -- Random Generation Key
        cek   (254),   -- Certificate Encryption Key
        exk   (255),   -- Exclusion Key
        ... -- Expect additional key use values -- }

   The values for the key-use attribute has the following semantics:

      o ffk: A FIREFLY/CROSSTALK key is used to establish a Key
        Establishment Key (KEK) or a Transmission Encryption Key (TEK)
        between two parties. The KEK or TEK generated from the exchange

   is used with a symmetric encryption algorithm. This key use
   value is associated with keys in the basic format.

   o kek: A Key Encryption Key is used to encrypt or decrypt other
     keys for transmission or storage.

   o kpk: A Key Production Key is used to initialize a keystream
     generator for the production of other electronically generated
     keys.

   o msk: A Message Signature Key is used in a digital signature
     process that operates on a message to assure message source
     authentication, message integrity, and non-repudiation.

   o qkek: QUADRANT Key Encryption Key is one part of a tamper
     resistance solution.

   o tek: A Traffic Encryption Key is used to encrypt plaintext or to
     superencrypt previously encrypted data and/or to decrypt
     ciphertext.

   o tsk: A Transmission Security Key is used to protect transmissions
     from interception and exploitation by means other than
     cryptanalysis.

   o trkek: Transfer Key Encryption Key. For example, the keys used
     by the KP and DTD.

   o nfk: A Netted FIREFLY Key is a FIREFLY key that has an edition
     number associated with it. When rekeyed, it is incremented,
     preventing communications with FIREFLY key of previous editions.
     This edition number is maintained within a universal edition.

   o effk: Enhanced FIREFLY Key is used to establish a KEK or a TEK
     between two parties. The KEK or TEK generated from an exchange
     is used with a symmetric encryption algorithm. This key use
     value is associated with keys in the enhanced format.

   o ebfk: Enhanceable Basic FIREFLY Key is used to establish a KEK or
     a TEK between two parties. The KEK or TEK generated from an
     exchange is used with a symmetric encryption algorithm. This key
     use value is associated with keys in the enhanceable basic
     format.

   o aek: An Algorithm Encryption Key is used to encrypt or decrypt an
     algorithm implementation as well as other functionality in the
     implementation.

   o wod: A key used to generate the Word of the Day (WOD).

   o kek: A Key Establishment Key is an asymmetric key set (e.g.
     public/private/parameters) used to enable the establishment of
     symmetric key(s) between entities.

   o eik: An Entity Identification Key is an asymmetric key set (e.g.
     public/private/parameters) used to identify one entity to another
     for access control and other similar purposes.

   o ask: An Authority Signature Key is an asymmetric key set (e.g.
     public/private/parameters) used by designated authorities to sign
     objects such as TAMP messages and firmware packages.

   o kmk: A Key Modifier Key is a symmetric key used to modify the
     results of the process that forms a symmetric key from a public
     key exchange process.

   o rsk: A Revocation Signature Key is an asymmetric key set (e.g.
     public/private/parameters) used to sign and authenticate
     revocation lists and compromised key lists.

   o csk: A Certificate Signature Key is an asymmetric key set (e.g.
     public/private/parameters) used to sign and authenticate public-
     key certificates.

   o sak: A Symmetric Authentication Key is used in a Message
     Authentication Code (MAC) algorithm to provide message integrity.
      Differs from a Message Signature Key in that it is symmetric key
     material and it does not provide source authentication or non-
     repudiation.

   o rgk: Random Generation Key is a key used to seed a deterministic
     pseudo-random number generator.

   o cek: A Certificate Encryption Key is used to encrypt public-key
     certificates to support privacy.

   o exk: An Exclusion Key is a symmetric key used to
     cryptographically subdivide a single large security domain into
     smaller segregated domains.

   Due to multiple layers of encapsulation or the use of content
   collections, the key-use attribute can appear in more than one
   location in the overall key package. When there are multiple
   occurrences of the key-use attribute within the same scope, all
   fields within the attribute MUST contain exactly the same values.
   Receivers MUST reject any key package that fails these consistency

   checks.

13. **Transport Key**

   The transport-key attribute identifies whether an asymmetric key is a
   transport key or an operational key (i.e., the key can either be used
   as is or not). It can appear as an asymmetric key, signed,
   authenticated, authenticated/unprotected, or content attribute. If
   the transport-key attribute appears as a signed, authenticated,
   authenticated/unprotected, or content attribute, then all of the
   keying material within the associated content MUST have the same
   operational/transport key material.

```
     aa-transportKey ATTRIBUTE ::= {
       TYPE TransOp
       IDENTIFIED BY id-kma-transportKey }

     id-kma-transportKey OBJECT IDENTIFIER ::= {
       joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
       dod(2) infosec(1) keying-material-attributes(13) 15 }

     TransOp ::= ENUMERATED {
       transport    (1),
       operational  (2) }
```

   Due to multiple layers of encapsulation or the use of content
   collections, the transport-key attribute can appear in more than one
   location in the overall key package. When there are multiple
   occurrences of the transport-key attribute within the same scope, all
   fields within the attribute MUST contain exactly the same values.
   Receivers MUST reject any key package that fails these consistency
   checks.

14. **Key Distribution Period**

   The key-distribution-period attribute indicates the period of time
   that the keying material is intended for distribution. Keying
   material is often distributed before it is intended to be used. Time
   of day must be represented in Coordinated Universal Time (UTC). It
   can appear as a symmetric key, symmetric key package, asymmetric key,
   signed, authenticated, authenticated/unprotected, or content
   attribute. If the key-distribution-period attribute appears as a
   signed, authenticated, authenticated/unprotected, or content
   attribute, then all of the keying material within the content MUST
   have the same key distribution period.

   The key-distribution-period attribute has the following syntax:

```
   aa-keyDistributionPeriod ATTRIBUTE ::= {
     TYPE KeyDistPeriod
     IDENTIFIED BY id-kma-keyDistPeriod }

   id-kma-keyDistPeriod OBJECT IDENTIFIER ::= {
     joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
     dod(2) infosec(1) keying-material-attributes(13) 5 }

   KeyDistPeriod ::= SEQUENCE {
     doNotDistBefore  [0] BinaryTime OPTIONAL,
     doNotDistAfter       BinaryTime }

   BinaryTime ::= INTEGER
```

The fields in the key-distribution-period attribute have the
following semantics:

   o The doNotDistBefore field is OPTIONAL, and when it is present,
     the keying material SHOULD NOT be distributed before the date and
     time provided.

   o The doNotDistAfter field is REQUIRED, and the keying material
     SHOULD NOT be distributed after the date and time provided.

When the key-distribution-period attribute is associated with a
collection of keying material, the distribution period applies to all
of the keys in the collection. None of the keying material in the
collection SHOULD be distributed outside the indicated period.

Due to multiple layers of encapsulation or the use of content
collections, the key-distribution-period attribute can appear in more
than one location in the overall key package. When there are
multiple occurrences of the key-distribution-period attribute within
the same scope, all of the included attribute fields MUST contain
exactly the same value. However, if the doNotDistBefore field is
absent in an inner layer, a value MAY appear in an outer layer
because the outer layer constrains the inner layer. Receivers MUST
reject any key package that fails these consistency checks.

15. Key Validity Period

The key-validity-period attribute indicates the period of time that
the keying material is intended for use. Time of day MUST be
represented in Coordinated Universal Time (UTC). It can appear as a
symmetric key, symmetric key package, asymmetric key, signed,
authenticated, authenticated/unprotected, or content attribute. If
the key-validity-period attribute appears as a signed, authenticated,
authenticated/unprotected, or content attribute, then all of the

keying material within the content MUST have the same key validity
period.

The key-validity-period attribute has the following syntax:

```
aa-keyValidityPeriod ATTRIBUTE ::= {
  TYPE KeyValidityPeriod
  IDENTIFIED BY id-kma-keyValidityPeriod }

id-kma-keyValidityPeriod OBJECT IDENTIFIER ::= {
  joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
  dod(2) infosec(1) keying-material-attributes(13) 6 }

KeyValidityPeriod ::= SEQUENCE {
  doNotUseBefore      BinaryTime,
  doNotUseAfter       BinaryTime OPTIONAL }

BinaryTime ::= INTEGER
```

The fields in the key-validity-period attribute have the following
semantics:

   o The doNotUseBefore field is required, and the keying material
     should not be used before the date and time provided.

   o The doNotUseAfter field is optional, and when it is present, the
     keying material should not be used after the date and time
     provided.

For a key package that is being used for rekey, the doNotUseAfter
field MAY be required by some templates even though the syntax is
OPTIONAL.

When the key-validity-period attribute is associated with a
collection of keying material, the validity period applies to all of
the keys in the collection. None of the keying material in the
collection SHOULD be used outside the indicated period.

The key-validity-period attribute described in this section and the
key-duration attribute described in the next section provide a
complementary function. The key-validity-period attribute provides
explicit date and time values, which indicate the beginning and
ending of the keying material usage period. The key-duration
attribute provides the maximum length of time that the keying
material SHOULD be used. If both attributes are provided, this
duration MAY occur at any time within the specified period, but the
limits imposed by both attributes SHOULD be honored.

Due to multiple layers of encapsulation or the use of content
collections, the key-validity-period attribute can appear in more
than one location in the overall key package. When there are
multiple occurrences of the key-validity-period attribute within the
same scope, all of the included attribute fields MUST contain exactly
the same value. However, if the doNotUseAfter field is absent in an
inner layer, a value MAY appear in an outer layer. Receivers MUST
reject any key package that fails these consistency checks.

16. Key Duration

The key-duration attribute indicates the maximum period of time that
the keying material is intended for use. The date and time that the
duration begins is not specified, but the maximum amount of time that
the keying material can be used to provide security services is
specified. It can appear as a symmetric key, symmetric key package,
asymmetric key, signed, authenticated, authenticated/unprotected, or
content attribute. If the key-duration attribute appears as a
signed, authenticated, authenticated/unprotected, or content
attribute, then all of the keying material within the content MUST
have the same key duration.

The key-duration attribute has the following syntax:

```
aa-keyDurationPeriod ATTRIBUTE ::= {
  TYPE KeyDuration
  IDENTIFIED BY id-kma-keyDuration }

id-kma-keyDuration OBJECT IDENTIFIER ::= {
  joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
  dod(2) infosec(1) keying-material-attributes(13) 7 }

KeyDuration ::= CHOICE {
  hours      [0] INTEGER (1..ub-KeyDuration-hours),
  days           INTEGER (1..ub-KeyDuration-days),
  weeks      [1] INTEGER (1..ub-KeyDuration-weeks),
  months     [2] INTEGER (1..ub-KeyDuration-months),
  years      [3] INTEGER (1..ub-KeyDuration-years) }

ub-KeyDuration-hours   INTEGER ::=  96
ub-KeyDuration-days    INTEGER ::= 732
ub-KeyDuration-weeks   INTEGER ::= 104
ub-KeyDuration-months  INTEGER ::=  72
ub-KeyDuration-years   INTEGER ::= 100
```

The key-validity-period attribute described in the previous section
and the key-duration attribute described in this section provide a
complementary function. The relationship between these attributes is

described in the previous section.

Due to multiple layers of encapsulation or the use of content collections, the key-duration attribute can appear in more than one location in the overall key package. When there are multiple occurrences of the key-duration attribute within the same scope, all of the included attribute fields MUST contain exactly the same value. Receivers MUST reject any key package that fails these consistency checks.

17. Classification

The classification attribute indicates level of classification. The classification attribute specifies the aggregate classification of the package content. It can appear as a symmetric key, symmetric key package, asymmetric key, signed, authenticated, authenticated/unprotected, or content attribute. If the classification attribute appears as a signed, authenticated, authenticated/unprotected, or content attribute, then the value MUST represent the classification of all of the keying material within the content. Encrypted layers MAY contain content at a higher classification that will be revealed once they are decrypted. If the classification attribute is associated with a collection, then the sensitivity of all the data within the collection MUST be dominated by the classification carried in this attribute.

The classification attribute makes use of the ESSSecurityLabel defined in Section 17.1 and from [RFC2634][RFC5911]. The term "classification" is used in this document, but the term "security label" is used in [RFC2634]. The two terms have the same meaning.

[RFC2634][RFC5911] specifies an object identifier and syntax for the security label attribute. The same values are used for the classification attribute:

```
  aa-classificationAttribute ATTRIBUTE ::= {
    TYPE Classification
    IDENTIFIED BY id-aa-KP-classification }

  id-aa-KP-classification OBJECT IDENTIFIER ::= id-aa-securityLabel

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

  Classification ::= ESSSecurityLabel
```

The syntax of ESSSecurityLabel is not repeated here; however, see

section 17.1 for security label conventions that MUST be followed by
implementations of this specification. See [RFC2634] for a complete
discussion of the semantics and syntax.

When the classification attribute appears in more than one location
in the overall key package, each occurrence is evaluated
independently. The content originator MUST ensure that the
classification attribute represents the sensitivity of the plaintext
within the content. That is, the classification MUST dominate any
other plaintext classification attribute value that is present
elsewhere in the overall key package. Note that the classification
attribute value may exceed these other plaintext classification
attribute values if the other attribute values within the SignerInfo,
AuthEnvelopedData, or AuthenticatedData are themselves classified and
warrant the higher security label value.

When the classification attribute appears in more than one location
in the overall key package, each security label might be associated
with a different security policy. Content originators SHOULD avoid
mixing multiple security policies in the same key package whenever
possible since this requires that receivers and intermediaries that
check the classification attribute values to include support for the
union of the security policies that are present. Failure to
recognize an included security policy MUST result in rejection of the
key package.

Receivers MUST reject any key package that includes a classification
for which the receiver's processing environment is not authorized.

17.1. Security Label

The ESSSecurityLabel ASN.1 type is used to represent the
classification. The ESSSecurityLabel is defined in Section 3.2 of
 [RFC2634].

The classification attribute values and classification attribute
values have ASN.1 type ESSSecurityLabel, which is defined in
[RFC2634]. The syntax definition is repeated here to facilitate
discussion:

```
  ESSSecurityLabel ::= SET {
    security-policy-identifier SecurityPolicyIdentifier,
    security-classification    SecurityClassification OPTIONAL,
    privacy-mark               ESSPrivacyMark OPTIONAL,
    security-categories        SecurityCategories OPTIONAL }

  ESSPrivacyMark ::= CHOICE {
    pString      PrintableString (SIZE (1..ub-privacy-mark-length)),
```

```
       utf8String   UTF8String (SIZE (1..MAX)) }
```

A security policy is a set of criteria for the provision of security
services. The security-policy-identifier, which is an object
identifier, is used to identify the security policy associated with
the security label. It indicates the semantics of the other security
label components.

If the key package receiver does not recognize the object identifier
in the security-policy-identifier field and the security label
includes a security-categories field, then the key package contents
MUST NOT be accepted and the enclosed keying material MUST NOT be
used. If the key package receiver does not recognize the object
identifier in the security-policy-identifier field and the security
label does not include a security-categories field, then the key
package contents MAY be accepted only if the security-classification
field is present and it contains a value from the basic hierarchy as
described below.

This specification defines the use of the SecurityClassification
field exactly as is it specified in the 1988 edition of ITU-T
Recommendation X.411 [X.411], which states in part:

  "If present, a security-classification may have one of a
  hierarchical list of values. The basic security-classification
  hierarchy is defined in this Recommendation, but the use of these
  values is defined by the security-policy in force. Additional
  values of security-classification, and their position in the
  hierarchy, may also be defined by a security-policy as a local
  matter or by bilateral agreement. The basic security-
  classification hierarchy is, in ascending order: unmarked,
  unclassified, restricted, confidential, secret, top-secret."

Implementations MUST support the basic security classification
hierarchy. Such implementations MAY also support other security-
classification values; however, the placement of additional values in
the hierarchy MUST be specified by the security policy.

Implementations MUST NOT make access control decisions based on the
privacy-mark. However, information in the privacy-mark can be
displayed to human users by devices that have displays to do so. The
privacy-mark length MUST NOT exceed 128 characters. The privacy-mark
SHALL use the PrintableString choice if all of the characters in the
privacy mark are members of the printable string character set.

If present, security-categories provide further granularity for the
keying material. The security policy in force indicates the
permitted syntaxes of any entries in the set of security categories.

At most, 64 security categories may be present. The security-
categories have ASN.1 type SecurityCategories and further
SecurityCategory [RFC5912], which are both repeated here to
facilitate discussion:

```
SecurityCategories ::= SET SIZE (1..ub-security-categories) OF
                       SecurityCategory
                         {{SupportedSecurityCategories}}

SecurityCategory {SECURITY-CATEGORY:Supported} ::= SEQUENCE {
    type      [0]  IMPLICIT SECURITY-CATEGORY.
                    &id({Supported}),
    value     [1]  EXPLICIT SECURITY-CATEGORY.
                    &Type({Supported}{@type})
}
```

Four security categories are defined and are referred to as the
Restrictive Tag, the Enumerated Tag, the Permissive Tag, and the
Informative Tag. Only the Enumerated Tag and Informative Tag are
permitted in the classification attribute.

The Enumerated Tag is composed of one or more non-negative integers.
Each non-negative integer represents a non-hierarchical security
attribute that applies to the labeled content. Use of the integer
representation is intended to minimize the size of the label since a
particular key package generally contains only a few security
categories attributes, even though a security policy might define a
large set of security categories attributes. Security attributes
enumerated by tags of this type could be restrictive (such as
compartments) or permissive (such as release permissions). Two
object identifiers for the SecurityCategory type field have been
defined, one restrictive and one for permissive. The object
identifiers are:

```
id-enumeratedRestrictiveAttributes OBJECT IDENTIFIER ::= {
  2 16 840 1 101 2 1 8 3 4 }

id-enumeratedPermissiveAttributes OBJECT IDENTIFIER ::= {
  2 16 840 1 101 2 1 8 3 1 }
```

With both the restrictive and permissive security category types, the
corresponding SecurityCategory value has the following ASN.1
definition:

```
EnumeratedTag ::= SEQUENCE {
  tagName          OBJECT IDENTIFIER,
  attributeList    SET OF SecurityAttribute }
```

```
   SecurityAttribute ::= INTEGER (0..MAX)
```

Any security policy that makes use of security categories MUST assign
object identifiers for each tagName, assign the set of integer values
associated with each tagName, and specify the semantic meaning for
each integer value. Restrictive security attributes and permissive
security attributes SHOULD be associated with different tagName
object identifiers.

The Informative Tag is composed of either one or more non-negative
integers or a bit string. Only the integer choice is allowed in this
specification. Each non-negative integer represents a non-
hierarchical security attribute that applies to the labeled content.
Use of the integer representation is intended to minimize the size of
the label since a particular key package generally contains only a
few security categories attributes, even though a security policy
might define a large set of security categories attributes. Security
attributes enumerated by tags of this type are informative (i.e., no
access control is performed). One object identifier for the
SecurityCategory type field has been defined and it is as follows:

```
   id-informativeAttributes OBJECT IDENTIFIER ::= {
     2 16 840 1 101 2 1 8 3 3 }
```

The corresponding SecurityCategory value has the following ASN.1
definition:

```
   InformativeTag ::= SEQUENCE {
     tagName     OBJECT IDENTIFIER,
     attributes  FreeFormField }

   FreeFormField ::= CHOICE {
     bitSetAttributes    BIT STRING,
     securityAttributes  SET OF SecurityAttribute }
```

Any security policy that makes use of security categories MUST assign
object identifiers for each tagName, assign the set of integer values
associated with each tagName, and specify the semantic meaning for
each integer value.

18. Split Key Identifier

The key package originator may include a split-identifier attribute
to designate that the keying material contains a split rather than a
complete key. It may appear as a symmetric and asymmetric key
attribute. The split-identifier attribute MUST NOT appear as a
symmetric key package, signed, authenticated,
authenticated/unprotected, or content attribute. Split keys have two

halves, which are called "A" and "B." The split-identifier attribute
indicates which half is included in the key package, and it
optionally indicates the algorithm that is needed to combine the two
halves. The combine algorithm is OPTIONAL since each key algorithm
has a default mechanism for this purpose, and the combine algorithm
is present only if the default mechanism is not employed.

The key-split-identifier attribute has the following syntax:

```
  aa-splitIdentifier ATTRIBUTE ::= {
    TYPE SplitID
    IDENTIFIED BY id-kma-splitID }

  id-kma-splitID OBJECT IDENTIFIER ::= {
    joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
    dod(2) infosec(1) keying-material-attributes(13) 11 }

  SplitID ::= SEQUENCE {
    ENUMERATED { a(0), b(1) },
    combineAlg  AlgorithmIdentifier
               {COMBINE-ALGORITHM, {CombineAlgorithms}} OPTIONAL }
```

In most cases the default combine algorithm will be employed, which
makes this attribute a simple constant that identifies either the "A"
or "B" half of the split key, which supports implementation of some
key distribution policies.

Note that each split might have its own CRC, but the key and the
check word are both recovered when the two splits are combined.

Since the split-identifier attribute MUST NOT appear as a signed,
authenticated, authenticated/unprotected, or content attribute, a key
package cannot include multiple occurrences of the split-identifier
attribute within the same scope. Receivers MUST reject any key
package in which the split-identifier attribute appears as a signed,
authenticated, authenticated/unprotected, or content attribute.

19. Key Package Type

The key-package-type attribute is a shorthand method for specifying
all aspects of the key package format, including which attributes are
present and the structure of the encapsulated content or collection.
The key-package-type attribute can be used as a signed,
authenticated, authenticated/unprotected, or content attribute.

Rather than implementing the full flexibility of this specification,
some devices may implement support for one or more specific key
package formats instantiating this specification. Those specific

   formats are called templates and can be identified using a key-
   package-type attribute.

   The key-package-type attribute has the following syntax:

```
      aa-keyPackageType ATTRIBUTE ::= {
        TYPE KeyPkgType
        IDENTIFIED BY id-kma-keyPkgType }

      id-kma-keyPkgType OBJECT IDENTIFIER ::= {
        joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
        dod(2) infosec(1) keying-material-attributes(13) 12 }

      KeyPkgType ::= OBJECT IDENTIFIER
```

   Due to multiple layers of encapsulation or the use of content
   collections, the key-package-type attribute can appear in more than
   one location in the overall key package. When there are multiple
   occurrences of the key-package-type attribute, each occurrence is
   used independently. Since the receiver is likely to use the key-
   package-type attribute value as a decoding aid, any error will most
   likely lead to parsing problems, and these problems could result in
   many different errors being reported.

20. Signature Usage

   The signature-usage attribute identifies the CMS content types that
   this key can be used to sign, or be in the cert path for validation.
   Symmetric key packages do not contain signature generation or
   signature validation keying material, so the signature-usage
   attribute MUST NOT appear in a symmetric key package. For an
   asymmetric key package, the signature-usage attribute indicates the
   kind of objects that are to be signed with the private key in the
   package. However, if the asymmetric key package contains a
   Certificate Signature Key, then the signature-usage attribute also
   indicates what signed objects can be validated using certificates
   that are signed by the private key in the asymmetric key package.
   Therefore, the signature-usage attribute also indicates what kind of
   objects that can be signed by the private keys associated with these
   certificates. The signature-usage attribute MUST NOT appear as a
   signed, authenticated, authenticated/unprotected, or content
   attribute.

   The signature-usage attribute has the following syntax:

```
      aa-signatureUsage-v3 ATTRIBUTE ::= {
        TYPE SignatureUsage
        IDENTIFIED BY id-kma-sigUsageV3 }
```

```
   id-kma-sigUsageV3 OBJECT IDENTIFIER ::= {
     joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
     dod(2) infosec(1) keying-material-attributes(13) 22 }

   SignatureUsage ::= CMSContentConstraints
```

The SignatureUsage structure has the same syntax as the
CMSContentConstraints structure from [RFC6010], and it is repeated
here for convenience.

```
   CMSContentConstraints ::= SEQUENCE SIZE (1..MAX) OF
                             ContentTypeConstraint

   ContentTypeGeneration ::= ENUMERATED  {
     canSource(0),
     cannotSource(1)}

   ContentTypeConstraint ::= SEQUENCE {
     contentType     CONTENT-TYPE.&id ({ContentSet|ct-Any,...}),
     canSource       ContentTypeGeneration DEFAULT canSource,
     attrConstraints  AttrConstraintList OPTIONAL }

   Constraint { ATTRIBUTE:ConstraintList } ::= SEQUENCE {
     attrType   ATTRIBUTE.&id({ConstraintList}),
     attrValues  SET SIZE (1..MAX) OF ATTRIBUTE.
                 &Type({ConstraintList}{@attrType})  }

   SupportedConstraints ATTRIBUTE ::= {SignedAttributesSet, ... }

   AttrConstraintList ::= SEQUENCE SIZE (1..MAX) OF
                       Constraint {{ SupportedConstraints }}

   NOTE: SignedAttributesSet is updated by this specification.
```

The SignatureUsage contains a type of CMSContentConstraints. One or
more ContentTypeConstraint MUST appear in CMSContentConstraints.

Within ContentTypeConstraint, the contentType field indicates the
encapsulated content type identifier that can be signed with the
signature key. A particular content type MUST NOT appear more than
once in the list. The CMS protecting content types need not be
included in the list of permitted content types as the use of CMS is
always authorized (see [RFC6010]).

Within ContentTypeConstraint, the canSource enumeration indicates
whether the signature key can be used to directly sign the indicated
content type. If the ContentTypeConstraint is canSource (the default
value), then the signature key can be used to directly sign the

specified content type. If the ContentTypeConstraint is
cannotSource, then the signature key can only be used with the
specified content type if it encapsulates a signature that was
generated by an originator with a ContentTypeConstraint that is
canSource.

Within ContentTypeList, the attrConstraints OPTIONAL field contains a
sequence of content type specific constraints. If the
attrConstraints field is absent, the signature key can be used to
sign the specified content type, without any further checking. If
the either the attrConstraints field is present, then the signature
key can only be used to sign the specified content type if all of the
constraints for that content type are satisfied. Content type
constraints are checked by matching the attribute values in the
attrConstraint field against the attribute value in the content. The
constraints succeed if the attribute is not present; they fail if the
attribute is present and the value is not one of the values provided
in attrConstraint.

The fields of attrConstraints implement content type-specific
constraints. The attrType field is an AttributeType, which is an
object identifier of a signed attribute carried in the SignerInfo of
the content. The attrValues field provides one or more acceptable
signed attribute values. It is a set of AttributeValue. For a
signed content to satisfy the constraint, the SignerInfo MUST include
a signed attribute of the type identified in the attrType field, and
the signed attribute MUST contain one of the values in the set
carried in attrValues.

Since the signature-usage attribute MUST NOT appear as a signed,
authenticated, authenticated/unprotected, or content attribute, an
asymmetric key package cannot include multiple occurrences of the
signature-usage attribute within the same scope. Receivers MUST
reject any asymmetric key package in which the signature-usage
attribute appears as a signed, authenticated,
authenticated/unprotected, or content attribute.

21. Other Certificate Format

The other-certificate-formats attribute specifies the type, format,
and value of certificates that are not X.509 public key certificates.
 Symmetric key packages do not contain any certificates, so the
other-certificate-formats attribute MUST NOT appear in a symmetric
key package. It SHOULD appear in the attributes field, when the
publicKey field is absent and the certificate format is not X.509.
This attribute MUST NOT appear in an attributes field that includes
the user-certificate attribute from Section 8. The other-
certificate-formats attribute MUST NOT appear as a signed,

authenticated, authenticated/unprotected, or content attribute.

The other-certificate-formats attribute has the following syntax:

```
aa-otherCertificateFormats ATTRIBUTE ::= {
  TYPE CertificateChoices
  IDENTIFIED BY id-kma-otherCertFormats }

id-kma-otherCertFormats OBJECT IDENTIFIER ::= {
  joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
  dod(2) infosec(1) keying-material-attributes(13) 19 }

CertificateChoices ::= CHOICE {
  certificate             Certificate,
  extendedCertificate [0] IMPLICIT ExtendedCertificate,
                                 -- Obsolete
  v1AttrCert          [1] IMPLICIT AttributeCertificateV1,
                                 -- Obsolete
  v2AttrCert          [2] IMPLICIT AttributeCertificateV2,
  other               [3] IMPLICIT OtherCertificateFormat }

OtherCertificateFormat ::= SEQUENCE {
  otherCertFormat         OBJECT IDENTIFIER,
  otherCert ANY DEFINED BY  otherCertFormat }
```

The other-certificate-formats attribute makes use of the
CertificateChoices field defined in Section 10.2.2 of [RFC5652].   The
certificate, extendedCertificate, and v1AttrCert fields MUST be
omitted. The v2AttrCert field can include Version 2 Attribute
Certificates. The other field can include EFF certificates and other
as-yet undefined certificate formats.

Since the other-certificate-formats attribute MUST NOT appear as a
signed, authenticated, authenticated/unprotected, or content
attribute, an asymmetric key package cannot include multiple
occurrences of the other-certificate-formats attribute within the
same scope. Receivers MUST reject any asymmetric key package in
which the other-certificate-formats attribute appears as a signed,
authenticated, authenticated/unprotected, or content attribute.

22. PKI Path

The pki-path attribute includes certificates that can aid in the
validation of the certificate carried in the user-certificate
attribute. Symmetric key packages do not contain any certificates,
so the pkiPath attribute MUST NOT appear in a symmetric key package.
It can appear as an asymmetric key, signed, authenticated,
authenticated/unprotected, or content attribute. It can appear in

   the attributes field, when the publicKey field is absent and the
   certificate format is X.509. This attribute MUST NOT appear in an
   AsymmetricKeyPackage that has an other-certificate-formats attribute
   in the attributes field. If the pki-path attribute appears as a
   signed, authenticated, authenticated/unprotected, or content
   attribute, then the value includes certificates that can be used to
   construct certification path to all of the keying material within the
   content. This attribute MUST be supported.

   The syntax is taken from [X.509] but redefined using the ATTRIBUTE
   CLASS from [RFC5911]. The pki-path attribute has the following
   syntax:

     aa-pkiPath ATTRIBUTE ::= {
       TYPE PkiPath
       IDENTIFIED BY id-at-pkiPath }

     id-at-pkiPath OBJECT IDENTIFIER ::= {
      joint-iso-itu-t(2) ds(5) attributes(4) 70 }

     PkiPath ::= SEQUENCE SIZE (1..MAX) OF Certificate

   The first certificate in the sequence is the subject's parent
   Certification Authority (CA). The next certificate is that CA's
   parent, and so on. The end-entity and Trust Anchor are not included
   in this attribute.

   Due to multiple layers of encapsulation or the use of content
   collections, the pki-path attribute can appear in more than one
   location in the overall key package. When the pki-path attribute
   appears in more than one location in the overall key package, each
   occurrence is evaluated independently.

23. Useful Certificates

   The useful-certificates attribute includes certificates that can aid
   in the validation of certificates associated with other parties with
   whom secure communications are anticipated. It can appear as an
   asymmetric key, signed, authenticated, authenticated/unprotected, or
   content attribute. For an asymmetric key that has an other-
   certificate-formats attribute from Section 21 in the attributes
   field, the useful-certificates attribute MUST NOT appear. If the
   useful-certificates attribute appears as a signed, authenticated,
   authenticated/unprotected, or content attribute, then the value
   includes certificates that may be used to validate certificate of
   others the receiver communicates with. This attribute MUST be
   supported.

The useful-certificates attribute has the following syntax:

```
aa-usefulCertificates ATTRIBUTE ::= {
  TYPE CertificateSet
  IDENTIFIED BY id-kma-usefulCerts }

id-kma-usefulCerts OBJECT IDENTIFIER ::= {
  joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
  dod(2) infosec(1) keying-material-attributes(13) 20 }

CertificateSet ::= SET OF CertificateChoices
```

The useful-certificates attribute makes use of the CertificateSet field defined in Section 10.2.3 of [RFC5652]. Within the CertificateChoices field, the extendedCertificate and v1AttrCert fields MUST always be omitted. If the userCertificate attribute from Section 8 is included, the other field MUST NOT be present. If the other-certificate-formats attribute from Section 21 is included, the certificate field MUST NOT be present.

Due to multiple layers of encapsulation or the use of content collections, the useful-certificates attribute can appear in more than one location in the overall key package. When the useful-certificates attribute appears in more than one location in the overall key package, each occurrence is evaluated independently.

24. Key Wrap Algorithm

The key-wrap-algorithm attribute identifies a key wrap algorithm with an algorithm identifier. It can appear as a symmetric key or symmetric key package attribute. When this attribute is present in sKeyAttrs, it indicates that the associated sKey field contains a black key, which is an encrypted key, that that was wrapped by the identified algorithm. When this attribute is present in sKeyPkgAttrs, it indicates that every sKey field in that symmetric key package contains a black key, and that all keys are wrapped by the same designated algorithm.

The key-wrap-algorithm attribute has the following syntax:

```
aa-keyWrapAlgorithm ATTRIBUTE ::= {
  TYPE AlgorithmIdentifier{KEY-WRAP, {KeyEncryptionAlgorithmSet}}
  IDENTIFIED BY id-kma-keyWrapAlgorithm }

id-kma-keyWrapAlgorithm OBJECT IDENTIFIER ::= {
  joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
  dod(2) infosec(1) keying-material-attributes(13) 21 }
```

```
      KeyEncryptionAlgorithmSet KEY-WRAP ::= { ... }
```

25. Content Decryption Key Identifier

The content-decryption-key-identifier attribute can appear as an
unprotected attribute as well as a symmetric and symmetric key
package attribute. The attribute's semantics differ based on the
location.

25.1. Content Decryption Key Identifier: Symmetric Key and Symmetric Key Package

The content-decryption-key-identifier attribute [RFC6032] identifies
the keying material needed to decrypt the sKey. It can appear as a
symmetric key and symmetric key package attribute. If the key-wrap-
algorithm attribute appears in sKeyPkgAttrs, then the corresponding
content-decryption-identifier attribute can appear in either
sKeyPkgAttrs or sKeyAttrs. If the key-wrap-algorithm attribute
appears from Section 24 in sKeyAttrs, then the corresponding content-
decryption-identifier attribute MUST appear in sKeyAttrs.

The content-decryption-key-identifier attribute in included for
convenience:

```
   aa-contentDecryptKeyIdentifier 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.

25.2. Content Decryption Key Identifier: Unprotected

The content-decryption-key-identifier attribute can be used to
identify the keying material that is needed for decryption of the
EncryptedData content if there is any ambiguity.

The content-decryption-key-identifier attribute syntax is found in
Section 25.1. 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.

26. **Certificate Pointers**

   The certificate-pointers attribute can be used to reference one or
   more certificates that may be helpful in the processing of the
   content once it is decrypted. Sometimes certificates are omitted if
   they can be easily fetched. However, an intermediary may have better
   facilities to perform the fetching than the receiver. The
   certificate-pointers attribute may be useful in some environments.
   This attribute can appear as an unprotected and an
   unauthenticated/unprotected attribute.

   The certificate-pointers attribute uses the same syntax and semantics
   as the subject information access certificate extension [RFC5280].
   The certificate-pointers attribute has the following syntax:

```
     aa-certificatePointers ATTRIBUTE ::= {
       TYPE SubjectInfoAccessSyntax
       IDENTIFIED BY id-pe-subjectInfoAccess }

     id-pe-subjectInfoAccess OBJECT IDENTIFIER ::= {
       iso(1) identified-organization(3) dod(6) internet(1)
       security(5) mechanisms(5) pkix(7) pe(1) 11 }

     SubjectInfoAccessSyntax ::= SEQUENCE SIZE (1..MAX) OF
                                 AccessDescription

     AccessDescription ::= SEQUENCE {
       accessMethod       OBJECT IDENTIFIER,
       accessLocation     GeneralName }
```

   As specified in [RFC5280], the id-ad-caRepository access method can
   be used to point to a repository where a Certification Authority
   publishes certificates and Certificate Revocation Lists (CRLs). In
   this case, the accessLocation field tells how to access the
   repository. Where the information is available via http, ftp, or
   ldap, accessLocation contains a uniform resource identifier (URI).
   Where the information is available via the directory access protocol
   (dap), accessLocation contains a directory name.

27. CRL Pointers

The CRL-pointers attribute can be used to reference one or more CRLs
that may be helpful in the processing of the content once it is
decrypted. Sometimes CRLs are omitted to conserve space or to ensure
that the most recent CRL is obtained when the certificate is
validated. However, an intermediary may have better facilities to
perform the fetching than the receiver. The CRL-pointers attribute
may be useful in some environments. This attribute can appear as an
unprotected and unauthenticated/unprotected attribute.

The CRL-pointers attribute has the following syntax:

```
aa-crlPointers ATTRIBUTE ::= {
  TYPE GeneralNames
  IDENTIFIED BY id-aa-KP-crlPointers }

id-aa-KP-crlPointers OBJECT IDENTIFIER ::= {
  joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
  dod(2) infosec(1) attributes(5) 70 }
```

The CRL-pointers attribute uses the GeneralNames syntax from
[RFC5280]. Each name describes a different mechanism to obtain the
same CRL. Where the information is available via http, ftp, or ldap,
GeneralNames contains a uniform resource identifier (URI). Where the
information is available via the directory access protocol (dap),
GeneralNames contains a directory name.

28. Key Package Identifier and Receipt Request

The Key Package Identifier and Receipt Request attribute from
[RFC7191] is also supported. It can appear as a signed attribute,
authenticated, authenticated/unprotected, or content attribute.

29. Additional Error Codes

This specification also defines three additional extendedErrorCodes
[RFC7191]:

```
id-errorCodes OBJECT IDENTIFIER ::= {
  joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
  dod(2) infosec(1) errorCodes(22) }

id-missingKeyType OBJECT IDENTIFIER ::= {
  id-errorCodes 1 }

id-privacyMarkTooLong OBJECT IDENTIFIER ::= {
  id-errorCodes 2 }
```

```
      id-unrecognizedSecurityPolicy OBJECT IDENTIFIER ::= {
        id-errorCodes 3 }

      id-incorrectKeyProvince OBJECT IDENTIFIER ::= {
        id-errorCodes 4 }
```

missingKeyType indicates that all keying material within a package is of the same type; however, the key type attribute is not specified in sKeyPkgAttrs [RFC6031].

privacyMarkTooLong indicates that a classification attribute includes a privacy mark that exceeds 128 characters in length.

unrecognizedSecurityPolicy indicates that a security-policy-identifier is not supported.

incorrectKeyProvince indicates that the value of the key province attribute in a key package does not match the key province constraint of the TA used to validate the key package.

30. Processing Key Package Attribute Values and CMS Content Constraints

Trust anchors may contain constraints for any content type [RFC5934]. When the trust anchor contains constraints for the symmetric key package content type or the asymmetric key package content type, then the constraints provide default values for key package attributes that are not present in the key package and define the set of acceptable values for key package attributes that are present.

When a trust anchor delegates authority by issuing an X.509 certificate, the CMS content constraints certificate extension [RFC6010] may be included to constrain the authorizations. The trust anchor and the X.509 certification path provide default values for key package attributes that are not present in the key package and define the set of acceptable of values for key package attributes that are present.

Constraints on content type usage are represented as attributes.

The processing procedures for the CMS content constraints certificate extension [RFC6010] are part of the validation of a signed or authenticated object, and the procedures yield three output values: cms_constraints, cms_effective_attributes, and cms_default_attributes. Object validation MUST be performed before processing the key package contents, and these outputs values are used as part of key package processing. These same output values are easily generated directly from a trust anchor and the key package

   when no X.509 certification path is involved in validation.

   The cms_effective_attributes provides the set of acceptable values
   for attributes. Each attribute present in the key package that
   corresponds to an entry in cms_effective_attributes MUST contain a
   value that appears in cms_effective_attributes entry. Attributes
   that do not correspond to an entry in cms_effective_attributes are
   unconstrained and may contain any value. Correspondence between
   attributes and cms_effective_attributes is determined by comparing
   the attribute object identifier to object identifier for each entry
   in cms_effective_attributes.

   The cms_default_attributes provides values for attributes that do not
   appear in the key package. If cms_default_attributes includes only
   one attribute value for a particular attribute, then that value is
   used as if it were included in the key package itself. However, if
   cms_default_attributes includes more than one value for a particular
   attribute, then the appropriate value remains ambiguous and the key
   package should be rejected.

   Some attributes can appear in more than one place in the key package,
   and for this reason, the attribute definitions include consistency
   checks. These checks are independent of constraints checking. In
   addition to the consistency checks, each instance of the attribute
   MUST be checked against the set of cms_effective_attributes, and the
   key package MUST be rejected if any of the attributes values are not
   in the set of authorized set of values.

31. Attribute Scope

   This section provides an example symmetric key package in order to
   provide a discussion of the scope of attributes. This is an
   informative section; it is not a normative portion of this
   specification. Figure 1 provides the example. All of the concepts
   apply to either a symmetric key package or an asymmetric key package,
   with the exception of the key-algorithm attribute which is only
   applicable to a symmetric key package. Each of the components is
   labeled with a number inside parentheses for easy reference:

     o (1) is the ContentInfo that must be present as the outermost
       layer of encapsulation. It contains no attributes. It is shown
       for completeness.

     o (2) is a SignedData content type, which includes six signed
       attributes. Four of the signed attributes are keying material
       attributes.

     o (3) is a ContentCollection that includes two encapsulated content

types: a ContentWithAttributes and an EncryptedKeyPackage. This
content type does not provide any attributes.

o (4) is a ContentWithAttributes content type. It encapsulates a
   SignedData content type. Four key material attributes are
   provided.

o (5) is a SignedData content type. It encapsulates a
   SymmetricKeyPackage content type. Six signed attributes are
   provided. Four attributes are key material attributes.

o (6) is a SymmetricKeyPackage content type, and it includes three
   key material attributes. Note that the contents of this key
   package are not encrypted, but the contents are covered by two
   digital signatures.

o (7) is an EncryptedKeyPackage content type. It encapsulates a
   SignedData content type. This content type provides one
   unprotected attribute.

o (8) is a SignedData content type. It encapsulates a
   SymmetricKeyPackage content type. Six signed attributes are
   provided. Four attributes are key material attributes.

o (9) is a SymmetricKeyPackage content type, and it includes three
   key material attributes. Note that the contents of this key
   package are encrypted, and the plaintext keying material is
   covered by one digital signature, and the ciphertext keying
   material is covered by another digital signature.

SignedData content type (2) includes six signed attributes:

o The content-type attribute contains id-ct-contentCollection to
   indicate the type of the encapsulated content, and it has no
   further scope.

o The message-digest attribute contains the one-way hash value of
   the encapsulated content; it is needed to validate the digital
   signature. It has no further scope.

o The classification attribute contains security label for all of
   the plaintext in the encapsulated content. Each classification
   attribute is evaluated separately; it has no further scope. In
   general, the values of this attribute will match or dominate the
   security label values in (4), (5), and (6). The value of this
   attribute might not match or dominate the security label values
   in (8) and (9) since they are encrypted. It is possible that
   these various security label values are associated with different

      security policies. Comparison is not required in order to avoid
      the processing complexity associated with policy mapping.

   o The key-package-receivers-v2 attribute indicates the authorized
     key package receivers, and it has no further scope. The key-
     package-receivers-v2 attribute value within (4) is evaluated
     without regard to the value of this attribute.

   o The key-distribution-period attribute contains two date values:
     doNotDistBefore and doNotDistAfter. These values must match all
     others within the same scope, which in this example is the key-
     distribution-period within (4).

   o The key-package-type attributes indicates the format of the key
     package, and it has no further scope. The key-package-type
     attributes values within (5) and (8) are evaluated without regard
     to the value of this attribute.

   ContentWithAttributes content type (4) includes four attributes:

   o The classification attribute contains security label for all of
     the plaintext in the encapsulated content. Each classification
     attribute is evaluated separately; it has no further scope.

   o The TSEC-Nomenclature attribute includes only the shortTitle
     field, and the value must match all other instances within the
     same scope, which appear in (5) and (6). Note that the TSEC-
     Nomenclature attribute values in (8) and (9) are not in the same
     scope as the TSEC-Nomenclature attribute that appears in (4).

   o The key-package-receivers-v2 attribute indicates the authorized
     key package receivers, and it has no further scope. The key-
     package-receivers-v2 attribute value within (2) is evaluated
     without regard to the value of this attribute.

   o The key-distribution-period attribute contains two date values:
     doNotDistBefore and doNotDistAfter. These values must match all
     others within the same scope, which in this example is the key-
     distribution-period within (2).

   SignedData content type (5) includes six signed attributes:

   o The content-type attribute contains id-ct-KP-skeyPackage to
     indicate the type of the encapsulated content, and it has no
     further scope.

   o The message-digest attribute contains the one-way hash value of
     the encapsulated content; it is needed to validate the digital

signature. It has no further scope.

o The classification attribute contains security label for all of
  the plaintext in the encapsulated content. Each classification
  attribute is evaluated separately; it has no further scope.

o The TSEC-Nomenclature attribute includes only the shortTitle
  field, and the value must match all other instances within the
  same scope, which appear in (6). Since this is within the scope
  of (4), these shortTitle field values must match as well. Note
  that the TSEC-Nomenclature attribute values in (8) and (9) are
  not in the same scope.

o The key-purpose attribute specifies the purpose of the key
  material. All occurrences within the scope must have the same
  value, but in this example, there are no other occurrences within
  the scope. The key-purpose attribute value within (8) is
  evaluated without regard to the value of this value.

o The key-package-type attribute indicates the format of the key
  package, and it has no further scope. The key-package-type
  attribute values within (2) and (8) are evaluated without regard
  to the value of this attribute.

SymmetricKeyPackage content type (6) includes three keying material
attributes, which could appear in the sKeyPkgAttrs or sKeyAttrs
fields:

o The key-algorithm attribute includes only the keyAlg field, and
  it must match all other occurrences within the same scope.
  However, there are no other key-algorithm attribute occurrences
  in the same scope; the key-algorithm attribute value in (9) is
  not in the same scope.

o The classification attribute contains security label for all of
  the plaintext in the key package. Each classification attribute
  is evaluated separately; it has no further scope.

o The TSEC-Nomenclature attribute includes the shortTitle field as
  well as some of the optional fields. The shortTitle field value
  must match the values in (4) and (5), since this content type is
  within their scope. Note that the TSEC-Nomenclature attribute
  values in (8) and (9) are not in the same scope.

EncryptedKeyPackage content type (7) includes one unprotected
attribute, and the encryption will prevent any intermediary that does
not have the ability to decrypt the content from making any
consistency checks on (8) and (9):

     o The content-decryption-key-identifier attribute identifies the
       key that is needed to decrypt the encapsulated content; it has no
       further scope.

   SignedData content type (8) includes six signed attributes:

     o The content-type attribute contains id-ct-KP-skeyPackage to
       indicate the type of the encapsulated content, and it has no
       further scope.

     o The message-digest attribute contains the one-way hash value of
       the encapsulated content; it is needed to validate the digital
       signature. It has no further scope.

     o The classification attribute contains security label for content.
        Each classification attribute is evaluated separately; it has no
       further scope.

     o The TSEC-Nomenclature attribute includes only the shortTitle
       field, and the value must match all other instances within the
       same scope, which appear in (9). Note that the TSEC-Nomenclature
       attribute values in (4), (5), and (6) are not in the same scope.

     o The key-purpose attribute specifies the purpose of the key
       material. All occurrences within the scope must have the same
       value, but in this example, there are no other occurrences within
       the scope. The key-purpose attribute value within (5) is
       evaluated without regard to the value of this attribute.

     o The key-package-type attribute indicates the format of the key
       package, and it has no further scope. The key-package-type
       attribute values within (2) and (5) are evaluated without regard
       to the value of this attribute.

   SymmetricKeyPackage content type (9) includes three keying material
   attributes, which could appear in the sKeyPkgAttrs or sKeyAttrs
   fields:

     o The key-algorithm attribute includes only the keyAlg field, and
       it must match all other occurrences within the same scope.
       However, there are no other key-algorithm attribute occurrences
       in the same scope; the key-algorithm attribute value in (6) is
       not in the same scope.

     o The classification attribute contains security label for all of
       the plaintext in the key package. Each classification attribute
       is evaluated separately; it has no further scope.

   o The TSEC-Nomenclature attribute includes the shortTitle field as
     well as some of the optional fields. The shortTitle field value
     must match the values in (8), since this content type is within
     its scope. Note that the TSEC-Nomenclature attributes values in
     (4), (5), and (6) are not in the same scope.

   In summary, the scope of an attribute includes the encapsulated
   content of the CMS content type in which it appears, and some
   attributes also require consistency checks with other instances that
   appear within the encapsulated content. Proper recognition of scope
   is required to accurately perform attribute processing.

```
   +----------------------------------------------------------------+
   | ContentInfo (1)                                                |
   |+--------------------------------------------------------------+|
   || SignedData (2)                                               ||
   ||+------------------------------------------------------------+||
   ||| ContentCollection (3)                                      |||
   |||+-------------------------++----------------------------+|||
   |||| ContentWithAttributes (4) || EncryptedKeyPackage (7)     ||||
   ||||+-----------------------+||+-------------------------+||||
   ||||| SignedData (5)        |||| SignedData (8)          |||||
   |||||+---------------------+||||+-----------------------+|||||
   |||||| SymmetricKeyPackage (6) ||||||| SymmetricKeyPackage (9) ||||||
   |||||| Attributes:          |||||| Attributes:          ||||||
   ||||||  Key Algorithm      ||||||  Key Algorithm       ||||||
   ||||||  Classification     ||||||  Classification      ||||||
   ||||||  TSEC-Nomenclature  ||||||  TSEC-Nomenclature   ||||||
   |||||+---------------------+||||+-----------------------+|||||
   ||||| Attributes:          ||||  Attributes:           |||||
   |||||  Content Type        ||||   Content Type          |||||
   |||||  Message Digest      ||||   Message Digest        |||||
   |||||  Classification      ||||   Classification        |||||
   |||||  TSEC-Nomenclature   ||||   TSEC-Nomenclature     |||||
   |||||  Key Purpose         ||||   Key Purpose           |||||
   |||||  Key Package Type    ||||   Key Package Type      |||||
   ||||+---------------------- +||+-------------------------+||||
   |||| Attributes:           || Unprotect Attributes:     ||||
   ||||  Classification       ||   Content Decrypt Key ID   ||||
   ||||  TSEC-Nomenclature    |+---------------------------+|||
   ||||  Key Package Receivers |                            |||
   ||||  Key Distribution Period |                          |||
   |||+-------------------------+                           |||
   ||+-------------------------------------------------------+||
   || Attributes:                                            ||
   ||  Content Type                                          ||
   ||  Message Digest                                        ||
   ||  Classification                                        ||
   ||  Key Package Receivers                                 ||
   ||  Key Distribution Period                               ||
   ||  Key Package Type                                      ||
   |+------------------------------------------------------+|
   +----------------------------------------------------------------+
```

Figure 1: Example Illustrating Scope of Attributes

32. Security Considerations

   The majority of this specification is devoted to the syntax and
   semantics of key package attributes. It relies on other
   specifications, especially [RFC2634] [RFC4073] [RFC4108] [RFC5652]
   [RFC5911] [RFC5912] [RFC5958] [RFC6010] [RFC6031]; their security
   considerations apply here. Additionally, cryptographic algorithms
   are used with CMS protecting content types [RFC5959] [RFC6160]
   [RFC6162]; their security considerations apply here as well.

   This specification also relies upon [RFC5280] for the syntax and
   semantics of X.509 certificates. Digital signatures provide data
   integrity or data origin authentication, and encryption provides
   confidentiality.

   Security factors outside the scope of this specification greatly
   affect the assurance provided. The procedures used by Certification
   Authorities (CAs) to validate the binding of the subject identity to
   their public key greatly affect the assurance that ought to be placed
   in the certificate. This is particularly important when issuing
   certificates to other CAs.

   The CMS AuthenticatedData content type MUST be used with care since a
   message authentication code (MAC) is used. The same key is needed to
   generate the MAC or validate the MAC. Thus, any party with access to
   the key needed to validate the MAC can generate a replacement that
   will be acceptable to other recipients.

   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.

33. IANA Considerations

   None.

34. References

34.1 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.

   [RFC4108]   Housley, R., "Using Cryptographic Message Syntax (CMS) to
               Protect Firmware Packages", RFC 4108, August 2005.

   [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.

   [RFC5959]   Turner, S., "Algorithms for Asymmetric Key Package Content
               Type", RFC 5959, 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.

   [RFC6160] Turner, S., "Algorithms for Cryptographic Message Syntax
              (CMS) Protection of Symmetric Key Package Content Types",
              RFC 6160, April 2011.

   [RFC6162] Turner, S., "Elliptic Curve Algorithms for Cryptographic
              Message Syntax (CMS) Asymmetric Key Package Content Type",
              RFC 6162, April 2011.

   [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.

   [RFC7191] Housley, R., "Cryptographic Message Syntax (CMS) Key
              Package Receipt and Error Content Types", RFC 7191, April
              2014.

   [X.509]   ITU-T Recommendation X.509 (2005) | ISO/IEC 9594-8:2005,
              Information technology - Open Systems Interconnection -
              The Directory: Public-key and attribute certificate
              frameworks.

   [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).

34.2 Informative References

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

[X.411]   ITU-T Recommendation X.411 (1988) | ISO/IEC 10021-4:1988,
Data Communication Networks Message Handling Systems -
Message Transfer System - Abstract Service Definition and
Procedures.

Appendix A. **ASN.1 Module**

```
KMAttributes2012
  { joint-iso-itu-t(2) country(16) us(840) organization(1)
    gov(101) dod(2) infosec(1) modules(0) 39 }

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

-- EXPORT ALL

IMPORTS

-- From [RFC5911]

aa-communityIdentifiers, CommunityIdentifier
  FROM CMSFirmwareWrapper-2009
    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
      smime(16) modules(0) id-mod-cms-firmware-wrap-02(40) }

-- From [RFC5911]

aa-contentHint, ESSSecurityLabel, id-aa-securityLabel
  FROM ExtendedSecurityServices-2009
    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
      smime(16) modules(0) id-mod-ess-2006-02(42) }

-- From [RFC5911] [RFC5912]

AlgorithmIdentifier{}, SMIME-CAPS, ParamOptions, KEY-WRAP
  FROM AlgorithmInformation-2009
    { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-algorithmInformation-02(58) }

-- From [RFC5912]

Name, Certificate
  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) }
```

```
   -- From [RFC5912]

   GeneralNames, SubjectInfoAccessSyntax, id-pe-subjectInfoAccess
     FROM PKIX1Implicit-2009
       { iso(1) identified-organization(3) dod(6) internet(1)
         security(5) mechanisms(5) pkix(7) id-mod(0)
         id-mod-pkix1-implicit-02(59) }

   -- FROM [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) }

   -- From [RFC6010]

   CMSContentConstraints
     FROM CMSContentConstraintsCertExtn
       { iso(1) identified-organization(3) dod(6) internet(1)
          security(5) mechanisms(5) pkix(7) id-mod(0)
          cmsContentConstr-93(42) }

   -- From [RFC6268]

   aa-binarySigningTime, BinaryTime
     FROM BinarySigningTimeModule-2010
       { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
         smime(16) modules(0) id-mod-binSigningTime-2009(55) }

   -- From [RFC6268]

   CertificateChoices, CertificateSet, Attribute {},
   aa-contentType, aa-messageDigest
     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 [RFC7191]

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

```
   -- From [X.509]

   certificateExactMatch
     FROM CertificateExtensions
       { joint-iso-itu-t ds(5) module(1) certificateExtensions(26) 4 }

   ;

   -- ATTRIBUTES

   -- Replaces SignedAttributesSet information object set from
   -- [RFC6268].

   SignedAttributesSet ATTRIBUTE ::= {
     aa-contentType                             |
     aa-messageDigest                           |
     aa-contentHint                             |
     aa-communityIdentifiers                    |
     aa-binarySigningTime                       |
     aa-keyProvince-v2                          |
     aa-keyPackageIdentifierAndReceiptRequest   |
     aa-manifest                                |
     aa-keyAlgorithm                            |
     aa-userCertificate                         |
     aa-keyPackageReceivers-v2                  |
     aa-tsecNomenclature                        |
     aa-keyPurpose                              |
     aa-keyUse                                  |
     aa-transportKey                            |
     aa-keyDistributionPeriod                   |
     aa-keyValidityPeriod                       |
     aa-keyDurationPeriod                       |
     aa-classificationAttribute                 |
     aa-keyPackageType                          |
     aa-pkiPath                                 |
     aa-usefulCertificates,
     ... }

   -- Replaces UnsignedAttributes from [RFC6268].

   UnsignedAttributes ATTRIBUTE ::= {
      ...
      }
```

```
   -- Replaces UnprotectedEnvAttributes from [RFC6268].

   UnprotectedEnvAttributes ATTRIBUTE ::= {
     aa-contentDecryptKeyIdentifier |
     aa-certificatePointers         |
     aa-cRLDistributionPoints,
     ...
     }

   -- Replaces UnprotectedEncAttributes from [RFC6268].

   UnprotectedEncAttributes ATTRIBUTE ::= {
     aa-certificatePointers |
     aa-cRLDistributionPoints,
     ...
     }

   -- Replaces AuthAttributeSet from [RFC6268]

   AuthAttributeSet ATTRIBUTE ::= {
     aa-contentType                            |
     aa-messageDigest                          |
     aa-contentHint                            |
     aa-communityIdentifiers                   |
     aa-keyProvice-v2                          |
     aa-binarySigningTime                      |
     aa-keyPackageIdentifierAndReceiptRequest  |
     aa-manifest                               |
     aa-keyAlgorithm                           |
     aa-userCertificate                        |
     aa-keyPackageReceivers-v2                 |
     aa-tsecNomenclature                       |
     aa-keyPurpose                             |
     aa-keyUse                                 |
     aa-transportKey                           |
     aa-keyDistributionPeriod                  |
     aa-keyValidityPeriod                      |
     aa-keyDurationPeriod                      |
     aa-classificationAttribute                |
     aa-keyPackageType                         |
     aa-pkiPath                                |
     aa-usefulCertificates,
     ... }
```

```
   -- Replaces UnauthAttributeSet from [RFC6268]

   UnauthAttributeSet ATTRIBUTE ::= {
     ...
     }

   -- Replaces AuthEnvDataAttributeSet from [RFC6268]

   AuthEnvDataAttributeSet ATTRIBUTE ::= {
     aa-certificatePointers |
     aa-cRLDistributionPoints,
     ...
     }

    -- Replaces UnauthEnvDataAttributeSet from [RFC6268]

   UnauthEnvDataAttributeSet ATTRIBUTE ::= {
     ...
     }

   -- Replaces OneAsymmetricKeyAttributes from [RFC5958]

   OneAsymmetricKeyAttributes ATTRIBUTE ::= {
     aa-userCertificate             |
     aa-tsecNomenclature            |
     aa-keyPurpose                  |
     aa-keyUse                      |
     aa-transportKey                |
     aa-keyDistributionPeriod       |
     aa-keyValidityPeriod           |
     aa-keyDurationPeriod           |
     aa-classificationAttribute     |
     aa-splitIdentifier             |
     aa-signatureUsage-v3           |
     aa-otherCertificateFormats     |
     aa-pkiPath                     |
     aa-usefulCertificates,
     ... }
```

```
   -- Replaces SKeyPkgAttributes from [RFC6031]

   SKeyPkgAttributes ATTRIBUTE ::= {
     aa-keyAlgorithm              |
     aa-tsecNomenclature          |
     aa-keyPurpose                |
     aa-keyUse                    |
     aa-keyDistributionPeriod     |
     aa-keyValidityPeriod         |
     aa-keyDurationPeriod         |
     aa-classificationAttribute   |
     aa-keyWrapAlgorithm          |
     aa-contentDecryptKeyIdentifier,
     ... }

   -- Replaces SKeyAttributes from [RFC6031]

   SKeyAttributes ATTRIBUTE ::= {
     aa-keyAlgorithm              |
     aa-tsecNomenclature          |
     aa-keyPurpose                |
     aa-keyUse                    |
     aa-keyDistributionPeriod     |
     aa-keyValidityPeriod         |
     aa-keyDurationPeriod         |
     aa-classificationAttribute   |
     aa-splitIdentifier           |
     aa-keyWrapAlgorithm          |
     aa-contentDecryptKeyIdentifier,
     ... }
```

```
   -- Replaces ContentAttributeSet from [RFC6268]

   ContentAttributeSet ATTRIBUTE ::= {
     aa-communityIdentifiers                  |
     aa-keyPackageIdentifierAndReceiptRequest |
     aa-keyAlgorithm                          |
     aa-keyPackageReceivers-v2                |
     aa-tsecNomenclature                      |
     aa-keyPurpose                            |
     aa-keyUse                                |
     aa-transportKey                          |
     aa-keyDistributionPeriod                 |
     aa-transportKey                          |
     aa-keyDistributionPeriod                 |
     aa-keyValidityPeriod                     |
     aa-keyDurationPeriod                     |
     aa-classificationAttribute               |
     aa-keyPackageType                        |
     aa-pkiPath                               |
     aa-usefulCertificates,
     ... }

   -- Content Type, Message Digest, and Content Hint, and Binary Signing
   -- Time are imported from [RFC6268].
   -- Community Identifiers is imported from [RFC5911].

   -- Key Province

   aa-keyProvince-v2 ATTRIBUTE ::= {
     TYPE KeyProvinceV2
     IDENTIFIED BY id-aa-KP-keyProvinceV2 }

   id-aa-KP-keyProvinceV2 OBJECT IDENTIFIER ::=
     { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
       dod(2) infosec(1) attributes(5) 71 }

   KeyProvinceV2 ::= OBJECT IDENTIFIER

   -- Manifest Attribute

   aa-manifest ATTRIBUTE ::= {
     TYPE Manifest
     IDENTIFIED BY id-aa-KP-manifest }

   id-aa-KP-manifest OBJECT IDENTIFIER ::=
     { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
       dod(2) infosec(1) attributes(5) 72 }
```

```
   Manifest ::= SEQUENCE SIZE (1..MAX) OF ShortTitle

   -- Key Algorithm Attribute

   aa-keyAlgorithm ATTRIBUTE ::= {
     TYPE KeyAlgorithm
     IDENTIFIED BY id-kma-keyAlgorithm }

   id-kma-keyAlgorithm  OBJECT IDENTIFIER ::=
     { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
       dod(2) infosec(1) keying-material-attributes(13) 1 }

   KeyAlgorithm ::= SEQUENCE {
     keyAlg           OBJECT IDENTIFIER,
     checkWordAlg  [1] OBJECT IDENTIFIER OPTIONAL,
     crcAlg        [2] OBJECT IDENTIFIER OPTIONAL }

   -- User Certificate Attribute

   aa-userCertificate ATTRIBUTE ::= {
     TYPE Certificate
     EQUALITY MATCHING RULE certificateExactMatch
     IDENTIFIED BY id-at-userCertificate }

   id-at-userCertificate OBJECT IDENTIFIER ::=
     { joint-iso-itu-t(2) ds(5) attributes(4) 36 }

   -- Key Package Receivers Attribute

   aa-keyPackageReceivers-v2 ATTRIBUTE ::= {
     TYPE KeyPkgReceiversV2
     IDENTIFIED BY id-kma-keyPkgReceiversV2 }

   id-kma-keyPkgReceiversV2  OBJECT IDENTIFIER ::=
     { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
       dod(2) infosec(1) keying-material-attributes(13) 16 }

   KeyPkgReceiversV2 ::= SEQUENCE SIZE (1..MAX) OF KeyPkgReceiver

   KeyPkgReceiver ::= CHOICE {
     sirEntity  [0] SIREntityName,
     community  [1] CommunityIdentifier }
```

```
   -- TSEC Nomenclature Attribute

   aa-tsecNomenclature ATTRIBUTE ::= {
     TYPE TSECNomenclature
     IDENTIFIED BY id-kma-TSECNomenclature }

   id-kma-TSECNomenclature  OBJECT IDENTIFIER ::=
     { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
       dod(2) infosec(1) keying-material-attributes(13) 3 }

   TSECNomenclature ::= SEQUENCE {
     shortTitle  ShortTitle,
     editionID   EditionID OPTIONAL,
     registerID  RegisterID OPTIONAL,
     segmentID   SegmentID OPTIONAL }

   ShortTitle ::= PrintableString

   EditionID ::= CHOICE {
     char CHOICE {
       charEdition      [1] CharEdition,
       charEditionRange [2] CharEditionRange },
     num CHOICE {
       numEdition       [3] NumEdition,
       numEditionRange  [4] NumEditionRange } }

   CharEdition ::= PrintableString

   CharEditionRange ::= SEQUENCE {
     firstCharEdition  CharEdition,
     lastCharEdition   CharEdition }

   NumEdition ::= INTEGER (0..308915776)

   NumEditionRange ::= SEQUENCE {
     firstNumEdition  NumEdition,
     lastNumEdition   NumEdition }

   RegisterID ::= CHOICE {
     register       [5] Register,
     registerRange  [6] RegisterRange }

   Register ::= INTEGER (0..2147483647)

   RegisterRange ::= SEQUENCE {
     firstRegister  Register,
     lastRegister   Register }
```

```
   SegmentID ::= CHOICE {
     segmentNumber  [7] SegmentNumber,
     segmentRange   [8] SegmentRange }

   SegmentNumber ::= INTEGER (1..127)

   SegmentRange ::= SEQUENCE {
     firstSegment  SegmentNumber,
     lastSegment   SegmentNumber }

   -- Key Purpose Attribute

   aa-keyPurpose ATTRIBUTE ::= {
     TYPE KeyPurpose
     IDENTIFIED BY id-kma-keyPurpose }

   id-kma-keyPurpose  OBJECT IDENTIFIER ::=
     { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
       dod(2) infosec(1) keying-material-attributes(13) 13 }

   KeyPurpose ::= ENUMERATED {
     n-a    (0),    -- Not Applicable
     a      (65),   -- Operational
     b      (66),   -- Compatible Multiple Key
     l      (76),   -- Logistics Combinations
     m      (77),   -- Maintenance
     r      (82),   -- Reference
     s      (83),   -- Sample
     t      (84),   -- Training
     v      (86),   -- Developmental
     x      (88),   -- Exercise
     z      (90),   -- "On the Air" Testing
     ... -- Expect additional key purpose values -- }

   -- Key Use Attribute

   aa-keyUse ATTRIBUTE ::= {
     TYPE KeyUse
     IDENTIFIED BY id-kma-keyUse }

   id-kma-keyUse  OBJECT IDENTIFIER ::=
      { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
        dod(2) infosec(1) keying-material-attributes(13) 14 }
```

```
   KeyUse ::= ENUMERATED {
     n-a    (0),     -- Not Applicable
     ffk    (1),     -- FIREFLY/CROSSTALK Key (Basic Format)
     kek    (2),     -- Key Encryption Key
     kpk    (3),     -- Key Production Key
     msk    (4),     -- Message Signature Key
     qkek   (5),     -- QUADRANT Key Encryption Key
     tek    (6),     -- Traffic Encryption Key
     tsk    (7),     -- Transmission Security Key
     trkek  (8),     -- Transfer Key Encryption Key
     nfk    (9),     -- Netted FIREFLY Key
     effk   (10),     -- FIREFLY Key (Enhanced Format)
     ebfk   (11),     -- FIREFLY Key (Enhanceable Basic Format)
     aek    (12),     -- Algorithm Encryption Key
     wod    (13),     -- Word of Day
     kesk (246),     -- Key Establishment Key
     eik  (247),     -- Entity Identification Key
     ask  (248),     -- Authority Signature Key
     kmk  (249),     -- Key Modifier Key
     rsk  (250),     -- Revocation Signature Key
     csk  (251),     -- Certificate Signature Key
     sak  (252),     -- Symmetric Authentication Key
     rgk  (253),     -- Random Generation Key
     cek  (254),     -- Certificate Encryption Key
     exk  (255),     -- Exclusion Key
     ... -- Expect additional key use values -- }

   -- Transport Key Attribute

   aa-transportKey ATTRIBUTE ::= {
     TYPE TransOp
     IDENTIFIED BY id-kma-transportKey }

   id-kma-transportKey  OBJECT IDENTIFIER ::=
     { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
       dod(2) infosec(1) keying-material-attributes(13) 15 }

   TransOp ::= ENUMERATED {
     transport    (1),
     operational  (2) }

   -- Key Distribution Period Attribute

   aa-keyDistributionPeriod ATTRIBUTE ::= {
     TYPE KeyDistPeriod
     IDENTIFIED BY id-kma-keyDistPeriod }
```

```
   id-kma-keyDistPeriod  OBJECT IDENTIFIER ::=
     { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
       dod(2) infosec(1) keying-material-attributes(13) 5 }

   KeyDistPeriod ::= SEQUENCE {
     doNotDistBefore  [0] BinaryTime OPTIONAL,
     doNotDistAfter       BinaryTime }

   -- Key Validity Period Attribute

   aa-keyValidityPeriod ATTRIBUTE ::= {
     TYPE KeyValidityPeriod
     IDENTIFIED BY id-kma-keyValidityPeriod }

   id-kma-keyValidityPeriod  OBJECT IDENTIFIER ::=
     { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
       dod(2) infosec(1) keying-material-attributes(13) 6 }

   KeyValidityPeriod ::= SEQUENCE {
     doNotUseBefore  BinaryTime,
     doNotUseAfter   BinaryTime OPTIONAL }

   -- Key Duration Attribute

   aa-keyDurationPeriod ATTRIBUTE ::= {
     TYPE KeyDuration
     IDENTIFIED BY id-kma-keyDuration }

   id-kma-keyDuration  OBJECT IDENTIFIER ::=
     { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
       dod(2) infosec(1) keying-material-attributes(13) 7 }

   KeyDuration ::= CHOICE {
     hours   [0] INTEGER (1..ub-KeyDuration-hours),
     days        INTEGER (1..ub-KeyDuration-days),
     weeks   [1] INTEGER (1..ub-KeyDuration-weeks),
     months  [2] INTEGER (1..ub-KeyDuration-months),
     years   [3] INTEGER (1..ub-KeyDuration-years) }

   ub-KeyDuration-hours  INTEGER ::=  96
   ub-KeyDuration-days   INTEGER ::= 732
   ub-KeyDuration-weeks  INTEGER ::= 104
   ub-KeyDuration-months INTEGER ::=  72
   ub-KeyDuration-years  INTEGER ::= 100
```

```
   -- Classification Attribute

   -- The attribute syntax is imported from [RFC6268].  The term
   -- "classification" is used in this document, but the term "security
   -- label" is used in [RFC2634].  The terms have the same meaning.

   aa-classificationAttribute ATTRIBUTE ::= {
     TYPE Classification
     IDENTIFIED BY id-aa-KP-classification }

   id-aa-KP-classification OBJECT IDENTIFIER ::= id-aa-securityLabel

   Classification ::= ESSSecurityLabel

   id-enumeratedRestrictiveAttributes OBJECT IDENTIFIER ::=
     { 2 16 840 1 101 2 1 8 3 4 }

   id-enumeratedPermissiveAttributes OBJECT IDENTIFIER ::=
     { 2 16 840 1 101 2 1 8 3 1 }

   EnumeratedTag ::= SEQUENCE {
     tagName         OBJECT IDENTIFIER,
     attributeList   SET OF SecurityAttribute }

   SecurityAttribute ::= INTEGER (0..MAX)

   -- Split Identifier Attribute

   aa-splitIdentifier ATTRIBUTE ::= {
     TYPE SplitID
     IDENTIFIED BY id-kma-splitID }

   id-kma-splitID  OBJECT IDENTIFIER ::=
     { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
       dod(2) infosec(1) keying-material-attributes(13) 11 }

   SplitID ::= SEQUENCE {
     half        ENUMERATED { a(0), b(1) },
     combineAlg  AlgorithmIdentifier
                 {COMBINE-ALGORITHM, {CombineAlgorithms}}  OPTIONAL }
```

```
   COMBINE-ALGORITHM ::= CLASS {
     &id                OBJECT IDENTIFIER UNIQUE,
     &Params            OPTIONAL,
     &paramPresence     ParamOptions DEFAULT absent,
     &smimeCaps         SMIME-CAPS OPTIONAL
   }
   WITH SYNTAX {
     IDENTIFIER &id
     [PARAMS [TYPE &Params] ARE &paramPresence]
     [SMIME-CAPS &smimeCaps]
   }

   CombineAlgorithms COMBINE-ALGORITHM ::= {
     ...
     }

   -- Key Package Type Attribute

   aa-keyPackageType ATTRIBUTE ::= {
     TYPE KeyPkgType
     IDENTIFIED BY id-kma-keyPkgType }

   id-kma-keyPkgType  OBJECT IDENTIFIER ::=
     { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
       dod(2) infosec(1) keying-material-attributes(13) 12 }

   KeyPkgType ::= OBJECT IDENTIFIER

   -- Signature Usage Attribute

   aa-signatureUsage-v3 ATTRIBUTE ::= {
     TYPE SignatureUsage
     IDENTIFIED BY id-kma-sigUsageV3 }

   id-kma-sigUsageV3  OBJECT IDENTIFIER ::=
     { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
       dod(2) infosec(1) keying-material-attributes(13) 22 }

   SignatureUsage ::= CMSContentConstraints

   -- Other Certificate Format Attribute

   aa-otherCertificateFormats ATTRIBUTE ::= {
     TYPE CertificateChoices
     IDENTIFIED BY id-kma-otherCertFormats }
```

```
   id-kma-otherCertFormats OBJECT IDENTIFIER ::=
     { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
       dod(2) infosec(1) keying-material-attributes(13) 19 }

   -- PKI Path Attribute

   aa-pkiPath ATTRIBUTE ::= {
     TYPE PkiPath
     IDENTIFIED BY id-at-pkiPath }

   id-at-pkiPath OBJECT IDENTIFIER ::=
     { joint-iso-itu-t(2) ds(5) attributes(4) 70 }

   PkiPath ::= SEQUENCE SIZE (1..MAX) OF Certificate

   -- Useful Certificates Attribute

   aa-usefulCertificates ATTRIBUTE ::= {
     TYPE CertificateSet
     IDENTIFIED BY id-kma-usefulCerts }

   id-kma-usefulCerts OBJECT IDENTIFIER ::=
     { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
       dod(2) infosec(1) keying-material-attributes(13) 20 }

   -- Key Wrap Attribute

   aa-keyWrapAlgorithm ATTRIBUTE ::= {
     TYPE AlgorithmIdentifier{KEY-WRAP, {KeyEncryptionAlgorithmSet}}
     IDENTIFIED BY id-kma-keyWrapAlgorithm }

   id-kma-keyWrapAlgorithm OBJECT IDENTIFIER ::=
     { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
       dod(2) infosec(1) keying-material-attributes(13) 21 }

   KeyEncryptionAlgorithmSet KEY-WRAP ::= { ... }

   -- Content Decryption Key Identifier Attribute

   aa-contentDecryptKeyIdentifier 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
```

```
   -- Certificate Pointers Attribute

   aa-certificatePointers ATTRIBUTE ::= {
     TYPE SubjectInfoAccessSyntax
     IDENTIFIED BY id-pe-subjectInfoAccess }

   -- CRL Pointers Attribute

   aa-cRLDistributionPoints ATTRIBUTE ::= {
     TYPE GeneralNames
     IDENTIFIED BY id-aa-KP-crlPointers }

   id-aa-KP-crlPointers  OBJECT IDENTIFIER ::=
     { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
       dod(2) infosec(1) attributes (5) 70 }

   -- ExtendedErrorCodes

   id-errorCodes OBJECT IDENTIFIER ::=
     { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
       dod(2) infosec(1) errorCodes(22) }

   id-missingKeyType OBJECT IDENTIFIER ::= {
     id-errorCodes 1 }

   id-privacyMarkTooLong OBJECT IDENTIFIER ::= {
     id-errorCodes 2 }

   id-unrecognizedSecurityPolicy OBJECT IDENTIFIER ::= {
     id-errorCodes 3 }

   END
```

Authors' Addresses

    Paul Timmel
    National Information Assurance Research Laboratory
    National Security Agency

    Email: pstimme@tycho.ncsc.mil

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

    Email: : housley@vigilsec.com

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

    Email: turners@ieca.com