Network Working Group
Request for Comments: 4055
Updates: 3279
Category: Standards Track

J. Schaad
Soaring Hawk Consulting
B. Kaliski
RSA Laboratories
R. Housley
Vigil Security

June 2005

Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile

Status of This Memo

Copyright Notice

Abstract

This document supplements RFC 3279. It describes the conventions for using the RSA Probabilistic Signature Scheme (RSASSA-PSS) signature algorithm, the RSA Encryption Scheme - Optimal Asymmetric Encryption Padding (RSAES-OAEP) key transport algorithm and additional one-way hash functions with the Public-Key Cryptography Standards (PKCS) #1 version 1.5 signature algorithm in the Internet X.509 Public Key Infrastructure (PKI). Encoding formats, algorithm identifiers, and parameter formats are specified.

Table of Contents

1. Introduction

This document supplements RFC 3279 [PKALGS]. This document describes the conventions for using the RSASSA-PSS signature algorithm and the RSAES-OAEP key transport algorithm in the Internet X.509 Public Key Infrastructure (PKI) [PROFILE]. Both of these RSA-based algorithms are specified in [P1v2.1]. The algorithm identifiers and associated parameters for subject public keys that employ either of these algorithms, and the encoding format for RSASSA-PSS signatures are specified. Also, the algorithm identifiers for using the SHA-224, SHA-256, SHA-384, and SHA-512 one-way hash functions with the PKCS #1 version 1.5 signature algorithm [P1v1.5] are specified.

This specification supplements RFC 3280 [PROFILE] which profiles the X.509 Certificates and Certificate Revocation Lists (CRLs) for use in the Internet. This specification extends the list of algorithms discussed in RFC 3279 [PKALGS]. The X.509 Certificate and CRL definitions use ASN.1 [X.208-88], the Basic Encoding Rules (BER) [X.209-88], and the Distinguished Encoding Rules (DER) [X.509-88].

This specification defines the contents of the signatureAlgorithm, signatureValue, signature, and subjectPublicKeyInfo fields within Internet X.509 Certificates and CRLs. For each algorithm, the appropriate alternatives for the keyUsage certificate extension are provided.

1.1. Terminology

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

1.2. RSA Public Keys

RFC 3280 [PROFILE] specifies the profile for using X.509 Certificates in Internet applications. When an RSA public key is used for RSASSA-PSS digital signatures or 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 Security Considerations (Section 8), the certificate user needs to be able to determine the form of digital signature or 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 either RSASSA-PSS or 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 [PKALGS], Section 2.3.1).

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

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, if present, 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 4.

Note: It is not possible to restrict the use of a key to a set of algorithms (i.e., RSASSA-PSS and RSAES-OAEP).

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 keyUsage certificate extension (see [PROFILE], Section 4.2.1.3).

If the keyUsage extension is present in an end-entity certificate that conveys an RSA public key with the id-RSASSA-PSS object identifier, then the keyUsage extension MUST contain one or both of the following values:

```
   nonRepudiation; and
   digitalSignature.
```

If the keyUsage extension is present in a certification authority certificate that conveys an RSA public key with the id-RSASSA-PSS object identifier, then the keyUsage extension MUST contain one or more of the following values:

```
   nonRepudiation;
   digitalSignature;
   keyCertSign; and
   cRLSign.
```

When a certificate conveys an RSA public key with the id-RSASSA-PSS object identifier, the certificate user MUST only use the certified RSA public key for RSASSA-PSS operations, and, if RSASSA-PSS-params is present, the certificate user MUST perform those operations using the one-way hash function, mask generation function, and trailer field identified in the subject public key algorithm identifier parameters within the certificate.

If the keyUsage extension is present in a certificate conveys an RSA public key with the id-RSAES-OAEP object identifier, then the keyUsage extension MUST contain only 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, if RSAES-OAEP-params is present, the certificate user MUST perform those operations using the one-way hash function and mask generation function identified in the subject public key algorithm identifier parameters within the certificate.

2. Common Functions

The RSASSA-PSS signature and the RSAES-OAEP key transport algorithms make use of one-way hash functions and mask generation functions.

2.1. One-way Hash Functions

PKCS #1 version 2.1 [P1v2.1] supports four one-way hash functions for use with the RSASSA-PSS signature algorithm and the RSAES-OAEP key transport algorithm: SHA-1, SHA-256, SHA-384, and SHA-512 [SHA2]. This document adds support for SHA-224 [SHA-224] with both the RSASSA-PSS and the RSAES-OAEP algorithms. While support for additional one-way hash functions could be added in the future, no other one-way hash functions are supported by this specification.

These one-way hash functions are identified by the following object identifiers:

```
id-sha1  OBJECT IDENTIFIER  ::=  { iso(1)
                     identified-organization(3) oiw(14)
                     secsig(3) algorithms(2) 26 }
id-sha224  OBJECT IDENTIFIER  ::=  {{ joint-iso-itu-t(2)
                     country(16) us(840) organization(1) gov(101)
                     csor(3) nistalgorithm(4) hashalgs(2) 4 }
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 }
```

There are two possible encodings for the AlgorithmIdentifier parameters field associated with these object identifiers. The two alternatives arise from the loss of the OPTIONAL associated with the algorithm identifier parameters when the 1988 syntax for AlgorithmIdentifier was translated into the 1997 syntax. Later the OPTIONAL was recovered via a defect report, but by then many people thought that algorithm parameters were mandatory. Because of this history some implementations encode parameters as a NULL element while others omit them entirely. The correct encoding is to omit the parameters field; however, when RSASSA-PSS and RSAES-OAEP were defined, it was done using the NULL parameters rather than absent parameters.

All implementations MUST accept both NULL and absent parameters as legal and equivalent encodings.

To be clear, the following algorithm identifiers are used when a NULL parameter MUST be present:

```
      sha1Identifier  AlgorithmIdentifier  ::=  { id-sha1, NULL }
      sha224Identifier  AlgorithmIdentifier  ::=  { id-sha224, NULL }
      sha256Identifier  AlgorithmIdentifier  ::=  { id-sha256, NULL }
      sha384Identifier  AlgorithmIdentifier  ::=  { id-sha384, NULL }
      sha512Identifier  AlgorithmIdentifier  ::=  { id-sha512, NULL }
```

2.2. Mask Generation Functions

One mask generation function is used with the RSASSA-PSS signature algorithm and the RSAES-OAEP key transport algorithm: MGF1 [P1v2.1]. No other mask generation functions are supported by this specification.

MGF1 is identified by the following object identifier:

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

The parameters field associated with id-mgf1 MUST have a hashAlgorithm value which identifies the hash function being used with MGF1. This value MUST be sha1Identifier, sha224Identifier, sha256Identifier, sha384Identifier, or sha512Identifier, as specified in Section 2.1. Implementations MUST support the default value, sha1Identifier, and MAY support the other four values.

The following algorithm identifiers have been assigned for each of these alternatives:

```
   mgf1SHA1Identifier  AlgorithmIdentifier  ::=
                          { id-mgf1, sha1Identifier }
   mgf1SHA224Identifier  AlgorithmIdentifier  ::=
                          { id-mgf1, sha224Identifier }
   mgf1SHA256Identifier  AlgorithmIdentifier  ::=
                          { id-mgf1, sha256Identifier }
   mgf1SHA384Identifier  AlgorithmIdentifier  ::=
                          { id-mgf1, sha384Identifier }
   mgf1SHA512Identifier  AlgorithmIdentifier  ::=
                          { id-mgf1, sha512Identifier }
```

3. RSASSA-PSS Signature Algorithm

This section describes the conventions for using the RSASSA-PSS signature algorithm with the Internet X.509 Certificate and CRL profile [PROFILE]. The RSASSA-PSS signature algorithm is specified in PKCS #1 version 2.1 [P1v2.1]. The five one-way hash functions discussed in Section 2.1 and the one mask generation function discussed in Section 2.2 can be used with RSASSA-PSS.

CAs that issue certificates with the id-RSASSA-PSS algorithm identifier SHOULD require the presence of parameters in the publicKeyAlgorithms field if the cA boolean flag is set in the basic constraints certificate extension. CAs MAY require that the parameters be present in the publicKeyAlgorithms field for end-entity certificates.

CAs that use the RSASSA-PSS algorithm for signing certificates SHOULD include RSASSA-PSS-params in the subjectPublicKeyInfo algorithm parameters in their own certificates. CAs that use the RSASSA-PSS algorithm for signing certificates or CRLs MUST include RSASSA-PSS-params in the signatureAlgorithm parameters in the TBSCertificate or TBSCertList structures.

Entities that validate RSASSA-PSS signatures MUST support SHA-1. They MAY also support any other one-way hash functions in Section 2.1.

The data to be signed (e.g., the one-way hash function output value) is formatted for the signature algorithm to be used. Then, a private key operation (e.g., RSA decryption) is performed to generate the signature value. This signature value is then ASN.1 encoded as a BIT STRING and included in the Certificate or CertificateList in the signatureValue field. Section 3.2 specifies the format of RSASSA-PSS signature values.

3.1. RSASSA-PSS Public Keys

When RSASSA-PSS is used in an AlgorithmIdentifier, the parameters MUST employ the RSASSA-PSS-params syntax. The parameters may be either absent or present when used as subject public key information. The parameters MUST be present when used in the algorithm identifier associated with a signature value.

When signing, it is RECOMMENDED that the parameters, except for possibly saltLength, remain fixed for all usages of a given RSA key pair.

```
   id-RSASSA-PSS  OBJECT IDENTIFIER  ::=  { pkcs-1 10 }

   RSASSA-PSS-params  ::=  SEQUENCE  {
      hashAlgorithm      [0] HashAlgorithm DEFAULT
                                sha1Identifier,
      maskGenAlgorithm   [1] MaskGenAlgorithm DEFAULT
                                mgf1SHA1Identifier,
      saltLength         [2] INTEGER DEFAULT 20,
      trailerField       [3] INTEGER DEFAULT 1  }
```

The fields of type RSASSA-PSS-params have the following meanings:

hashAlgorithm

The hashAlgorithm field identifies the hash function. It MUST be one of the algorithm identifiers listed in Section 2.1, and the default hash function is SHA-1. Implementations MUST support SHA-1 and MAY support any of the other one-way hash functions listed in Section 2.1. Implementations that perform signature generation MUST omit the hashAlgorithm field when SHA-1 is used, indicating that the default algorithm was used. Implementations that perform signature validation MUST recognize both the sha1Identifier algorithm identifier and an absent hashAlgorithm field as an indication that SHA-1 was used.

maskGenAlgorithm

The maskGenAlgorithm field identifies the mask generation function. The default mask generation function is MGF1 with SHA-1. For MGF1, it is strongly RECOMMENDED that the underlying hash function be the same as the one identified by hashAlgorithm. Implementations MUST support MGF1. MGF1 requires a one-way hash function that is identified in the parameters field of the MGF1 algorithm identifier. Implementations MUST support SHA-1 and MAY support any of the other one-way hash functions listed in section Section 2.1. The MGF1 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 MGF1. The SHA-1 algorithm identifier is comprised of the id-sha1 object identifier and an (optional) parameter of NULL. Implementations that perform signature generation MUST omit the maskGenAlgorithm field when MGF1 with SHA-1 is used, indicating that the default algorithm was used.

Although mfg1SHA1Identifier is defined as the default value for this field, implementations MUST accept both the default value encoding (i.e., an absent field) and mfg1SHA1Identifier to be explicitly present in the encoding.

saltLength

The saltLength field is the octet length of the salt. For a given hashAlgorithm, the recommended value of saltLength is the number of octets in the hash value. Unlike the other fields of type RSASSA-PSS-params, saltLength does not need to be fixed for a given RSA key pair; a different value could be used for each RSASSA-PSS signature generated.

trailerField

The trailerField field is an integer. It provides compatibility with IEEE Std 1363a-2004 [P1363A]. The value MUST be 1, which represents the trailer field with hexadecimal value 0xBC. Other trailer fields, including the trailer field composed of HashID concatenated with 0xCC that is specified in IEEE Std 1363a, are not supported. Implementations that perform signature generation MUST omit the trailerField field, indicating that the default trailer field value was used. Implementations that perform signature validation MUST recognize both a present trailerField field with value 1 and an absent trailerField field.

If the default values of the hashAlgorithm, maskGenAlgorithm, and trailerField fields of RSASSA-PSS-params are used, then the algorithm identifier will have the following value:

```
   rSASSA-PSS-Default-Identifier  AlgorithmIdentifier  ::=  {
                        id-RSASSA-PSS, rSASSA-PSS-Default-Params }

   rSASSA-PSS-Default-Params RSASSA-PSS-Params ::= {
                        sha1Identifier, mgf1SHA1Identifier, 20, 1}
```

3.2. RSASSA-PSS Signature Values

The output of the RSASSA-PSS signature algorithm is an octet string, which has the same length in octets as the RSA modulus n.

Signature values in CMS [CMS] are represented as octet strings, and the output is used directly. However, signature values in certificates and CRLs [PROFILE] are represented as bit strings, and conversion is needed.

To convert a signature value to a bit string, the most significant bit of the first octet of the signature value SHALL become the first bit of the bit string, and so on through the least significant bit of the last octet of the signature value, which SHALL become the last bit of the bit string.

3.3. RSASSA-PSS Signature Parameter Validation

Three possible parameter validation scenarios exist for RSASSA-PSS signature values.

1. The key is identified by the rsaEncryption algorithm identifier. In this case no parameter validation is needed.

2. The key is identified by the id-RSASSA-PSS signature algorithm identifier, but the parameters field is absent. In this case no parameter validation is needed.

3. The key is identified by the id-RSASSA-PSS signature algorithm identifier and the parameters are present. In this case all parameters in the signature structure algorithm identifier MUST match the parameters in the key structure algorithm identifier except the saltLength field. The saltLength field in the signature parameters MUST be greater or equal to that in the key parameters field.

4. RSAES-OAEP Key Transport Algorithm

This section describes the conventions for using the RSAES-OAEP key transport algorithm with the Internet X.509 Certificate and CRL profile [PROFILE]. RSAES-OAEP is specified in PKCS #1 version 2.1 [P1v2.1]. The five one-way hash functions discussed in Section 2.1 and the one mask generation function discussed in Section 2.2 can be used with RSAES-OAEP. Conforming CAs and applications MUST support RSAES-OAEP key transport algorithm using SHA-1. The other four one-way hash functions MAY also be supported.

CAs that issue certificates with the id-RSAES-OAEP algorithm identifier SHOULD require the presence of parameters in the publicKeyAlgorithms field for all certificates. Entities that use a certificate with a publicKeyAlgorithm value of id-RSA-OAEP where the parameters are absent SHOULD use the default set of parameters for RSAES-OAEP-params. Entities that use a certificate with a publicKeyAlgorithm value of rsaEncryption SHOULD use the default set of parameters for RSAES-OAEP-params.

4.1. RSAES-OAEP Public Keys

When id-RSAES-OAEP is used in an AlgorithmIdentifier, the parameters MUST employ the RSAES-OAEP-params syntax. The parameters may be either absent or present when used as subject public key information. The parameters MUST be present when used in the algorithm identifier associated with an encrypted value.

```
   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  }

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

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

The fields of type RSAES-OAEP-params have the following meanings:

hashFunc

The hashFunc field identifies the one-way hash function. It MUST be one of the algorithm identifiers listed in Section 2.1, and the default hash function is SHA-1. Implementations MUST support SHA-1 and MAY support other one-way hash functions listed in Section 2.1. 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 sha1Identifier algorithm identifier and an absent hashFunc field as an indication that SHA-1 was used.

maskGenFunc

The maskGenFunc field identifies the mask generation function. The default mask generation function is MGF1 with SHA-1. For MGF1, it is strongly RECOMMENDED that the underlying hash function be the same as the one identified by hashFunc. Implementations MUST support MGF1. MGF1 requires a one-way hash function that is identified in the parameter field of the MGF1 algorithm identifier. Implementations MUST support SHA-1 and MAY support any of the other one-way hash functions listed in Section 2.1. The MGF1 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 MGF1. The SHA-1 algorithm identifier is comprised of the id-sha1 object identifier and an (optional) parameter of NULL. Implementations that perform encryption MUST omit the maskGenFunc field when MGF1 with SHA-1 is used, indicating that the default algorithm was used.

Although mfg1SHA1Identifier is defined as the default value for this field, implementations MUST accept both the default value encoding (i.e., an absent field) and the mfg1SHA1Identifier to be explicitly present in the encoding.

pSourceFunc

The pSourceFunc field identifies the source (and possibly the value) of the encoding parameters, commonly called P. Implementations MUST represent P by an 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. Implementations that perform decryption MUST support a zero length P value and MAY support other values. Compliant implementations MUST NOT use any value other than id-pSpecified for pSourceFunc.

If the default values of the hashFunc, maskGenFunc, and pSourceFunc fields of RSAES-OAEP-params are used, then the algorithm identifier will have the following value:

```
   rSAES-OAEP-Default-Identifier  AlgorithmIdentifier  ::=
                         { id-RSAES-OAEP,
                           rSAES-OAEP-Default-Params }

   rSAES-OAEP-Default-Params RSASSA-OAEP-params ::=
                            { sha1Identifier,
                              mgf1SHA1Identifier,
                              pSpecifiedEmptyIdentifier  }
```

5. PKCS #1 Version 1.5 Signature Algorithm

RFC 2313 [P1v1.5] specifies the PKCS #1 Version 1.5 signature algorithm. This specification is also included in PKCS #1 Version 2.1 [P1v2.1]. RFC 3279 [PKALGS] specifies the use of the PKCS #1 Version 1.5 signature algorithm with the MD2, MD5, and the SHA-1 one-way hash functions. This section specifies the algorithm identifiers for using the SHA-224, SHA-256, SHA-384, and SHA-512 one-way hash functions with the PKCS #1 version 1.5 signature algorithm.

The RSASSA-PSS signature algorithm is preferred over the PKCS #1 Version 1.5 signature algorithm. Although no attacks are known against PKCS #1 Version 1.5 signature algorithm, in the interest of increased robustness, RSASSA-PSS signature algorithm is recommended for eventual adoption, especially by new applications. This section is included for compatibility with existing applications, and while still appropriate for new applications, a gradual transition to the RSASSA-PSS signature algorithm is encouraged.

The PKCS #1 Version 1.5 signature algorithm with these one-way hash functions and the RSA cryptosystem is implemented using the padding and encoding conventions described in RFC 2313 [P1v1.5].

The message digest is computed using the SHA-224, SHA-256, SHA-384, or SHA-512 one-way hash function.

The PKCS #1 version 1.5 signature algorithm, as specified in RFC 2313, includes a data encoding step. In this step, the message digest and the object identifier for the one-way hash function used to compute the message digest are combined. When performing the data encoding step, the id-sha224, id-sha256, id-sha384, and id-sha512 object identifiers (see Section 2.1) MUST be used to specify the SHA-224, SHA-256, SHA-384, and SHA-512 one-way hash functions, respectively.

The object identifier used to identify the PKCS #1 version 1.5 signature algorithm with SHA-224 is:

```
   sha224WithRSAEncryption  OBJECT IDENTIFIER  ::=  { pkcs-1 14 }
```

The object identifier used to identify the PKCS #1 version 1.5 signature algorithm with SHA-256 is:

```
   sha256WithRSAEncryption  OBJECT IDENTIFIER  ::=  { pkcs-1 11 }
```

The object identifier used to identify the PKCS #1 version 1.5 signature algorithm with SHA-384 is:

```
   sha384WithRSAEncryption  OBJECT IDENTIFIER  ::=  { pkcs-1 12 }
```

The object identifier used to identify the PKCS #1 version 1.5 signature algorithm with SHA-512 is:

```
   sha512WithRSAEncryption  OBJECT IDENTIFIER  ::=  { pkcs-1 13 }
```

When any of these four object identifiers appears within an AlgorithmIdentifier, the parameters MUST be NULL. Implementations MUST accept the parameters being absent as well as present.

The RSA signature generation process and the encoding of the result are described in detail in RFC 2313 [P1v1.5].

6. ASN.1 Module

```
PKIX1-PSS-OAEP-Algorithms
     { iso(1) identified-organization(3) dod(6)
       internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
       id-mod-pkix1-rsa-pkalgs(33) }

   DEFINITIONS EXPLICIT TAGS ::= BEGIN

   -- EXPORTS All;

   IMPORTS

     AlgorithmIdentifier
         FROM PKIX1Explicit88 -- Found in [PROFILE]
         { iso(1) identified-organization(3) dod(6) internet(1)
           security(5) mechanisms(5) pkix(7) id-mod(0)
           id-pkix1-explicit(18) } ;
```

```
      -- ============================
      --   Basic object identifiers
      -- ============================

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

      -- When rsaEncryption is used in an AlgorithmIdentifier the
      -- parameters MUST be present and MUST be NULL.

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

      -- When id-RSAES-OAEP is used in an AlgorithmIdentifier,
      -- and the parameters field is present, it MUST be
      -- RSAES-OAEP-params

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

      -- When id-pSpecified is used in an AlgorithmIdentifier the
      -- parameters MUST be an OCTET STRING.

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

      -- When id-RSASSA-PSS is used in an AlgorithmIdentifier, and the
      -- parameters field is present, it MUST be RSASSA-PSS-params.

      id-RSASSA-PSS  OBJECT IDENTIFIER  ::=  { pkcs-1 10 }

      -- When id-mgf1 is used in an AlgorithmIdentifier the parameters
      -- MUST be present and MUST be a HashAlgorithm.

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

      -- When the following OIDs are used in an AlgorithmIdentifier, the
      -- parameters MUST be present and MUST be NULL.

      sha224WithRSAEncryption  OBJECT IDENTIFIER  ::=  { pkcs-1 14 }

      sha256WithRSAEncryption  OBJECT IDENTIFIER  ::=  { pkcs-1 11 }

      sha384WithRSAEncryption  OBJECT IDENTIFIER  ::=  { pkcs-1 12 }

      sha512WithRSAEncryption  OBJECT IDENTIFIER  ::=  { pkcs-1 13 }
      -- When the following OIDs are used in an AlgorithmIdentifier the
      -- parameters SHOULD be absent, but if the parameters are present,
      -- they MUST be NULL.

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

      id-sha224  OBJECT IDENTIFIER  ::=  { joint-iso-itu-t(2)
                           country(16) us(840) organization(1) gov(101)
                           csor(3) nistalgorithm(4) hashalgs(2) 4 }

      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 }

      -- ============
      --   Constants
      -- ============

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

      nullParameters NULL  ::=  NULL

      -- =========================
      --   Algorithm Identifiers
      -- =========================

      sha1Identifier  AlgorithmIdentifier  ::=  {
                              algorithm id-sha1,
                              parameters nullParameters  }

      sha224Identifier    AlgorithmIdentifier  ::=  {
                              algorithm id-sha224,
                              parameters nullParameters  }

      sha256Identifier    AlgorithmIdentifier  ::=  {
                              algorithm id-sha256,
                              parameters nullParameters  }
      sha384Identifier    AlgorithmIdentifier  ::=  {
                              algorithm id-sha384,
                              parameters nullParameters  }

      sha512Identifier    AlgorithmIdentifier  ::=  {
                              algorithm id-sha512,
                              parameters nullParameters  }

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

      mgf1SHA224Identifier  AlgorithmIdentifier  ::=  {
                              algorithm id-mgf1,
                              parameters sha224Identifier }

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

      mgf1SHA384Identifier  AlgorithmIdentifier  ::=  {
                              algorithm id-mgf1,
                              parameters sha384Identifier }

      mgf1SHA512Identifier  AlgorithmIdentifier  ::=  {
                              algorithm id-mgf1,
                              parameters sha512Identifier }

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

      rSASSA-PSS-Default-Params RSASSA-PSS-params ::=  {
                              hashAlgorithm sha1Identifier,
                              maskGenAlgorithm mgf1SHA1Identifier,
                              saltLength 20,
                              trailerField 1  }

      rSASSA-PSS-Default-Identifier  AlgorithmIdentifier  ::=  {
                              algorithm id-RSASSA-PSS,
                              parameters rSASSA-PSS-Default-Params }

      rSASSA-PSS-SHA224-Identifier  AlgorithmIdentifier  ::=  {
                              algorithm id-RSASSA-PSS,
                              parameters rSASSA-PSS-SHA224-Params }
      rSASSA-PSS-SHA224-Params RSASSA-PSS-params ::= {
                              hashAlgorithm sha224Identifier,
                              maskGenAlgorithm mgf1SHA224Identifier,
                              saltLength 20,
                              trailerField 1  }

      rSASSA-PSS-SHA256-Identifier  AlgorithmIdentifier  ::=  {
                              algorithm id-RSASSA-PSS,
                              parameters rSASSA-PSS-SHA256-Params }

      rSASSA-PSS-SHA256-Params RSASSA-PSS-params ::=  {
                              hashAlgorithm sha256Identifier,
                              maskGenAlgorithm mgf1SHA256Identifier,
                              saltLength 20,
                              trailerField 1  }

      rSASSA-PSS-SHA384-Identifier  AlgorithmIdentifier  ::=  {
                              algorithm id-RSASSA-PSS,
                              parameters rSASSA-PSS-SHA384-Params }

      rSASSA-PSS-SHA384-Params RSASSA-PSS-params ::= {
                              hashAlgorithm sha384Identifier,
                              maskGenAlgorithm mgf1SHA384Identifier,
                              saltLength 20,
                              trailerField 1  }

      rSASSA-PSS-SHA512-Identifier  AlgorithmIdentifier  ::=  {
                              algorithm id-RSASSA-PSS,
                              parameters rSSASSA-PSS-SHA512-params }

      rSSASSA-PSS-SHA512-params RSASSA-PSS-params ::= {
                              hashAlgorithm sha512Identifier,
                              maskGenAlgorithm mgf1SHA512Identifier,
                              saltLength 20,
                              trailerField 1  }

      rSAES-OAEP-Default-Params RSAES-OAEP-params ::=  {
                              hashFunc sha1Identifier,
                              maskGenFunc mgf1SHA1Identifier,
                              pSourceFunc pSpecifiedEmptyIdentifier  }

      rSAES-OAEP-Default-Identifier  AlgorithmIdentifier  ::=  {
                              algorithm id-RSAES-OAEP,
                              parameters rSAES-OAEP-Default-Params }

      rSAES-OAEP-SHA224-Identifier  AlgorithmIdentifier  ::=  {
                              algorithm id-RSAES-OAEP,
                              parameters rSAES-OAEP-SHA224-Params }
```

```
   rSAES-OAEP-SHA224-Params RSAES-OAEP-params ::=   {
                            hashFunc sha224Identifier,
                            maskGenFunc mgf1SHA224Identifier,
                            pSourceFunc pSpecifiedEmptyIdentifier  }

   rSAES-OAEP-SHA256-Identifier  AlgorithmIdentifier  ::=  {
                            algorithm id-RSAES-OAEP,
                            parameters rSAES-OAEP-SHA256-Params }

   rSAES-OAEP-SHA256-Params RSAES-OAEP-params ::=  {
                            hashFunc sha256Identifier,
                            maskGenFunc mgf1SHA256Identifier,
                            pSourceFunc pSpecifiedEmptyIdentifier  }

   rSAES-OAEP-SHA384-Identifier  AlgorithmIdentifier  ::=  {
                            algorithm id-RSAES-OAEP,
                            parameters rSAES-OAEP-SHA384-Params }

   rSAES-OAEP-SHA384-Params RSAES-OAEP-params ::=  {
                            hashFunc sha384Identifier,
                            maskGenFunc mgf1SHA384Identifier,
                            pSourceFunc pSpecifiedEmptyIdentifier  }

   rSAES-OAEP-SHA512-Identifier  AlgorithmIdentifier  ::=  {
                            algorithm id-RSAES-OAEP,
                            parameters rSAES-OAEP-SHA512-Params }

   rSAES-OAEP-SHA512-Params RSAES-OAEP-params ::=  {
                            hashFunc sha512Identifier,
                            maskGenFunc mgf1SHA512Identifier,
                            pSourceFunc pSpecifiedEmptyIdentifier  }

   -- ===================
   --   Main structures
   -- ===================

   -- Used in SubjectPublicKeyInfo of X.509 Certificate.

   RSAPublicKey  ::=  SEQUENCE  {
      modulus           INTEGER,    -- n
      publicExponent    INTEGER  }  -- e
   -- AlgorithmIdentifier parameters for id-RSASSA-PSS.
   -- Note that the tags in this Sequence are explicit.

   RSASSA-PSS-params  ::=  SEQUENCE  {
      hashAlgorithm     [0] HashAlgorithm DEFAULT
                               sha1Identifier,
      maskGenAlgorithm  [1] MaskGenAlgorithm DEFAULT
                               mgf1SHA1Identifier,
      saltLength        [2] INTEGER DEFAULT 20,
      trailerField      [3] INTEGER DEFAULT 1  }

   HashAlgorithm  ::=  AlgorithmIdentifier

   MaskGenAlgorithm  ::=  AlgorithmIdentifier

   -- AlgorithmIdentifier parameters for id-RSAES-OAEP.
   -- Note that the tags in this Sequence are explicit.

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

   END
```

7. References

7.1. Normative References

[P1v1.5]   Kaliski, B., "PKCS #1: RSA Encryption Version 1.5", RFC 2313, March 1998.

[P1v2.1]   Jonsson, J. and B. Kaliski, "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.

[SHA2]   National Institute of Standards and Technology (NIST), FIPS 180-2: Secure Hash Standard, 1 August 2002.

[SHA224]   Housley, R., "A 224-bit One-way Hash Function: SHA-224", RFC 3874, September 2004.

[STDWORDS] Bradner, S., "Key Words for Use in RFCs to Indicate Requirement Levels", 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.

7.2. Informative References

[CMS] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 3852, July 2004.

[GUIDE] 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]

[P1363A] IEEE Std 1363a-2004, Standard Specifications for Public Key Cryptography - Amendment 1: Additional Techniques, 2004.

[PKALGS] 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.

[RANDOM] Eastlake 3rd, D., Crocker, S., and J. Schiller, "Randomness Recommendations for Security", RFC 1750, December 1994.

[SHA-1-ATTACK] Wang, X., Yin, Y.L., and H. Yu, "Finding Collisions in the Full SHA1", to appear, CRYPTO 2005. Preprint available at http://theory.csail.mit.edu/~yiqun/shanote.pdf.

8. Security Considerations

This specification supplements RFC 3280 [PROFILE]. The Security Considerations section of that document applies to this specification as well.

Implementations must protect the RSA private key. Compromising the RSA private key may result in the disclosure of all messages protected with that key.

The generation of RSA public/private key pairs relies on a random numbers. Using 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 and search the resulting small set of possibilities, than to brute force search the whole key space. The generation of quality random numbers is difficult and 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 [P1v1.5] 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 RSASSA-PSS signature generation should not be used for other purposes. For similar reasons, one RSA key pair should always be used with the same RSASSA-PSS parameters (except possibly for the salt length). Likewise, an RSA key pair used for RSAES-OAEP key transport should not 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 implementations to support the SHA-1 one-way hash function for interoperability, but support for other one-way hash functions is permitted. Wang et al. [SHA-1-ATTACK] have recently discovered a collision attack against SHA-1 with complexity 2^69. This attack, which can produce two new messages with the same hash value, is the first attack on SHA-1 faster than the generic attack with complexity 2^80, where 80 is one-half the bit length of the hash value.

In general, when a one-way hash function is used with a digital signature scheme, a collision attack is easily translated into a signature forgery. Therefore, using SHA-1 in a digital signature scheme provides a security level of no more than 69 bits if the attacker can persuade the signer to sign a message resulting from a collision attack. If the attacker can't persuade the signer to sign such a message, however, then SHA-1 still provides a security level of at least 80 bits since the best (known) inversion attack (which produces a new message with a previous hash value) is the generic attack with complexity 2^160. 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 reasonable choices [SHA2], although their security needs to be reconfirmed in light of the SHA-1 results.

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 vary (in which case a collision of 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 selecting 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.

The key size selected impacts the strength achieved when implementing cryptographic services. Thus, selecting appropriate key sizes is critical to implementing appropriate security. A 1024-bit RSA public key is considered to provide a security level of about 80 bits. In [GUIDE], the National Institute of Standards and Technology (NIST) suggests that a security level of 80 bits is adequate for the protection of sensitive information until 2015. This recommendation is likely to be revised based on recent advances, and is expected to be more conservative, suggesting that a security level of 80 bits is adequate protection of sensitive information until 2010. 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. SHA-224, SHA-256, SHA-384, and SHA-512 are reasonable choices for such a one-way hash function, modulo the reconfirmation noted above. For this reason, the algorithm identifiers for these one-way hash functions are included in the ASN.1 module in Section 6.

Current implementations MUST support 1024-bit RSA public key sizes. Before the end of 2007, implementations SHOULD support RSA public key sizes of at least 2048 bits and SHOULD support SHA-256. This requirement is intended to allow adequate time for users to deploy the stronger digital signature capability by 2010.

When using RSASSA-PSS, the same one-way hash function should be employed for the hashAlgorithm and the maskGenAlgorithm, but it is not required. When using RSAES-OAEP, the same one-way hash function should be employed for the hashFunc and the maskGenFunc, but it is not required. In each case, using the same one-way hash function helps with security analysis and reduces implementation complexity.

9. IANA Considerations

Within the certificates and CRLs, algorithms are identified by object identifiers. All object identifiers 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.

Authors' Addresses

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

EMail: housley@vigilsec.com

Burt Kaliski
RSA Laboratories
174 Middlesex Turnpike
Bedford, MA 01730
USA

EMail: bkaliski@rsasecurity.com

Jim Schaad
Soaring Hawk Consulting
PO Box 675
Gold Bar, WA 98251
USA

EMail: jimsch@exmsft.com

Acknowledgement

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