Groupe de travail Réseau
Request for Comments : 4055
RFC mise à jour : 3279
Catégorie : En cours de normalisation
Traduction Claude Brière de L'Isle

J. Schaad, Soaring Hawk Consulting
B. Kaliski, RSA Laboratories
R. Housley, Vigil Security
juin 2005

Algorithmes et identifiants supplémentaires pour la cryptographie RSA à utiliser dans le profil de certificat de clé publique et de liste de révocation de certificat (CRL) X.509 de l'Internet

Statut du présent mémoire
Le présent document spécifie un protocole Internet en cours de normalisation pour la communauté de l'Internet. Il appelle à la discussion et à des suggestions pour son amélioration. Prière de se référer à l'édition actuelle des "Normes officielles des protocoles de l'Internet" (STD 1) pour connaître l'état de normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.
(La traduction du présent document incorpore les erratas acceptés jusqu'en 2018)

Résumé
Le présent document complète la RFC 3279. Il décrit les conventions d'utilisation de l'algorithme de signature RSA de schéma de signature probabiliste (RSASSA-PSS, *RSA Probabilistic Signature Scheme*) l'algorithme de transport de clé de RSA de schéma de chiffrement avec bourrage optimal de chiffrement asymétrique (RSAES-OAEP, *Encryption Scheme - Optimal Asymmetric Encryption Padding*) et des fonctions de hachage unidirectionnelles supplémentaires avec l'algorithme de signature des normes de cryptographie à clé publique n° 1 (PKCS, *Public-Key Cryptography Standards*) version 1.5 dans l'infrastructure de clé publique (PKI, *Public Key Infrastructure*) X.509 dans l'Internet. Les formats de codage, les identifiants d'algorithmes, et les formats de paramètres sont spécifiés.

Table des Matières

1. Introduction

Le présent document complète la [RFC3279]. Il décrit les conventions d'utilisation de l'algorithme de signature RSASSA-PSS et l'algorithme de transport de clés RSAES-OAEP dans l'infrastructure de clé publique (PKI, *Public Key Infrastructure*) X.509 de l'Internet [RFC3280]. Ces deux algorithmes fondés sur RSA sont spécifiés dans la [RFC3447]. Les identifiants d'algorithme et les paramètres associés pour les clés publiques sujettes qui emploient l'un de ces algorithmes, et le format de codage pour les signatures RSASSA-PSS sont spécifiés. Sont aussi spécifiés les identifiants d'algorithme pour utiliser les fonctions de hachage

unidirectionnelles SHA-224, SHA-256, SHA-384, et SHA-512 avec l'algorithme de signature PKCS n° 1, version 1.5 [RFC2313].

La présente spécification complète la [RFC3280] qui définit les profils des certificats X.509 et les listes de révocation de certificat (CRL, *Certificate Revocation List*) à utiliser dans l'Internet. Elle étend la liste des algorithmes exposés dans la [RFC3279]. Les définitions de certificat X.509 et de CRL utilisent l'ASN.1 [X.208-88], les règles de codage de base (BER, *Basic Encoding Rules*) [X.209-88], et les règles de codage en métalangage distinctif (DER, *Distinguished Encoding Rules*) [X.509-88].

La présente spécification définit le contenu des champs signatureAlgorithm, signatureValue, signature, et subjectPublicKeyInfo dans les certificats X.509 et les CRL pour l'Internet. Pour chaque algorithme sont fournies les solutions de remplacement appropriées pour l'extension de certificat keyUsage.

1.1 Terminologie

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDÉ", "PEUT", et "FACULTATIF" dans ce document sont à interpréter comme décrit dans le BCP 14, RFC 2119 [RFC2119].

1.2 Clés publiques RSA

La [RFC3280] spécifie le profil d'utilisation des certificats X.509 dans les applications Internet. Lorsque une clé publique RSA est utilisée pour des signatures numériques RSASSA-PSS ou un transport de clé RSAES-OAEP, les conventions spécifiées dans cette section augmentent celles de la RFC 3280.

Traditionnellement, l'identifiant d'objet rsaEncryption *(chiffrement RSA)* est utilisé pour identifier les clés publiques RSA. Cependant, pour mettre en œuvre toutes les recommandations décrites dans les considérations sur la sécurité (Section 8), l'utilisateur de certificat doit être capable de déterminer la forme de signature numérique ou de transport de clé que le possesseur de la clé privée RSA associe à la clé publique.

L'identifiant d'objet rsaEncryption continue d'identifier la clé publique sujette lorsque le possesseur de la clé privée RSA ne souhaite pas limiter l'utilisation de la clé publique à exclusivement RSASSA-PSS ou RSAES-OAEP. Dans ce cas, l'identifiant d'objet rsaEncryption DOIT être utilisé dans le champ d'algorithme dans les informations de clé publique sujette, et le champ Paramètres DOIT contenir NULL.

```
IDENTIFIANT D'OBJET rsaEncryption ::= { pkcs-1 1 }
```

On trouvera plus d'éléments sur les conventions associées à l'utilisation de l'identifiant d'objet rsaEncryption au paragraphe 2.3.1 de la [RFC3279].

Lorsque le possesseur de la clé privée RSA souhaite limiter l'utilisation de la clé publique exclusivement à RSASSA-PSS, l'identifiant d'objet id-RSASSA-PSS DOIT alors être utilisé dans le champ Algorithme dans les informations de clé publique sujette, et, s'il est présent, le champ Paramètres DOIT contenir RSASSA-PSS-params. La valeur de l'identifiant d'objet id-RSASSA-PSS et la syntaxe de RSASSA-PSS-params sont décrites à la Section 3.

Lorsque le possesseur de la clé privée RSA souhaite limiter l'utilisation de la clé publique exclusivement à RSAES-OAEP, l'identifiant d'objet id-RSAES-OAEP DOIT alors être utilisé dans le champ Algorithme dans les informations de clé publique sujette, et, si il est présent, le champ Paramètres DOIT contenir RSAES-OAEP-params. La valeur de l'identifiant d'objet id-RSAES-OAEP et la syntaxe de RSAES-OAEP-params sont décrites à la Section 4.

Note : Il n'est pas possible de restreindre l'utilisation d'une clé à un ensemble d'algorithmes (c'est-à-dire, à RSASSA-PSS et RSAES-OAEP).

Sans considération de l'identifiant d'objet utilisé, la clé publique RSA est codée de la même façon que dans les informations de clé publique sujette. La clé publique RSA DOIT être codée en utilisant le type RSAPublicKey :

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

Ici, le module est modulo n, et publicExponent est l'exposant public e. La clé publique RSA RSAPublicKey codée en DER est

portée dans la chaîne binaire subjectPublicKey au sein des informations de clé publique sujette.

L'application prévue pour la clé PEUT être indiquée dans l'extension de certificat keyUsage (voir au paragraphe 4.2.1.3 de la [RFC3280]).

Si l'extension keyUsage est présente dans un certificat d'entité d'extrémité qui porte une clé publique RSA avec l'identifiant d'objet id-RSASSA-PSS, l'extension keyUsage DOIT alors contenir une ou les deux valeurs suivantes : nonRepudiation et digitalSignature.

Si l'extension keyUsage est présente dans un certificat d'autorité de certification qui porte une clé publique RSA avec l'identifiant d'objet id-RSASSA-PSS, l'extension keyUsage DOIT alors contenir une ou plusieurs des valeurs suivantes :
nonRepudiation *(non répudiation)*,
digitalSignature *(signature numérique)*,
keyCertSign *(signature de certificat de clé)*,
cRLSign *(signature de liste de révocation de certificat)*.

Lorsque un certificat porte une clé publique RSA avec l'identifiant d'objet id-RSASSA-PSS, l'utilisateur du certificat DOIT utiliser la clé publique RSA certifiée pour les seules opérations RSASSA-PSS, et, si RSASSA-PSS-params est présent, l'utilisateur du certificat DOIT effectuer ces opérations en utilisant la fonction de hachage unidirectionnelle, la fonction de génération de gabarit, et le champ d'en-queue identifiés dans les paramètres d'identifiant d'algorithme de clé publique sujette au sein du certificat.

Si l'extension keyUsage présente dans un certificat porte une clé publique RSA avec l'identifiant d'objet id-RSAES-OAEP, l'extension keyUsage DOIT alors contenir les seules valeurs suivantes : keyEncipherment *(chiffrement de clé)*, et dataEncipherment *(chiffrement de données)*.

Cependant les valeurs keyEncipherment et dataEncipherment NE DEVRAIENT PAS être toutes les deux présentes.

Lorsque un certificat porte une clé publique RSA avec l'identifiant d'objet id-RSAES-OAEP, l'utilisateur du certificat NE DOIT utiliser la clé publique RSA certifiée QUE pour les opérations RSAES-OAEP, et, si RSAES-OAEP-params est présent, l'utilisateur du certificat DOIT effectuer ces opérations en utilisant la fonction de hachage unidirectionnel et la fonction de génération de gabarit identifiées dans les paramètres d'identifiant d'algorithme de clé publique sujette au sein du certificat.

2. Fonctions communes

Les algorithmes de signature RSASSA-PSS et de transport de clé RSAES-OAEP utilisent des fonctions de hachage unidirectionnelles et des fonctions de génération de gabarit.

2.1 Fonctions de hachage unidirectionnelles

PKCS n° 1 version 2.1 [RFC3447] accepte quatre fonctions de hachage unidirectionnelles à utiliser avec l'algorithme de signature RSASSA-PSS et l'algorithme de transport de clé RSAES-OAEP : SHA-1, SHA-256, SHA-384, et SHA-512 [SHA2]. Le présent document ajoute la prise en charge de SHA-224 [SHA-224] avec les deux algorithmes RSASSA-PSS et RSAES-OAEP. Bien que la prise en charge de fonctions de hachage unidirectionnelles supplémentaires puisse être ajoutée à l'avenir, aucune autre fonction de hachage unidirectionnelle n'est prise en charge par la présente spécification.

Ces fonctions de hachage unidirectionnelles sont identifiées par les identifiants d'objet suivants :

```
IDENTIFIANT D'OBJET id-sha1    ::=  { iso(1) identified-organization(3) oiw(14) secsig(3) algorithms(2) 26 }
IDENTIFIANT D'OBJET id-sha224  ::=  { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3)
                                      nistalgorithm(4) hashalgs(2) 4 }
IDENTIFIANT D'OBJET id-sha256  ::=  { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3)
                                      nistalgorithm(4) hashalgs(2) 1 }
IDENTIFIANT D'OBJET id-sha384  ::=  { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3)
                                      nistalgorithm(4) hashalgs(2) 2 }
IDENTIFIANT D'OBJET id-sha512  ::=  { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3)
                                      nistalgorithm(4) hashalgs(2) 3 }
```

Il y a deux codages possibles pour le champ de paramètres AlgorithmIdentifier associé à ces identifiants d'objet. Les deux solutions proviennent de la perte du FACULTATIF associé aux paramètres d'identifiant d'algorithme lorsque la syntaxe de 1988

pour AlgorithmIdentifier a été traduite en syntaxe 1997. Plus tard le FACULTATIF a été récupéré via un rapport d'erreur, mais à ce moment là, de nombreuses personnes pensaient que les paramètres d'algorithme étaient obligatoires. À cause de cette histoire certaines mises en œuvre codent les paramètres comme un élément NULL tandis que d'autres les omettent entièrement. Le codage correct est d'omettre le champ Paramètres ; cependant, lorsque RSASSA-PSS et RSAES-OAEP ont été définis, cela a été fait en utilisant des paramètres NULL plutôt que des paramètres absents.

Toutes les mises en œuvre DOIVENT accepter les deux paramètres NULL et absents comme codages légaux et équivalents.

Pour être clair, les identifiants d'algorithme suivants sont utilisés lorsque un paramètre NULL DOIT être présent :
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 Fonctions de génération de gabarit

Une fonction de génération de gabarit est utilisée avec l'algorithme de signature RSASSA-PSS et l'algorithme de transport de clé RSAES-OAEP : MGF1 [RFC3447]. Aucune autre fonction de génération de gabarit n'est prise en charge par la présente spécification.

MGF1 est identifié par l'identifiant d'objet suivant :

IDENTIFIANT D'OBJET id-mgf1 ::= { pkcs-1 8 }

Le champ Paramètres associé à id-mgf1 DOIT avoir une valeur de hashAlgorithm qui identifie la fonction de hachage qui est utilisée avec MGF1. Cette valeur DOIT être sha1Identifier, sha224Identifier, sha256Identifier, sha384Identifier, ou sha512Identifier, comme spécifié au paragraphe 2.1. Les mises en œuvre DOIVENT prendre en charge la valeur par défaut sha1Identifier, et PEUVENT prendre en charge les quatre autres valeurs.

Les identifiants d'algorithme suivants ont été alloués pour chacune des solutions de remplacement :
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. Algorithme de signature RSASSA-PSS

La présente section décrit les conventions pour l'utilisation de l'algorithme de signature RSASSA-PSS avec le certificat X.509 Internet et le profil de CRL [RFC3280]. L'algorithme de signature RSASSA-PSS est spécifié dans PKCS n° 1 version 2.1 [RFC3447]. les cinq fonctions de hachage unidirectionnelles discutées au paragraphe 2.1 et la fonction de génération de gabarit discutée au paragraphe 2.2 peuvent être utilisées avec RSASSA-PSS.

Les CA qui produisent les certificats avec l'identifiant d'algorithme id-RSASSA-PSS DEVRAIENT exiger la présence des paramètres dans le champ d'algorithme subjectPublicKeyInfo si le fanion booléen cA est établi dans l'extension de certificat des contraintes de base. Les CA PEUVENT exiger que les paramètres soient présents dans le champ publicKeyAlgorithms pour les certificats de l'entité d'extrémité.

Les CA qui utilisent l'algorithme RSASSA-PSS pour signer les certificats DEVRAIENT inclure RSASSA-PSS-params dans les paramètres d'algorithme subjectPublicKeyInfo dans leurs propres certificats. Les CA qui utilisent l'algorithme RSASSA-PSS pour signer des certificats ou des CRL DOIVENT inclure RSASSA-PSS-params dans le paramètre signatureAlgorithm dans les structures TBSCertificate ou TBSCertList.

Les entités qui valident les signatures RSASSA-PSS DOIVENT prendre en charge SHA-1. Elles PEUVENT aussi prendre en charge toutes les autres fonctions de hachage unidirectionnelles du paragraphe 2.1.

Les données à signer (par exemple, la valeur de résultat de la fonction de hachage unidirectionnelle) sont formatées pour l'algorithme de signature à utiliser. Ensuite, une opération de clé privée (par exemple, déchiffrement RSA) est effectuée pour générer la valeur de signature. Cette valeur de signature est ensuite codée en ASN.1 comme CHAÎNE BINAIRE et incluse

dans Certificate ou CertificateList dans le champ signatureValue. Le paragraphe 3.2 spécifie le format des valeurs de signature RSASSA-PSS.

3.1 Clés publiques RSASSA-PSS

Lorsque RSASSA-PSS est utilisé dans un AlgorithmIdentifier, les paramètres DOIVENT employer la syntaxe de RSASSA-PSS-params. Les paramètres peuvent être soit absents, soit présents quand ils sont utilisés comme des informations de clé publique sujette. Les paramètres DOIVENT être présents lorsque ils sont utilisés dans l'identifiant d'algorithme associé à une valeur de signature.

Lors de la signature, il est RECOMMANDÉ que les paramètres, sauf éventuellement saltLength, restent fixes pour toutes les utilisations d'une certaine paire de clés RSA.

```
IDENTIFIANT D'OBJET  id-RSASSA-PSS   ::=  { pkcs-1 10 }
    RSASSA-PSS-params  ::=  SEQUENCE  {
      hashAlgorithm       [0] HashAlgorithm DEFAUT sha1Identifier,
      maskGenAlgorithm  [1] MaskGenAlgorithm DEFAUT mgf1SHA1Identifier,
      saltLength             [2] ENTIER DEFAUT 20,
      trailerField           [3] ENTIER DEFAUT 1  }
```

Les champs de type RSASSA-PSS-params ont la signification suivante :

hashAlgorithm
Le champ hashAlgorithm identifie la fonction de hachage. Il DOIT être un des identifiants d'algorithmes mentionnés au paragraphe 2.1, et la fonction de hachage par défaut est SHA-1. Les mises en œuvre DOIVENT prendre en charge SHA-1 et PEUVENT prendre en charge n'importe laquelle des autres fonctions de hachage unidirectionnelles mentionnées au paragraphe 2.1. Les mises en œuvre qui effectuent la génération de signature DOIVENT omettre le champ hashAlgorithm lorsque SHA-1 est utilisé, ce qui indique que l'algorithme par défaut a été utilisé. Les mises en œuvre qui effectuent la validation de signature DOIVENT reconnaître l'identifiant d'algorithme sha1Identifier et un champ hashAlgorithm absent comme l'indication que SHA-1 a été utilisé.

maskGenAlgorithm
Le champ maskGenAlgorithm identifie la fonction de génération de gabarit. La fonction de génération de gabarit par défaut est MGF1 avec SHA-1. Pour MGF1, il est fortement RECOMMANDÉ que la fonction de hachage sous-jacente soit la même que celle identifiée par hashAlgorithm. Les mises en œuvre DOIVENT prendre en charge MGF1. MGF1 exige une fonction de hachage unidirectionnelle qui est identifiée dans le champ Paramètres de l'identifiant d'algorithme MGF1. Les mises en œuvre DOIVENT prendre en charge SHA-1 et PEUVENT prendre en charge n'importe laquelle des autres fonctions de hachage unidirectionnelles mentionnées au paragraphe 2.1. L'identifiant d'algorithme MGF1 est composé de l'identifiant d'objet id-mgf1 et d'un paramètre qui contient l'identifiant d'algorithme de la fonction de hachage unidirectionnelle employée avec MGF1. L'identifiant d'algorithme SHA-1 se compose de l'identifiant d'objet id-sha1 et d'un paramètre (facultatif) NULL. Les mises en œuvre qui effectuent la génération de signature DOIVENT omettre le champ maskGenAlgorithm lorsque MGF1 avec SHA-1 est utilisé, ce qui indique que l'algorithme par défaut a été utilisé.

Bien que mgf1SHA1Identifier soit défini comme valeur par défaut de ce champ, les mises en œuvre DOIVENT accepter que les deux codages de valeur par défaut (c'est-à-dire, un champ absent) et mgf1SHA1Identifier soient explicitement présents dans le codage.

saltLength
Le champ saltLength est la longueur en octets du sel. Pour un certain hashAlgorithm, la valeur recommandée de saltLength est le nombre d'octets dans la valeur du hachage. À la différence des autres champs de type RSASSA-PSS-params, saltLength n'a pas besoin d'être fixé pour une certaine paire de clés RSA ; une valeur différente pourrait être utilisée pour chaque signature RSASSA-PSS générée.

trailerField
Le champ trailerField est un entier. Il assure la compatibilité avec la norme IEEE 1363a-2004 [P1363A]. Sa valeur DOIT être 1, qui représente le champ de queue avec la valeur hexadécimale de 0xBC. Les autres champs de queue, y compris le champ de queue composé de HashID enchaîné avec 0xCC qui est spécifié dans la norme IEEE 1363a, ne sont pas acceptés. Les mises en œuvre qui effectuent la génération de signature DOIVENT omettre le champ trailerField, ce qui indique que la valeur de champ de queue par défaut a été utilisée. Les mises en œuvre qui effectuent la validation de signature DOIVENT reconnaître à la fois un champ trailerField présent avec la valeur 1 et un champ trailerField absent.

Si la valeur par défaut des champs hashAlgorithm, maskGenAlgorithm, et trailerField de RSASSA-PSS-params est utilisée,

l'identifiant d'algorithme aura alors la valeur suivante :

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 Valeurs de signature RSASSA-PSS

Le résultat de l'algorithme de signature RSASSA-PSS est une chaîne d'octets, qui a la même longueur en octets que le module n RSA.

Les valeurs de signature en CMS [RFC3852] sont représentées comme des chaînes d'octets, et le résultat est utilisé directement. Cependant, les valeurs de signature dans les certificats et les CRL [RFC3280] sont représentées comme des chaînes binaires, et une conversion est nécessaire.

Pour convertir une valeur de signature en une chaîne binaire, le bit de poids fort du premier octet de la valeur de signature DEVRA devenir le premier bit de la chaîne binaire, et ainsi de suite jusqu'au bit de moindre poids du dernier octet de la valeur de signature, qui DEVRA devenir le dernier bit de la chaîne binaire.

3.3 Validation des paramètres de signature RSASSA-PSS

Trois scénarios possibles de validation de paramètre existent pour les valeurs de signature RSASSA-PSS.

1. La clé est identifiée par l'identifiant d'algorithme rsaEncryption. Dans ce cas, aucune validation de paramètre n'est nécessaire.

2. La clé est identifiée par l'identifiant d'algorithme de signature id-RSASSA-PSS, mais le champ Paramètres est absent. Dans ce cas, aucune validation de paramètre n'est nécessaire.

3. La clé est identifiée par l'identifiant d'algorithme de signature id-RSASSA-PSS et les paramètres sont présents. Dans ce cas, tous les paramètres dans l'identifiant d'algorithme de structure de signature DOIVENT correspondre aux paramètres qui sont dans l'identifiant d'algorithme de structure de clé sauf le champ saltLength. Le champ saltLength dans les paramètres de signature DOIT être supérieur ou égal à celui du champ Paramètres de clé.

4. Algorithme de transport de clés RSAES-OAEP

La présente section décrit les conventions pour utiliser l'algorithme de transport de clés RSAES-OAEP avec le certificat X.509 Internet et le profil de CRL [RFC3280]. RSAES-OAEP est spécifié dans PKCS n° 1 version 2.1 [RFC3447]. Les cinq fonctions de hachage unidirectionnelles exposées au paragraphe 2.1 et la fonction de génération de gabarit décrite au paragraphe 2.2 peuvent être utilisées avec RSAES-OAEP. Les CA et applications conformes DOIVENT prendre en charge l'algorithme de transport de clés RSAES-OAEP en utilisant SHA-1. Les autres fonctions de hachage unidirectionnelles PEUVENT aussi être prises en charge.

Les CA qui produisent les certificats avec l'identifiant d'algorithme id-RSAES-OAEP DEVRAIENT exiger la présence des paramètres dans le champ publicKeyAlgorithms pour tous les certificats. Les entités qui utilisent un certificat avec une valeur de publicKeyAlgorithm de id-RSA-OAEP où les paramètres sont absents DEVRAIENT utiliser l'ensemble par défaut de paramètres pour RSAES-OAEP-params. Les entités qui utilisent un certificat avec une valeur de publicKeyAlgorithm de rsaEncryption DEVRAIENT utiliser l'ensemble par défaut de paramètres pour RSAES-OAEP-params.

4.1 Clés publiques RSAES-OAEP

Lorsque id-RSAES-OAEP est utilisé dans un AlgorithmIdentifier, les paramètres DOIVENT employer la syntaxe RSAES-OAEP-params. Les paramètres peuvent être absents ou présents lorsque utilisés comme informations de clé publique sujette. Les paramètres DOIVENT être présents lorsque utilisés dans l'identifiant d'algorithme associé à une valeur chiffrée.

```
IDENTIFIANT D'OBJET  id-RSAES-OAEP   ::=  { pkcs-1 7 }
  RSAES-OAEP-params  ::=  SEQUENCE  {
    hashFunc          [0] AlgorithmIdentifier DEFAUT sha1Identifier,
    maskGenFunc       [1] AlgorithmIdentifier DEFAUT mgf1SHA1Identifier,
```

```
    pSourceFunc       [2] AlgorithmIdentifier DEFAUT pSpecifiedEmptyIdentifier  }
  pSpecifiedEmptyIdentifier  AlgorithmIdentifier  ::= { id-pSpecified, nullOctetString }
    nullOctetString  CHAINE D'OCTETS (TAILLE (0))  ::=  { "H }
```

Les champs de type RSAES-OAEP-params ont la signification suivante :

hashFunc
Le champ hashFunc identifie la fonction de hachage unidirectionnelle. Il DOIT être un des identifiants d'algorithme mentionnés au paragraphe 2.1, et la fonction de hachage par défaut est SHA-1. Les mises en œuvre DOIVENT prendre en charge SHA-1 et PEUVENT prendre en charge d'autres fonctions de hachage unidirectionnelles mentionnées au paragraphe 2.1. Les mises en œuvre qui effectuent le chiffrement DOIVENT omettre le champ hashFunc lorsque SHA-1 est utilisé, ce qui indique que l'algorithme par défaut a été utilisé. Les mises en œuvre qui effectuent le déchiffrement DOIVENT reconnaître l'identifiant d'algorithme sha1Identifier et le champ hashFunc absent comme l'indication que SHA-1 a été utilisé.

maskGenFunc
Le champ maskGenFunc identifie la fonction de génération de gabarit. La fonction de génération de gabarit par défaut est MGF1 avec SHA-1. Pour MGF1, il est fortement RECOMMANDÉ que la fonction de hachage sous-jacente soit la même que celle identifiée par hashFunc. Les mises en œuvre DOIVENT prendre en charge MGF1. MGF1 exige une fonction de hachage unidirectionnelle qui est identifiée dans le champ Paramètre de l'identifiant d'algorithme MGF1. Les mises en œuvre DOIVENT prendre en charge SHA-1 et PEUVENT prendre en charge toute autre fonction de hachage unidirectionnelle mentionnée au paragraphe 2.1. L'identifiant d'algorithme MGF1 se compose de l'identifiant d'objet id-mgf1 et d'un paramètre qui contient l'identifiant d'algorithme de la fonction de hachage unidirectionnelle employée avec MGF1. L'identifiant d'algorithme SHA-1 se compose de l'identifiant d'objet id-sha1 et d'un paramètre (facultatif) NULL. Les mises en œuvre qui effectuent le chiffrement DOIVENT omettre le champ maskGenFunc lorsque MGF1 est utilisé avec SHA-1, ce qui indique que l'algorithme par défaut a été utilisé.

Bien que mgf1SHA1Identifier soit défini comme la valeur par défaut pour ce champ, les mises en œuvre DOIVENT accepter le codage de valeur par défaut (c'est-à-dire, un champ absent) et le mgf1SHA1Identifier explicitement présent dans le codage.

pSourceFunc
Le champ pSourceFunc identifie la source (et éventuellement la valeur) des paramètres de codage, généralement appelée P. Les mises en œuvre DOIVENT représenter P par un identifiant d'algorithme, id-pSpecified, qui indique que P est explicitement fourni comme une CHAÎNE D'OCTETS dans les paramètres. La valeur par défaut pour P est une chaîne vide. Dans ce cas, pHash dans EME-OAEP contient le hachage d'une chaîne de longueur zéro. Les mises en œuvre DOIVENT prendre en charge une valeur de P de longueur zéro. Les mises en œuvre qui effectuent le chiffrement DOIVENT omettre le champ pSourceFunc lorsque une valeur de P de longueur zéro est utilisée, ce qui indique que la valeur par défaut a été utilisée. Les mises en œuvre qui effectuent le déchiffrement DOIVENT reconnaître l'identifiant d'objet id-pSpecified et un champ pSourceFunc absent comme l'indication que la valeur de P de longueur zéro a été utilisée. Les mises en œuvre qui effectuent le déchiffrement DOIVENT prendre en charge une valeur de P de longueur zéro et PEUVENT prendre en charge d'autres valeurs. Les mises en œuvre conformes NE DOIVENT PAS utiliser d'autre valeur que id-pSpecified pour pSourceFunc.

Si les valeurs par défaut des champs hashFunc, maskGenFunc, et pSourceFunc de RSAES-OAEP-params sont utilisées, les identifiants d'algorithme auront alors la valeur suivante :

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

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

5. Algorithme de signature PKCS #1 version 1.5

La [RFC2313] spécifie l'algorithme de signature PKCS n° 1 Version 1.5. Cette spécification est aussi incluse dans PKCS n° 1 version 2.1 [RFC3447]. La [RFC3279] spécifie l'utilisation de l'algorithme de signature PKCS n° 1 version 1.5 avec les fonctions de hachage unidirectionnelles MD2, MD5, et SHA-1. Cette section spécifie les identifiants d'algorithme pour utiliser les fonctions de hachage unidirectionnelles SHA-224, SHA-256, SHA-384, et SHA-512 avec l'algorithme de signature PKCS n° 1 version 1.5.

L'algorithme de signature RSASSA-PSS est préféré à PKCS n° 1 version 1.5. Bien qu'aucune attaque ne soit connue contre l'algorithme de signature PKCS n° 1 version 1.5, dans l'intérêt d'une robustesse accrue, l'algorithme de signature RSASSA-PSS est recommandé pour une adoption éventuelle, en particulier par les nouvelles applications. Cette section est incluse pour la compatibilité avec les applications existantes, et bien que toujours appropriée pour les nouvelles applications, une transition graduelle vers l'algorithme de signature RSASSA-PSS est encouragée.

L'algorithme de signature PKCS n° 1 version 1.5 avec ces fonctions de hachage unidirectionnelles et le système de chiffrement RSA est mis en œuvre en utilisant les conventions de bourrage et de codage décrites dans la [RFC2313].

Le résumé de message est calculé en utilisant la fonction de hachage unidirectionnelle SHA-224, SHA-256, SHA-384, ou SHA-512.

L'algorithme de signature PKCS n° 1 version 1.5, tel que spécifié dans la RFC 2313, inclut une étape de codage des données. Dans cette étape sont combinés le résumé de message et l'identifiant d'objet pour la fonction de hachage unidirectionnelle utilisée pour calculer le résumé de message. Lorsque ils effectuent l'étape de codage des données, les identifiants d'objet id-sha224, id-sha256, id-sha384, et id-sha512 (voir au paragraphe 2.1) DOIVENT être utilisés pour spécifier les fonctions de hachage unidirectionnelles, respectivement SHA-224, SHA-256, SHA-384, et SHA-512.

L'identifiant d'objet utilisé pour identifier l'algorithme de signature PKCS n° 1 version 1.5 avec l'algorithme SHA-224 est :

IDENTIFIANT D'OBJET sha224WithRSAEncryption ::= { pkcs-1 14 }

L'identifiant d'objet utilisé pour identifier l'algorithme de signature PKCS n° 1 version 1.5 avec SHA-256 est :

IDENTIFIANT D'OBJET sha256WithRSAEncryption ::= { pkcs-1 11 }

L'identifiant d'objet utilisé pour identifier l'algorithme de signature PKCS n° 1 version 1.5 avec SHA-384 est :

IDENTIFIANT D'OBJET sha384WithRSAEncryption ::= { pkcs-1 12 }

L'identifiant d'objet utilisé pour identifier l'algorithme de signature PKCS n° 1 version 1.5 avec SHA-512 est :

IDENTIFIANT D'OBJET sha512WithRSAEncryption ::= { pkcs-1 13 }

Lorsque un de ces quatre identifiants d'objet apparaît dans un AlgorithmIdentifier, les paramètres DOIVENT être NULL. Les mises en œuvre DOIVENT accepter que les paramètres soient absents aussi bien que présents.

Le processus RSA de génération de signature et le codage du résultat sont décrits en détail dans la [RFC2313].

6. Module ASN.1

```
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) }

ÉTIQUETTES EXPLICITES DE DÉFINITIONS ::= DÉBUT

-- EXPORTE TOUT ;

IMPORTE

AlgorithmIdentifier
  FROM PKIX1Explicit88 – Provient de la [RFC3280]
  { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0) id-pkix1-explicit(18) } ;

-- ============================
--  Identifiants d'objet de base
-- ============================

IDENTIFIANT D'OBJETpkcs-1 ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 1 }

-- Lorsque rsaEncryption est utilisé dans un AlgorithmIdentifier, les paramètres DOIVENT être présents et DOIVENT être
   NULL.

IDENTIFIANT D'OBJET rsaEncryption ::= { pkcs-1 1 }
```

-- Lorsque id-RSAES-OAEP est utilisé dans un AlgorithmIdentifier, et que le champ Paramètres est présent, il DOIT être RSAES-OAEP-params.

IDENTIFIANT D'OBJET id-RSAES-OAEP ::= { pkcs-1 7 }

-- Lorsque id-pSpecified est utilisé dans un AlgorithmIdentifier les paramètres DOIVENT être une CHAÎNE D'OCTETS.

IDENTIFIANT D'OBJET id-pSpecified ::= { pkcs-1 9 }

-- Lorsque id-RSASSA-PSS est utilisé dans un AlgorithmIdentifier, et que le champ Paramètres est présent, il DOIT être RSASSA-PSS-params.

IDENTIFIANT D'OBJET id-RSASSA-PSS ::= { pkcs-1 10 }

-- Lorsque id-mgf1 est utilisé dans un AlgorithmIdentifier, le champ Paramètres DOIT être présent et DOIT être HashAlgorithm.

IDENTIFIANT D'OBJET id-mgf1 ::= { pkcs-1 8 }

-- Lorsque les OID suivants sont utilisés dans un AlgorithmIdentifier, le champ Paramètres DOIT être présent et DOIT être NULL.

IDENTIFIANT D'OBJET sha224WithRSAEncryption ::= { pkcs-1 14 }

IDENTIFIANT D'OBJETsha256WithRSAEncryption ::= { pkcs-1 11 }

IDENTIFIANT D'OBJET sha384WithRSAEncryption ::= { pkcs-1 12 }

IDENTIFIANT D'OBJET sha512WithRSAEncryption ::= { pkcs-1 13 }

-- Lorsque les OID suivants sont utilisés dans un AlgorithmIdentifier, le champ Paramètres DEVRAIT être absent, mais si il est présent, il DOIT être NULL.

IDENTIFIANT D'OBJET id-sha1 ::= { iso(1) identified-organization(3) oiw(14) secsig(3) algorithms(2) 26 }

IDENTIFIANT D'OBJET id-sha224 ::= { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3)
nistalgorithm(4) hashalgs(2) 4 }

IDENTIFIANT D'OBJET id-sha256 ::= { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3)
nistalgorithm(4) hashalgs(2) 1 }

IDENTIFIANT D'OBJET id-sha384 ::= { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3)
nistalgorithm(4) hashalgs(2) 2 }

IDENTIFIANT D'OBJET id-sha512 ::= { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3)
nistalgorithm(4) hashalgs(2) 3 }

-- ==========
-- Constantes
-- ==========

nullOctetString CHAÎNE D'OCTETS (TAILLE (0)) ::= "H

nullParameters NULL ::= NULL

-- ========================
-- Identifiants d'algorithme
-- ========================

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 rSASSA-PSS-SHA512-
                                                      params }

rSASSA-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 }

-- ====================
--  Principales structures
-- ====================

-- Utilisé dans les SubjectPublicKeyInfo de certificat X.509.

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

-- Paramètres de AlgorithmIdentifier pour id-RSASSA-PSS.
-- Noter que les étiquettes dans cette séquence sont explicites.

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

    HashAlgorithm ::= AlgorithmIdentifier

    MaskGenAlgorithm ::= AlgorithmIdentifier

-- Paramètres de AlgorithmIdentifier pour id-RSAES-OAEP.
-- Noter que les étiquettes dans cette séquence sont explicites.

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

FIN
```

7. Références

7.1 Références normatives

[RFC2119] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997.

[RFC2313] B. Kaliski, "PKCS n° 1 : Chiffrement RSA version 1.5", mars 1998.

[RFC3280] R. Housley, W. Polk, W. Ford et D. Solo, "Profil de certificat d'infrastructure de clé publique X.509 et de liste de révocation de certificat (CRL) pour l'Internet", avril 2002. *(Obsolète, voir* RFC5280)

[RFC3447] J. Jonsson et B. Kaliski, "Normes de cryptographie à clés publiques (PKCS) n° 1 : Spécifications de la

cryptographie RSA version 2.1", février 2003.

[RFC3874] R. Housley, "SHA-224 : une fonction de hachage unidirectionnelle à 224 bits", septembre 2004. *(Information)*

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

[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 Références pour information

[GUIDE] National Institute of Standards and Technology, Second Draft: "Key Management Guideline, Part 1: General Guidance". juin 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.

[RFC1750] D. Eastlake 3rd et autres, "Recommandations d'aléa pour la sécurité", décembre 1994. *(Info., remplacée par la* RFC4086)

[RFC3279] L. Bassham, W. Polk et R. Housley, "Algorithmes et identifiants pour le profil de certificat d'infrastructure de clé publique X.509 et de liste de révocation de certificat (CRL) pour l'Internet", avril 2002.

[RFC3852] R. Housley, "Syntaxe de message cryptographique (CMS)", juillet 2004. *(Remplacée par la* RFC5652)

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

8. Considérations sur la sécurité

La présente spécification complète la [RFC3280]. La section "Considérations sur la sécurité" de ce document s'applique aussi à la présente spécification.

Les mises en œuvre doivent protéger la clé privée RSA. La compromission de la clé privée RSA peut résulter en la divulgation de tous les messages protégés avec cette clé.

La génération de la paire de clés publique/privée RSA s'appuie sur des nombres aléatoires. Utiliser des générateurs de nombres pseudo aléatoires (PRNG, *pseudo-random number generator*) inappropriés pour générer des clés de chiffrement peut résulter en peu ou pas du tout de sécurité. Un attaquant peut trouver beaucoup plus facile de reproduire l'environnement du PRNG qui a produit les clés et chercher dans le petit ensemble de possibilités résultant, que de faire une recherche en force brute sur l'ensemble de clés total. La génération de nombres aléatoire de qualité est difficile et la [RFC1750] offre des lignes directrices importantes dans ce domaine.

Généralement, les bonnes pratiques cryptographiques emploient une certaine paire de clés RSA dans un seul schéma. Cette pratique évite le risque qu'une vulnérabilité dans un schéma puisse compromettre la sécurité des autres, et peut être essentielle pour maintenir une sécurité démontrable. Bien que PKCS n°1 version 1.5 [RFC2313] ait été employé pour le transport de clé et la signature numérique sans aucune mauvaise interaction connue, une utilisation combinée d'une paire de clés RSA n'est pas recommandée à l'avenir. Donc, une paire de clés RSA utilisée pour la génération de signature RSASSA-PSS ne devrait pas être utilisée pour d'autres objets. Pour des raisons similaires, une paire de clés RSA devrait toujours être utilisée avec les mêmes paramètres RSASSA-PSS (sauf éventuellement la longueur de sel). De même, une paire de clés RSA utilisée pour le transport de clé RSAES-OAEP ne devrait pas être utilisée pour d'autres objets. Pour des raisons similaires, une paire de clés RSA devrait toujours être utilisée avec les mêmes paramètres RSAES-OAEP.

La présente spécification exige que les mises en œuvre prennent en charge la fonction de hachage SHA-1 unidirectionnelle pour l'interopérabilité, mais la prise en charge d'autres fonctions de hachage unidirectionnelles est permise. Wang et autres [SHA-1] ont récemment découvert une attaque de collision contre SHA-1 avec une complexité de 2^69. Cette attaque, qui peut produire deux nouveaux messages avec la même valeur de hachage, est la première attaque contre SHA-1 plus rapide que l'attaque générique d'une complexité de 2^80, où 80 est la moitié de la longueur binaire de la valeur du hachage.

En général, lorsque une fonction de hachage unidirectionnelle est utilisée avec un schéma de signature numérique, une attaque de collision est facilement traduite en une fausse signature. Donc, utiliser SHA-1 dans un schéma de signature numérique ne donne pas un niveau de sécurité supérieur à 69 bits si l'attaquant peut persuader le signataire de signer un message résultant d'une attaque de collision. Si l'attaquant ne peut pas persuader le signataire de signer un tel message, SHA-1 fournit cependant quand même un niveau de sécurité d'au moins 80 bits car la meilleure attaque (connue) d'inversion (qui produit un nouveau message avec une valeur de hachage précédente) est l'attaque générique d'une complexité de 2^160. Si on désire un niveau de sécurité supérieur, on a alors besoin d'une fonction de hachage unidirectionnelle sûre d'une plus longue valeur de hachage. SHA-256, SHA-384, et SHA-512 sont des choix raisonnables [SHA2], bien que leur sécurité ait besoin d'être reconfirmée à la lumière des résultats de SHA-1.

Les métriques de choix d'une fonction de hachage unidirectionnelle à utiliser dans les signatures numériques ne s'appliquent pas directement à l'algorithme de transport de clé RSAES-OAEP, car une attaque de collision sur la fonction de hachage unidirectionnelle ne se traduit pas directement en une attaque sur l'algorithme de transport de clé, sauf si les paramètres de codage de P varient (auquel cas une collision de la valeur du hachage pour les différents paramètres de codage pourrait être exploitée).

Néanmoins, pour la cohérence avec la pratique des schémas de signature numérique, et au cas où le codage du paramètre P n'est pas la chaîne vide, il est recommandé que la même règle d'approximation soit appliquée pour choisir une fonction de hachage unidirectionnelle à utiliser avec RSAES-OAEP. C'est-à-dire que la fonction de hachage unidirectionnelle devrait être choisie de façon à ce que la longueur binaire de la valeur du hachage soit en bits au moins deux fois celle du niveau de sécurité désiré.

La taille de clé choisie impacte la force réalisée lors de la mise en œuvre de services cryptographiques. Donc, le choix de tailles de clés appropriées est critique pour mettre en œuvre la sécurité appropriée. Une clé publique RSA de 1024 bits est considérée fournir un niveau de sécurité d'environ 80 bits. Dans [GUIDE], l'Institut National des normes et technologies (NIST, *National Institute of Standards and Technology*) suggère qu'un niveau de sécurité de 80 bits est adéquat pour la protection d'information sensibles jusqu'en 2015. Cette recommandation sera vraisemblablement révisée sur la base des avancées récentes, et est estimée être plus prudente, suggérant qu'un niveau de sécurité de 80 bits est une protection adéquate d'information sensibles jusqu'en 2010. Si un niveau de sécurité supérieur à 80 bits est nécessaire, une clé publique RSA plus longue et une fonction de hachage unidirectionnelle sûre d'une plus grande longueur de valeur de hachage est nécessaire. SHA-224, SHA-256, SHA-384, et SHA-512 sont des choix raisonnables pour une telles fonction de hachage unidirectionnelle, modulo la reconfirmation notée plus haut. Pour cette raison, les identifiants d'algorithme pour ces fonctions de hachage unidirectionnelles sont inclus dans le module ASN.1 de la Section 6.

Les mises en œuvre actuelles DOIVENT prendre en charge les tailles de clé publique RSA de 1024 bits. Avant la fin de 2007, les mises en œuvre DEVRAIENT prendre en charge de tailles de clé publique RSA d'au moins 2048 bits et DEVRAIENT prendre en charge SHA-256. Cette exigence est destinée à permettre un délai adéquat pour que les utilisateurs déploient une plus forte capacité de signature numérique dès 2010.

Lorsque on utilise RSASSA-PSS, la même fonction de hachage unidirectionnelle devrait être employée pour le hashAlgorithm et le maskGenAlgorithm, mais ce n'est pas exigé. Lorsque on utilisé RSAES-OAEP, la même fonction de hachage unidirectionnelle devrait être employée pour le hashFunc et le maskGenFunc, mais ce n'est pas exigé. Dans chaque cas, utiliser la même fonction de hachage unidirectionnelle aide à l'analyse de la sécurité et réduit la complexité de mise en œuvre.

9. Considérations relatives à l'IANA

Dans les certificats et les CRL, les algorithmes sont identifiés par des identifiants d'objet. Tous les identifiants d'objet utilisés dans le présent document ont été alloués dans des documents de normes de cryptographie à clé publique (PKCS, *Public-Key Cryptography Standards*) ou par l'Institut national des normes et technologies (NIST, *National Institute of Standards and Technology*). Aucune autre action de l'IANA n'est nécessaire pour le présent document ou ses futures mises à jour.

Adresse des auteurs

Russell Housley
Vigil Security, LLC
918 Spring Knoll Drive
Herndon, VA 20170
USA
mél : housley@vigilsec.com

Burt Kaliski
RSA Laboratories
174 Middlesex Turnpike
Bedford, MA 01730
USA
mél : bkaliski@rsasecurity.com

Jim Schaad
Soaring Hawk Consulting
PO Box 675
Gold Bar, WA 98251
USA
mél : jimsch@exmsft.com

Déclaration de droits de reproduction

Propriété intellectuelle

L'IETF ne prend pas position sur la validité et la portée de tout droit de propriété intellectuelle ou autres droits qui pourraient être revendiqués au titre de la mise en œuvre ou l'utilisation de la technologie décrite dans le présent document ou sur la mesure dans laquelle toute licence sur de tels droits pourrait être ou n'être pas disponible ; pas plus qu'elle ne prétend avoir accompli aucun effort pour identifier de tels droits. Les informations sur les procédures de l'ISOC au sujet des droits dans les documents de l'ISOC figurent dans les BCP 78 et BCP 79.

Des copies des dépôts d'IPR faits au secrétariat de l'IETF et toutes assurances de disponibilité de licences, ou le résultat de tentatives faites pour obtenir une licence ou permission générale d'utilisation de tels droits de propriété par ceux qui mettent en œuvre ou utilisent la présente spécification peuvent être obtenues sur répertoire en ligne des IPR de l'IETF à http://www.ietf.org/ipr.

L'IETF invite toute partie intéressée à porter son attention sur tous copyrights, licences ou applications de licence, ou autres droits de propriété qui pourraient couvrir les technologies qui peuvent être nécessaires pour mettre en œuvre la présente norme. Prière d'adresser les informations à l'IETF à ietf-ipr@ietf.org.

Remerciement

Le financement de la fonction d'édition des RFC est actuellement assuré par la Internet Society.