Network Working Group
Request for Comments: 3560
Category: Standards Track

R. Housley
Vigil Security

July 2003

Use of the RSAES-OAEP Key Transport Algorithm in the Cryptographic Message Syntax (CMS)

Status of this Memo

Copyright Notice

Abstract

This document describes the conventions for using the RSAES-OAEP key transport algorithm with the Cryptographic Message Syntax (CMS). The CMS specifies the enveloped-data content type, which consists of an encrypted content and encrypted content-encryption keys for one or more recipients. The RSAES-OAEP key transport algorithm can be used to encrypt content-encryption keys for intended recipients.

Table of Contents

1. Introduction

PKCS #1 Version 1.5 [PKCS#1v1.5] specifies a widely deployed variant of the RSA key transport algorithm. PKCS #1 Version 1.5 key transport is vulnerable to adaptive chosen ciphertext attacks, especially when it is used to for key management in interactive applications. This attack is often referred to as the "Million Message Attack," and it explained in [RSALABS] and [CRYPTO98]. Exploitation of this vulnerability, which reveals the result of a particular RSA decryption, requires access to an oracle which will respond to hundreds of thousands of ciphertexts, which are constructed adaptively in response to previously received replies that provide information on the successes or failures of attempted decryption operations.

The attack is significantly less feasible in store-and-forward environments, such as S/MIME. RFC 3218 [MMA] discussed the countermeasures to this attack that are available when PKCS #1 Version 1.5 key transport is used in conjunction with the Cryptographic Message Syntax (CMS) [CMS].

When PKCS #1 Version 1.5 key transport is applied as an intermediate encryption layer within an interactive request-response communications environment, exploitation could be more feasible. However, Secure Sockets Layer (SSL) [SSL] and Transport Layer Security (TLS) [TLS] protocol implementations could include countermeasures that detect and prevent the Million Message Attack and other chosen-ciphertext attacks. These countermeasures are performed within the protocol level.

In the interest of long-term security assurance, it is prudent to adopt an improved cryptographic technique rather than embedding countermeasures within protocols. To this end, an updated version of PKCS #1 has been published. PKCS #1 Version 2.1 [PKCS#1v2.1] supersedes RFC 2313. It preserves support for the PKCS #1 Version 1.5 encryption padding format, and it also defines a new one. To resolve the adaptive chosen ciphertext vulnerability, the PKCS #1 Version 2.1 specifies and recommends use of Optimal Asymmetric Encryption Padding (OAEP) for RSA key transport.

This document specifies the use of RSAES-OAEP key transport algorithm in the CMS. The CMS can be used in either a store-and-forward or an interactive request-response environment.

The CMS supports variety of architectures for certificate-based key management, particularly the one defined by the PKIX working group [PROFILE]. PKCS #1 Version 1.5 and PKCS #1 Version 2.1 require the same RSA public key information. Thus, a certified RSA public key may be used with either RSA key transport technique.

The CMS uses ASN.1 [X.208-88], the Basic Encoding Rules (BER) [X.209-88], and the Distinguished Encoding Rules (DER) [X.509-88].

Throughout this document, when the terms "MUST", "MUST NOT", "SHOULD", and "MAY" are used in capital letters, their use conforms to the definitions in RFC 2119 [STDWORDS]. These key word definitions help make the intent of standards documents as clear as possible. These key words are used in this document to help implementers achieve interoperability.

2. Enveloped-data Conventions

The CMS enveloped-data content type consists of an encrypted content and wrapped content-encryption keys for one or more recipients. The RSAES-OAEP key transport algorithm is used to wrap the content-encryption key for one recipient.

Compliant software MUST meet the requirements for constructing an enveloped-data content type stated in [CMS] Section 6, "Enveloped-data Content Type".

A content-encryption key MUST be randomly generated for each instance of an enveloped-data content type. The content-encryption key is used to encipher the content.

2.1. EnvelopedData Fields

The enveloped-data content type is ASN.1 encoded using the EnvelopedData syntax. The fields of the EnvelopedData syntax MUST be populated as described in this section when RSAES-OAEP key transport is employed for one or more recipients.

The EnvelopedData version MUST be 0, 2, or 3.

The EnvelopedData originatorInfo field is not used for the RSAES-OAEP key transport algorithm. However, this field MAY be present to support recipients using other key management algorithms.

The EnvelopedData recipientInfos CHOICE MUST be KeyTransRecipientInfo. See section 2.2 for further discussion of KeyTransRecipientInfo.

The EnvelopedData encryptedContentInfo contentEncryptionAlgorithm field MUST be a symmetric encryption algorithm identifier.

The EnvelopedData unprotectedAttrs MAY be present.

2.2. KeyTransRecipientInfo Fields

The fields of the KeyTransRecipientInfo syntax MUST be populated as described in this section when RSAES-OAEP key transport is employed for one or more recipients.

The KeyTransRecipientInfo version MUST be 0 or 2. If the RecipientIdentifier uses the issuerAndSerialNumber alternative, then the version MUST be 0. If the RecipientIdentifier uses the subjectKeyIdentifier alternative, then the version MUST be 2.

The KeyTransRecipientInfo RecipientIdentifier provides two alternatives for specifying the recipient's certificate, and thereby the recipient's public key. The recipient's certificate MUST contain a RSA public key. The content-encryption key is encrypted with the recipient's RSA public key. The issuerAndSerialNumber alternative identifies the recipient's certificate by the issuer's distinguished name and the certificate serial number; the subjectKeyIdentifier identifies the recipient's certificate by the X.509 subjectKeyIdentifier extension value.

The KeyTransRecipientInfo keyEncryptionAlgorithm specifies use of the RSAES-OAEP algorithm, and its associated parameters, to encrypt the content-encryption key for the recipient. The key-encryption process is described in [PKCS#1v2.1]. See section 3 of this document for the algorithm identifier and the parameter syntax.

The KeyTransRecipientInfo encryptedKey is the result of encrypting the content-encryption key in the recipient's RSA public key using the RSAES-OAEP algorithm. The RSA public key MUST be at least 1024 bits in length. When using a Triple-DES [3DES] content-encryption key, implementations MUST adjust the parity bits to ensure odd parity for each octet of each DES key comprising the Triple-DES key prior to RSAES-OAEP encryption.

3. RSAES-OAEP Algorithm Identifiers and Parameters

The RSAES-OAEP key transport algorithm is the RSA encryption scheme defined in RFC 3447 [PKCS#1v2.1], where the message to be encrypted is the content-encryption key. The algorithm identifier for RSAES-OAEP is:

```
id-RSAES-OAEP  OBJECT IDENTIFIER  ::=  { iso(1) member-body(2)
  us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 7 }
```

The AlgorithmIdentifier parameters field MUST be present, and the parameters field MUST contain RSAES-OAEP-params. RSAES-OAEP-params has the following syntax:

```
RSAES-OAEP-params  ::=  SEQUENCE  {
  hashFunc    [0] AlgorithmIdentifier DEFAULT sha1Identifier,
  maskGenFunc [1] AlgorithmIdentifier DEFAULT mgf1SHA1Identifier,
  pSourceFunc [2] AlgorithmIdentifier DEFAULT
                      pSpecifiedEmptyIdentifier  }

sha1Identifier  AlgorithmIdentifier  ::=  { id-sha1, NULL }

mgf1SHA1Identifier  AlgorithmIdentifier  ::=
                      { id-mgf1, sha1Identifier }

pSpecifiedEmptyIdentifier  AlgorithmIdentifier ::=
                      { id-pSpecified, nullOctetString }

nullOctetString  OCTET STRING (SIZE (0))  ::=  { ''H }

id-sha1  OBJECT IDENTIFIER  ::=  { iso(1)
                      identified-organization(3) oiw(14)
                      secsig(3) algorithms(2) 26 }

pkcs-1  OBJECT IDENTIFIER  ::=  { iso(1) member-body(2)
                      us(840) rsadsi(113549) pkcs(1) pkcs-1(1) }

id-mgf1  OBJECT IDENTIFIER  ::=  { pkcs-1 8 }

id-pSpecified  OBJECT IDENTIFIER  ::=  { pkcs-1 9 }
```

The fields within RSAES-OAEP-params have the following meanings:

hashFunc identifies the one-way hash function. Implementations MUST support SHA-1 [SHA1], and implementations MAY support other one-way hash functions. The SHA-1 algorithm identifier is comprised of the id-sha1 object identifier and a parameter of NULL. Implementations that perform encryption MUST omit the hashFunc field when SHA-1 is used, indicating that the default algorithm was used. Implementations that perform decryption MUST recognize both the id-sha1 object identifier and an absent hashFunc field as an indication that SHA-1 was used.

maskGenFunc identifies the mask generation function. Implementations MUST support MFG1 [PKCS#1v2.1]. MFG1 requires a one-way hash function, and it is identified in the parameter field of the MFG1 algorithm identifier. Implementations MUST support SHA-1 [SHA1], and implementations MAY support other one-way hash functions. The MFG1 algorithm identifier is comprised of the id-mgf1 object identifier and a parameter that contains the algorithm identifier of the one-way hash function employed with MFG1. The SHA-1 algorithm identifier is comprised of the id-sha1 object identifier and a parameter of NULL. Implementations that perform encryption MUST omit the maskGenFunc field when MFG1 with SHA-1 is used, indicating that the default algorithm was used. Implementations that perform decryption MUST recognize both the id-mgf1 and id-sha1 object identifiers as well as an absent maskGenFunc field as an indication that MFG1 with SHA-1 was used.

pSourceFunc identifies the source (and possibly the value) of the encoding parameters, commonly called P. Implementations MUST represent P by the algorithm identifier, id-pSpecified, indicating that P is explicitly provided as an OCTET STRING in the parameters. The default value for P is an empty string. In this case, pHash in EME-OAEP contains the hash of a zero length string. Implementations MUST support a zero length P value. Implementations that perform encryption MUST omit the pSourceFunc field when a zero length P value is used, indicating that the default value was used. Implementations that perform decryption MUST recognize both the id-pSpecified object identifier and an absent pSourceFunc field as an indication that a zero length P value was used.

4. Certificate Conventions

RFC 3280 [PROFILE] specifies the profile for using X.509 Certificates in Internet applications. When a RSA public key will be used for RSAES-OAEP key transport, the conventions specified in this section augment RFC 3280.

Traditionally, the rsaEncryption object identifier is used to identify RSA public keys. However, to implement all of the recommendations described in the Security Considerations section of this document (see section 6), the certificate user needs to be able to determine the form of key transport that the RSA private key owner associates with the public key.

The rsaEncryption object identifier continues to identify the subject public key when the RSA private key owner does not wish to limit the use of the public key exclusively to RSAES-OAEP. In this case, the

rsaEncryption object identifier MUST be used in the algorithm field within the subject public key information, and the parameters field MUST contain NULL.

```
rsaEncryption  OBJECT IDENTIFIER  ::=  { pkcs-1 1 }
```

Further discussion of the conventions associated with use of the rsaEncryption object identifier can be found in RFC 3279 (see [CERTALGS], section 2.3.1).

When the RSA private key owner wishes to limit the use of the public key exclusively to RSAES-OAEP, then the id-RSAES-OAEP object identifier MUST be used in the algorithm field within the subject public key information, and the parameters field MUST contain RSAES-OAEP-params. The id-RSAES-OAEP object identifier value and the RSAES-OAEP-params syntax are fully described in section 3 of this document.

Regardless of the object identifier used, the RSA public key is encoded in the same manner in the subject public key information. The RSA public key MUST be encoded using the type RSAPublicKey type:

```
RSAPublicKey ::= SEQUENCE {
   modulus            INTEGER,    -- n
   publicExponent     INTEGER  }  -- e
```

Here, the modulus is the modulus n, and publicExponent is the public exponent e. The DER encoded RSAPublicKey is carried in the subjectPublicKey BIT STRING within the subject public key information.

The intended application for the key MAY be indicated in the key usage certificate extension (see [PROFILE], section 4.2.1.3). If the keyUsage extension is present in a certificate that conveys an RSA public key with the id-RSAES-OAEP object identifier, then the key usage extension MUST contain a combination of the following values:

```
keyEncipherment; and
dataEncipherment.
```

However, both keyEncipherment and dataEncipherment SHOULD NOT be present.

When a certificate that conveys an RSA public key with the id-RSAES-OAEP object identifier, the certificate user MUST only use the certified RSA public key for RSAES-OAEP operations, and the certificate user MUST perform those operations using the parameters identified in the certificate.

5. SMIMECapabilities Attribute Conventions

RFC 2633 [MSG], Section 2.5.2 defines the SMIMECapabilities signed attribute (defined as a SEQUENCE of SMIMECapability SEQUENCEs) to be used to specify a partial list of algorithms that the software announcing the SMIMECapabilities can support. When constructing a signedData object, compliant software MAY include the SMIMECapabilities signed attribute announcing that it supports the RSAES-OAEP algorithm.

When all of the default settings are selected, the SMIMECapability SEQUENCE representing RSAES-OAEP MUST include the id-RSAES-OAEP object identifier in the capabilityID field and MUST include an empty SEQUENCE in the parameters field. In this case, RSAES-OAEP is represented by the rSAES-OAEP-Default-Identifier:

```
rSAES-OAEP-Default-Identifier  AlgorithmIdentifier  ::=
                       { id-RSAES-OAEP,
                          { sha1Identifier,
                            mgf1SHA1Identifier,
                            pSpecifiedEmptyIdentifier } }
```

The SMIMECapability SEQUENCE representing rSAES-OAEP-Default-Identifier MUST be DER-encoded as the following hexadecimal string:

```
30 0D 06 09 2A 86 48 86 F7 0D 01 01 07 30 00
```

When settings other than the defaults are selected, the SMIMECapability SEQUENCE representing RSAES-OAEP MUST include the id-RSAES-OAEP object identifier in the capabilityID field and MUST include the RSAES-OAEP-params SEQUENCE that identifies the non-default settings in the parameters field.

When SHA-256 is used in the hashFunc and SHA-256 is used with MGF1 in the maskGenFunc, the SMIMECapability SEQUENCE representing RSAES-OAEP is the rSAES-OAEP-SHA256-Identifier, as specified in Appendix A. The SMIMECapability SEQUENCE representing rSAES-OAEP-SHA256-Identifier MUST be DER-encoded as the following hexadecimal string:

```
30 38 06 09 2A 86 48 86 F7 0D 01 01 07 30 2B 30
0D 06 09 60 86 48 01 65 03 04 02 01 05 00 30 1A
06 09 2A 86 48 86 F7 0D 01 01 08 30 0D 06 09 60
86 48 01 65 03 04 02 01 05 00
```

When SHA-384 is used in the hashFunc and SHA-384 is used with MGF1 in the maskGenFunc, the SMIMECapability SEQUENCE representing RSAES-OAEP is the rSAES-OAEP-SHA384-Identifier, as specified in Appendix A. The

SMIMECapability SEQUENCE representing rSAES-OAEP-SHA384-Identifier MUST be DER-encoded as the following hexadecimal string:

```
30 38 06 09 2A 86 48 86 F7 0D 01 01 07 30 2B 30
0D 06 09 60 86 48 01 65 03 04 02 02 05 00 30 1A
06 09 2A 86 48 86 F7 0D 01 01 08 30 0D 06 09 60
86 48 01 65 03 04 02 02 05 00
```

When SHA-512 is used in the hashFunc and SHA-512 is used with MGF1 in the maskGenFunc, the SMIMECapability SEQUENCE representing RSAES-OAEP is the rSAES-OAEP-SHA512-Identifier, as specified in Appendix A. The SMIMECapability SEQUENCE representing rSAES-OAEP-SHA512-Identifier MUST be DER-encoded as the following hexadecimal string:

```
30 38 06 09 2A 86 48 86 F7 0D 01 01 07 30 2B 30
0D 06 09 60 86 48 01 65 03 04 02 03 05 00 30 1A
06 09 2A 86 48 86 F7 0D 01 01 08 30 0D 06 09 60
86 48 01 65 03 04 02 03 05 00
```

6. Security Considerations

Implementations must protect the RSA private key and the content-encryption key. Compromise of the RSA private key may result in the disclosure of all messages protected with that key. Compromise of the content-encryption key may result in disclosure of the associated encrypted content.

The generation of RSA public/private key pairs relies on a random numbers. The use of inadequate pseudo-random number generators (PRNGs) to generate cryptographic keys can result in little or no security. An attacker may find it much easier to reproduce the PRNG environment that produced the keys, searching the resulting small set of possibilities, rather than brute force searching the whole key space. The generation of quality random numbers is difficult. RFC 1750 [RANDOM] offers important guidance in this area.

Generally, good cryptographic practice employs a given RSA key pair in only one scheme. This practice avoids the risk that vulnerability in one scheme may compromise the security of the other, and may be essential to maintain provable security. While PKCS #1 Version 1.5 [PKCS#1v1.5] has been employed for both key transport and digital signature without any known bad interactions, such a combined use of an RSA key pair is not recommended in the future. Therefore, an RSA key pair used for RSAES-OAEP key transport should not also be used for other purposes. For similar reasons, one RSA key pair should always be used with the same RSAES-OAEP parameters.

This specification requires implementation to support the SHA-1 one-way hash function for interoperability, but support for other one-way hash function is permitted. At the time of this writing, the best (known) collision attacks against SHA-1 are generic attacks with complexity 2^80, where 80 is one-half the bit length of the hash value. When a one-way hash function is used with a digital signature scheme, a collision attack is easily translated into a signature forgery. Therefore, the use of SHA-1 in a digital signature scheme provides a security level of no more than 80 bits. If a greater level of security is desired, then a secure one-way hash function with a longer hash value is needed. SHA-256, SHA-384, and SHA-512 are likely candidates [SHA2].

The metrics for choosing a one-way hash function for use in digital signatures do not directly apply to the RSAES-OAEP key transport algorithm, since a collision attack on the one-way hash function does not directly translate into an attack on the key transport algorithm, unless the encoding parameters P varies (in which case a collision the hash value for different encoding parameters might be exploited).

Nevertheless, for consistency with the practice for digital signature schemes, and in case the encoding parameters P is not the empty string, it is recommended that the same rule of thumb be applied to selection of a one-way hash function for use with RSAES-OAEP. That is, the one-way hash function should be selected so that the bit length of the hash value is at least twice as long as the desired security level in bits.

A 1024-bit RSA public key and SHA-1 both provide a security level of about 80 bits. In [NISTGUIDE], the National Institute of Standards and Technology suggests that a security level of 80 bits is adequate for most applications until 2015. If a security level greater than 80 bits is needed, then a longer RSA public key and a secure one-way hash function with a longer hash value are needed. Again, SHA-256, SHA-384, and SHA-512 are likely candidates for such a one-way hash function. For this reason, the algorithm identifiers for these one-way hash functions are included in the ASN.1 module in Appendix A.

The same one-way hash function should be employed for the hashFunc and the maskGenFunc, but it is not required. Using the same one-way hash function reduces the potential for implementation errors.

7. IANA Considerations

Within the CMS, algorithms are identified by object identifiers (OIDs). All of the OIDs used in this document were assigned in Public-Key Cryptography Standards (PKCS) documents or by the National Institute of Standards and Technology (NIST). No further action by the IANA is necessary for this document or any anticipated updates.

8. Intellectual Property Rights Statement

9. Acknowledgments

This document is the result of contributions from many professionals. I appreciate the hard work of all members of the IETF S/MIME Working Group. Further, I extend a special thanks to Burt Kaliski, Jakob Jonsson, Francois Rousseau, and Jim Schaad.

10. References

This section provides normative and informative references.

10.1. Normative References

[3DES] American National Standards Institute. ANSI X9.52-1998, Triple Data Encryption Algorithm Modes of Operation. 1998.

[CMS] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 3369, August 2002.

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

[PKCS#1v2.1] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications, Version 2.1", RFC 3447, February 2003.

[PROFILE] Housley, R., Polk, W., Ford, W. and D. Solo, "Internet X.509 Public Key Infrastructure: Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002.

[SHA1] National Institute of Standards and Technology. FIPS Pub 180-1: "Secure Hash Standard." April 1995.

[STDWORDS] Bradner, S., "Key Words for Use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[X.208-88] CCITT. Recommendation X.208: Specification of Abstract Syntax Notation One (ASN.1). 1988.

[X.209-88] CCITT. Recommendation X.209: Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1). 1988.

[X.509-88] CCITT. Recommendation X.509: The Directory - Authentication Framework. 1988.

10.2. Informative References

[CERTALGS] Bassham, L., Polk, W. and R. Housley, "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3279, April 2002.

[CRYPTO98] Bleichenbacher, D. "Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1", in H. Krawczyk (editor), Advances in Cryptology - CRYPTO '98 Proceedings, Lecture Notes in Computer Science 1462 (1998), Springer-Verlag, pp. 1-12.

[MMA] Rescorla, E., "Preventing the Million Message Attack on Cryptographic Message Syntax", RFC 3218, January 2002.

[NISTGUIDE] National Institute of Standards and Technology. Second Draft: "Key Management Guideline, Part 1: General Guidance." June 2002. [http://csrc.nist.gov/encryption/kms/guideline-1.pdf]

[PKCS#1v1.5] Kaliski, B., "PKCS #1: RSA Encryption, Version 1.5", RFC 2313, March 1998.

[RANDOM] Eastlake, D., Crocker, S. and J. Schiller, "Randomness Recommendations for Security", RFC 1750, December 1994.

[RSALABS] Bleichenbacher, D., B. Kaliski, and J. Staddon. Recent Results on PKCS #1: RSA Encryption Standard. RSA Laboratories' Bulletin No. 7, June 26, 1998. [http://www.rsasecurity.com/rsalabs/bulletins]

[SHA2] National Institute of Standards and Technology. Draft FIPS Pub 180-2: "Specifications for the Secure Hash Standard." May 2001. [http://csrc.nist.gov/encryption/shs/dfips-180-2.pdf]

[SSL] Freier, A., P. Karlton, and P. Kocher. The SSL Protocol, Version 3.0. Netscape Communications. November 1996. [http://wp.netscape.com/eng/ssl3/draft302.txt]

[TLS] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

Appendix A. ASN.1 Module

```
   CMS-RSAES-OAEP
      { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
        pkcs-9(9) smime(16) modules(0) cms-rsaes-oaep(20) }

   DEFINITIONS IMPLICIT TAGS ::= BEGIN

   -- EXPORTS ALL --

   IMPORTS
      AlgorithmIdentifier
          FROM PKIX1Explicit88 -- RFC 3280
          { iso(1) identified-organization(3) dod(6) internet(1)
            security(5) mechanisms(5) pkix(7) id-mod(0)
            id-pkix1-explicit(18) };

   pkcs-1  OBJECT IDENTIFIER  ::=  { iso(1) member-body(2) us(840)
                        rsadsi(113549) pkcs(1) pkcs-1(1) }

   rsaEncryption  OBJECT IDENTIFIER  ::=  { pkcs-1 1 }

   id-RSAES-OAEP  OBJECT IDENTIFIER  ::=  { pkcs-1 7 }

   RSAES-OAEP-params  ::=  SEQUENCE  {
      hashFunc    [0] AlgorithmIdentifier DEFAULT sha1Identifier,
      maskGenFunc [1] AlgorithmIdentifier DEFAULT mgf1SHA1Identifier,
      pSourceFunc [2] AlgorithmIdentifier DEFAULT

                        pSpecifiedEmptyIdentifier }

   sha1Identifier  AlgorithmIdentifier  ::=  { id-sha1, NULL }

   sha256Identifier  AlgorithmIdentifier  ::=  { id-sha256, NULL }

   sha384Identifier  AlgorithmIdentifier  ::=  { id-sha384, NULL }

   sha512Identifier  AlgorithmIdentifier  ::=  { id-sha512, NULL }

   mgf1SHA1Identifier  AlgorithmIdentifier  ::=
                        { id-mgf1, sha1Identifier }

   mgf1SHA256Identifier  AlgorithmIdentifier  ::=
                        { id-mgf1, sha256Identifier }

   mgf1SHA384Identifier  AlgorithmIdentifier  ::=
                        { id-mgf1, sha384Identifier }
```

```
   mgf1SHA512Identifier  AlgorithmIdentifier  ::=
                         { id-mgf1, sha512Identifier }

   pSpecifiedEmptyIdentifier  AlgorithmIdentifier ::=
                         { id-pSpecified, nullOctetString }

   nullOctetString  OCTET STRING (SIZE (0))  ::=  { ''H }

   id-sha1  OBJECT IDENTIFIER  ::=  { iso(1)
                         identified-organization(3) oiw(14)
                         secsig(3) algorithms(2) 26 }

   id-sha256  OBJECT IDENTIFIER  ::=  { joint-iso-itu-t(2)
                         country(16) us(840) organization(1) gov(101)
                         csor(3) nistalgorithm(4) hashalgs(2) 1 }

   id-sha384  OBJECT IDENTIFIER  ::=  { joint-iso-itu-t(2)
                         country(16) us(840) organization(1) gov(101)
                         csor(3) nistalgorithm(4) hashalgs(2) 2 }

   id-sha512  OBJECT IDENTIFIER  ::=  { joint-iso-itu-t(2)
                         country(16) us(840) organization(1) gov(101)
                         csor(3) nistalgorithm(4) hashalgs(2) 3 }

   id-mgf1  OBJECT IDENTIFIER  ::=  { pkcs-1 8 }

   id-pSpecified  OBJECT IDENTIFIER  ::=  { pkcs-1 9 }

   rSAES-OAEP-Default-Identifier  AlgorithmIdentifier  ::=
                         { id-RSAES-OAEP,
                            { sha1Identifier,
                              mgf1SHA1Identifier,
                              pSpecifiedEmptyIdentifier } }

   rSAES-OAEP-SHA256-Identifier  AlgorithmIdentifier  ::=
                         { id-RSAES-OAEP,
                            { sha256Identifier,
                              mgf1SHA256Identifier,
                              pSpecifiedEmptyIdentifier } }

   rSAES-OAEP-SHA384-Identifier  AlgorithmIdentifier  ::=
                         { id-RSAES-OAEP,
                            { sha384Identifier,
                              mgf1SHA384Identifier,
                              pSpecifiedEmptyIdentifier } }

   rSAES-OAEP-SHA512-Identifier  AlgorithmIdentifier  ::=
                         { id-RSAES-OAEP,
```

```
                        { sha512Identifier,
                          mgf1SHA512Identifier,
                          pSpecifiedEmptyIdentifier } }

   END
```

Author's Address

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

EMail: housley@vigilsec.com

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.