                   Reuse of CMS Content Encryption Keys

Status of this Memo

   This document specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements. Please refer to the current edition of the "Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol. Distribution of this memo is unlimited.

Abstract

   This document describes a way to include a key identifier in a CMS
   (Cryptographic Message Syntax) enveloped data structure, so that the
   content encryption key can be re-used for further enveloped data
   packets.

Table Of Contents

1. Introduction

   CMS [CMS] specifies EnvelopedData. EnvelopedData supports data
   encryption using either symmetric or asymmetric key management
   techniques. Since asymmetric key establishment is relatively
   expensive, it is desirable in some environments to re-use a shared
   content-encryption key established using asymmetric mechanisms for
   encryption operations in subsequent messages.

   The basic idea here is to reuse the content-encryption key (CEK) from
   a message (say MSG1) to derive the key-encryption key (KEK) for a
   later message, (MSG2), by including a reference value for the CEK in
   message 1, and that same value as the KEKIdentifier for message 2.
   The CEK from message 1 is called the "referenced CEK".

   The key words "MUST", "REQUIRED", "SHOULD", "RECOMMENDED", and "MAY"
   in this document are to be interpreted as described in [RFC2119].

2. Applicability

   This specification is intended to be used to provide more efficient
   selective field confidentiality between communicating peers, in
   particular in the cases where:

   - The originator is a client that wishes to send a number of fields
      to a server (the recipient) in a single transaction, where the
      referenced CEK is used for the separate encryption of each field.

   - The originator and recipient are servers that communicate very
      frequently and use this specification purely for efficiency.

   This specification is not intended to be applicable in all cases. It
   is suited for use where:

   - Its use is further scoped: that is, this specification doesn't
      define a protocol but merely a trick that can be used in a larger
      context and additional specification will be needed for each such
      case. In particular, in order to use this specification, it is
      REQUIRED to define the originators' and recipients' behavior where
      a referenced CEK has been "lost".

   - This specification is not suitable for general group key
      management.

-    The underlying cryptographic API is suitable: it is very likely
     that any cryptographic API that completely "hides" the bits of
     cryptographic keys from the CMS layer will prevent reuse of a
     referenced CEK (since they won't have a primitive that allows
     MSG1.CEK to be transformed to MSG2.KEK).

-    The algorithms for content and key encryption have compatible key
     values and strengths, that is, if MSG1.contentEncryptionAlgorithm
     is a 40bit cipher and MSG2.keyEncryptionAlgorithm requires 168
     bits of keying material, then this specification SHOULD NOT be
     used.

   There are other ways that could be envisaged to establish the
   required symmetric keying material, e.g., by leveraging a group
   keying scheme or by defining a content type that contains a KEK
   value. Although this scheme is much simpler than generic group key
   management, if an implementation already supports group key
   management then this scheme doesn't add value. This scheme is also
   suitable for inclusion in CMS libraries (though the addition of new
   state might be a problem for some implementations), which can offer
   some advantages over application layer schemes (e.g., where the
   content includes MSG2.KEK).

3. How to do it

   In order to reference the content-encryption key (CEK) used in an
   EnvelopedData, a key identifier can be included in the
   unprotectedAttrs field of MSG1. This key can then be used to derive
   the key-encryption key (KEK) for other instances of EnvelopedData or
   for other purposes. If the CEK from MSG1 is to be used to derive the
   KEK for MSG2 then MSG1 MUST contain an unprotectedAttrs Attribute of
   type id-aa-CEKReference with a single value using the CEKReference
   syntax.

   MSG2.KEK is to be derived by reversing the bytes of MSG1.CEK. The
   byte reversal is to avoid an attack where the attacker has a known
   plaintext and the related ciphertext (encrypted with MSG1.CEK) that
   (otherwise) could be directly used as a MSG2.KEK.

   The application MUST ensure that the relevant algorithms are
   compatible. That is, a CEKReference attribute alone can only be used
   where the content-encryption algorithm from MSG1 employs the same
   type of symmetric key as the key-encryption algorithm from MSG2.

Notes:

1) There is nothing to prevent inclusion of a CEKReference attribute
   in MSG2 as well as in MSG1. That is, an originator could "roll"
   the referenced CEK with every message.

2) The CEKReference attribute can occur with any of the choices for
   RecipientInfo: ktri, kari or kekri. Implementors MUST NOT assume
   that CEKReference can only occur where ktri or kari is used.

```
id-aa-CEKReference OBJECT IDENTIFIER ::= { id-aa 30 }
CEKReference ::= OCTET STRING
```

id-aa is an object identifier defined in [CMS-MSG].

In order to allow the originator of MSG1 to indicate the "lifetime"
of the CEK, the originator MAY include a CEKMaxDecrypts attribute,
also in the unprotectedAttrs field of EnvelopedData. This attribute
has an INTEGER syntax (the value MUST be >=1 and maximally 2^31), and
indicates to the recipient the maximum number of messages (excluding
MSG1) that will use the referenced CEK. This Attribute MUST only be
sent when a CEKReference attribute is also included.

The recipient SHOULD maintain the CEKReference information (minimally
the key identifier and the CEK value) while less than maxDecrypt
messages have been successfully received. Recipients SHOULD delete
the CEKReference information after some locally configured period.

When this attribute is not present, originators and recipients SHOULD
behave as if a value of one had been sent.

```
id-aa-CEKMaxDecrypts OBJECT IDENTIFIER ::= { id-aa 31 }
CEKMaxDecrypts ::= INTEGER
```

If CEKMaxDecrypts is missing, or has the value one, then each CEK
will be re-used once as the KEK for the next message. This means
that MSG[n].KEK is the byte-reversal of MSG[n-1].CEK; subsequently
MSG[n+1].KEK will be the byte-reversal of MSG[n].CEK. Note that
MSG[n-1].CEK has no impact whatsoever to MSG[n+1], so long as CEKs
are generated randomly (and not e.g., derived from KEKs somehow).

4. Using different CEK and KEK algorithms

   Where MSG1.content-encryption algorithm and MSG2.key-encryption
   algorithm are the same then the MSG2.KEK is the byte-reverse of
   MSG1.CEK. However, in general, these algorithms MAY differ, e.g.,
   requiring different key lengths. This section specifies a generic
   way to derive MSG2.KEK for such cases.

Note: In some sense, the CEK and KEK algorithms are never the "same", e.g., id-alg-CMS3DESwrap and des-ede3-cbc differ. However, for the purposes of this specification, all we care about is that the algorithms use the same format and size of keying material (see also security considerations) and that they do not differ significantly in terms of the resulting cryptographic "strength." In that sense the two algorithms in the example above are the "same."

Implementations MAY include this functionality.

The basic approach is to use the PBKDF2 key derivation function defined in PKCS#5 [RFC2898], but using MSG1.CEK as input instead of a password. The output of the PBKDF2 function is MSG2.KEK. To this end, a new attribute type is defined which allows passing of the required parameters.

```
id-aa-KEKDerivationAlg OBJECT IDENTIFIER ::= { id-aa 32 }
KEKDerivationAlgorithm ::= SEQUENCE {
     kekAlg           AlgorithmIdentifier,
     pbkdf2Param      PBKDF2-params
}
```

kekAlg is the algorithm identifier (and associated parameters, if any), for the MSG2 key encryption algorithm. Note that it is not necessary to protect this field since modification of keyAlg only represents a denial-of-service attack.

The PBKDF2 algorithm parameters are to be handled as follows:

- The salt MUST use the "specified" element of the CHOICE.
- The message originator selects the iterationCount.
- The value of keyLength is determined by the kekAlg and MUST be present.
- The prf field MUST use the default algorithm specified in [RFC2898] which is algid-hmacWithSHA1 (and so the prf field MUST be omitted).

5. Conformance

This specification only applies to messages where the CEKReference attribute is present. All attributes specified here SHOULD be ignored unless they are present in a message containing a valid, new or recognized, existing value of CEKReference. The CEKMaxDecrypts attribute is to be treated by the recipient as a hint, but MUST be honored by the originator.

The optional to implement KEKDerivationAlgorithm attribute MUST only
be present when MSG1.content-encryption algorithm differs from
MSG2.key-encryption algorithm, in which case it MUST be present.
Implementations that recognize this attribute, but do not support the
functionality SHOULD ignore the attribute.

Ignoring attributes as discussed above, will lead to decryption
failures. CMS implementations SHOULD be able to signal the
particular reason for this failure to the calling application.

6. Security Considerations

Encryption does not provide authentication, for example, if the
encryptedContent is essentially random then recipients MUST NOT
assume that "knowing" a CEKReference value proves anything - anyone
could have created the EnvelopedData. This is relevant both for
security (the recovered plaintext should not be entirely random) and
for avoiding denial of service (the recipient MUST NOT assume that
using the right CEKReference means that message originator is
genuine).

Similarly, using the correct CEKReference does not mean that a
message has not been replayed or inserted, and recipients MUST NOT
assume that replay has been avoided.

The maxDecrypts field presents a potential denial-of-service attack
if a very large value is included by an originator in an attempt to
get a recipient to consume memory by storing the referenced CEKs for
a long period or if the originator never sends the indicated number
of ciphertexts. Recipients SHOULD therefore drop referenced CEKs
where the maxDecrypts value is too large (according to local
configuration) or the referenced CEK has been held for too long a
period.

Suppose MSG1 is sent to a set S1 of users. In the case where MSG2 is
sent to only a subset of users in S1, all users from S1 will still be
able to decrypt MSG2 (since MSG2.KEK is computed only from MSG1.CEK).
Implementers should be aware that in such cases, all members of the
original set of recipients (S1) can access the plaintext of MSG2 and
subsequent messages.

The reason for the byte reversal is as follows: without the byte
reversal, an attacker knowing some of MSG1.plaintext (a prefix in a
field for instance) can use the corresponding ciphertext block as the
next encrypted CEK, i.e., as MSG2.KEKRecipientInfo.encryptedKey. Now
the attacker knows the next CEK. This attacks something this note is
not claiming to protect (origin authentication), but is easily
avoided using the byte reversal. Byte-reversal was chosen over bit-

reversal since bit-reversal would cause parity bits from MSG1.CEK to
be used as keying bits for MSG2.KEK for DES-based algorithms. Note
that byte reversal would similarly affect parity if parity checks
spanned more than one octet, however no well-known algorithms operate
in this way.

Implementations should be very careful with this scheme if MSG[n].KEK
is used to derive MSG[n].CEK, e.g., if MSG[n].CEK were the byte-
reversal of MSG[n].KEK, then this scheme could result in a fixed key
being unexpectedly used.

7. References

    [CMS]   Housley, R., "Cryptographic Message Syntax", RFC 2630, June
               1999.

    [CMS-MSG] Ramsdell, B. "S/MIME Version 3 Message Specification", RFC
               2633, June 1999.

    [RFC2898] Kaliski, B., "PKCS #5: Password-Based Cryptography
               Specification Version 2.0", RFC 2898, September 2000.

    [RFC2026] Bradner, S., "The Internet Standards Process -- Revision
               3", BCP 9, RFC 2026, October 1996.

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

Authors' Addresses

    Stephen Farrell,
    Baltimore Technologies,
    39 Parkgate Street,
    Dublin 8
    IRELAND

    Phone: +353-1-881-6000
    EMail: stephen.farrell@baltimore.ie

    Sean Turner
    IECA, Inc.
    9010 Edgepark Road
    Vienna, VA 22182
    USA

    Phone: +1.703.628.3180
    EMail: turners@ieca.com

Appendix A: ASN.1 Module

```
    SMIMERcek
       { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
         smime(16) modules(0) rcek(13) }

    -- This module contains the definitions of the attributes
    -- used for re-using the content encryption key from a
    -- message in further messages.

    DEFINITIONS IMPLICIT TAGS ::=

    BEGIN
    -- EXPORTS ALL --

    IMPORTS

      AlgorithmIdentifier FROM
          AuthenticationFramework { joint-iso-itu-t ds(5)
                module(1) authenticationFramework(7) 3 } ;

    -- [RFC2898] uses 1993 ASN.1 to define PBKDF2-params.  Since
    -- this specification only uses 1988 ASN.1, the definition is
    -- repeated here for completeness.

    -- The DEFAULT prf field value, MUST be used for this
    -- specification
    digestAlgorithm OBJECT IDENTIFIER ::=
          { iso(1) member-body(2) us(840) rsadsi(113549) 2}
    id-hmacWithSHA1 OBJECT IDENTIFIER ::= {digestAlgorithm 7}

    -- [RFC2898] defines PBKDF2-params using 1993 ASN.1, which
    -- results in the same encoding as produced by the definition
    -- below.  See [RFC2898] for that definition.

    PBKDF2-params ::= SEQUENCE {
      salt CHOICE {
          specified OCTET STRING, -- MUST BE USED
          otherSource AlgorithmIdentifier -- DO NOT USE THIS FIELD
      },
      iterationCount INTEGER (1..MAX),
      keyLength INTEGER (1..MAX) OPTIONAL
    }

      -- id-aa is the arc with all new authenticated and
      -- unauthenticated attributes produced the by S/MIME
      -- Working Group.  It is also defined in [CMS-MSG]
```

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

         -- This attribute contains what will be the key identifier
         -- for subsequent messages
         id-aa-CEKReference OBJECT IDENTIFIER ::= { id-aa 30 }
         CEKReference ::= OCTET STRING

         -- This attribute contains a "hint" to the recipient
         -- indicating how many times the originator will use
         -- the re-used CEK
         id-aa-CEKMaxDecrypts      OBJECT IDENTIFIER ::= { id-aa 31 }
         CEKMaxDecrypts ::= INTEGER

         -- This attribute specifies the key derivation function
         -- to be used when the default byte reversal operation cannot
         -- be used.

         id-aa-KEKDerivationAlg     OBJECT IDENTIFIER ::= { id-aa 32 }
         KEKDerivationAlgorithm ::= SEQUENCE {
             kekAlg          AlgorithmIdentifier,
             pbkdf2Param      PBKDF2-params }


      END
```

Full Copyright Statement

Acknowledgement