Groupe de travail Réseau
Request for Comments : 3370
RFC rendues obsolètes : 2630, 3211
Catégorie : En cours de normalisation

R. Housley
RSA Laboratories
août 2002
Traduction Claude Brière de L'Isle

Algorithmes de la syntaxe de message cryptographique (CMS)

Résumé
Le présent document décrit les conventions pour utiliser plusieurs algorithmes cryptographiques avec la syntaxe de message cryptographique (CMS, *Cryptographic Message Syntax*). La CMS est utilisée pour la signature numérique, le résumé, l'authentification, ou le chiffrement de contenus de messages arbitraires.

Table des matières

1. Introduction

La syntaxe de message cryptographique (CMS) [RFC3369] est utilisée pour signer numériquement, résumer, authentifier, ou chiffrer des contenus arbitraires de message. Cette spécification d'accompagnement décrit l'utilisation des algorithmes cryptographiques courants avec la CMS. Les mises en œuvre de la CMS peuvent prendre en charge ces algorithmes ; les mises en œuvre de la CMS peuvent aussi bien prendre en charge d'autres algorithmes. Cependant, si une mise en œuvre choisit de prendre en charge un des algorithmes exposés dans le présent document, cette mise en œuvre DOIT le faire comme décrit dans le présent document.

Les valeurs de CMS sont générées en utilisant l'ASN.1 [X.208-88], en utilisant le codage BER [X.209-88]. Les identifiants

d'algorithme (qui incluent des identifiants d'objets ASN.1) identifient des algorithmes cryptographiques, et certains algorithmes exigent des paramètres supplémentaires. Lorsque nécessaire, les paramètres sont spécifiés avec une structure ASN.1. L'identifiant d'algorithme pour chaque algorithme est spécifié, et lorsque nécessaire, la structure du paramètre est spécifiée. Les champs dans la CMS employée par chaque algorithme sont identifiés.

1.1 Changements par rapport à la RFC2630

Le présent document rend obsolète la section 12 de la [RFC2630]. La [RFC3369] rend obsolète le reste de la RFC2630. La séparation des spécifications du protocole et de l'algorithme permet que chacun soit mis à jour sans avoir d'impact sur l'autre. Cependant, les conventions pour l'utilisation d'algorithmes supplémentaires avec la CMS seront vraisemblablement à spécifier dans d'autres documents.

1.2 Terminologie

Dans le présent document, les mots clés DOIT, NE DOIT PAS, EXIGE, DEVRAIT, NE DEVRAIT PAS, RECOMMANDE, et PEUT sont à interpréter comme décrit dans la [RFC2119].

2. Algorithmes de résumé de message

Cette section spécifie les conventions employées par les mises en œuvre de CMS qui prennent en charge SHA-1 ou MD5.

Les identifiants d'algorithme de résumé sont localisés dans le champ digestAlgorithms de SignedData, le champ digestAlgorithm de SignerInfo, le champ digestAlgorithm de DigestedData, et le champ digestAlgorithm de AuthenticatedData.

Les valeurs de résumé sont localisées dans le champ Résumé de DigestedData et dans l'attribut Authentifié du résumé de message. De plus, les valeurs de résultat sont des entrées des algorithmes de signature.

2.1 SHA-1

L'algorithme de résumé de message SHA-1 est défini dans la publication FIPS 180-1 [SHA1]. L'identifiant d'algorithme pour SHA-1 est :

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

Il y a deux codages possibles pour le champ de paramètres AlgorithmIdentifier de SHA-1. L'alternative vient du fait que lorsque la syntaxe de 1988 pour AlgorithmIdentifier a été traduite dans la syntaxe de 1997, le FACULTATIF associé aux paramètres AlgorithmIdentifier a été perdu. Plus tard, le FACULTATIF a été récupéré suite à un rapport d'erreur, mais alors de nombreuses personnes pensaient que les paramètres de l'algorithme étaient obligatoires. À cause de cette histoire, certaines mises en œuvre codent les paramètres comme un élément NUL, et d'autres les omettent entièrement. Le codage correct est d'omettre le champ Paramètres ; cependant, les mises en œuvre DOIVENT aussi traiter un champ Paramètres de AlgorithmIdentifier de SHA-1 qui contient un NUL.

Le champ Paramètres de AlgorithmIdentifier est FACULTATIF. Si il est présent, le champ Paramètres DOIT contenir un NUL. Les mises en œuvre DOIVENT accepter les AlgorithmIdentifiers de SHA-1 avec Paramètres absent. Les mises en œuvre DOIVENT accepter les AlgorithmIdentifiers SHA-1 avec Paramètres à NUL. Les mises en œuvre DEVRAIENT générer le AlgorithmIdentifiers SHA-1 avec Paramètres absent.

2.2 MD5

L'algorithme de résumé de message MD5 est défini dans la [RFC1321]. L'identifiant d'algorithme pour MD5 est :

IDENTIFIANT D'OBJET md5 ::= { iso(1) member-body(2) us(840) rsadsi(113549) digestAlgorithm(2) 5 }

Le champ Paramètres de AlgorithmIdentifier DOIT être présent, et le champ Paramètres DOIT contenir NUL. Les mises en œuvre PEUVENT accepter les AlgorithmIdentifier de MD5 avec Paramètres absent aussi bien que Paramètres NUL.

3. Algorithmes de signature

La présente section spécifie les conventions employées par les mises en œuvre de CMS qui prennent en charge DSA ou RSA (PKCS n° 1 v1.5).

Les identifiants d'algorithme de signature sont localisés dans le champ signatureAlgorithm de SignerInfo dans SignedData. Aussi, les identifiants d'algorithme de signature sont localisés dans le champ signatureAlgorithm de SignerInfo des attributs de contreseing.

Les valeurs de signature sont localisées dans le champ Signature de SignerInfo des SignedData. Aussi, les valeurs de signature sont localisées dans le champ Signature de SignerInfo des attributs de contreseing.

3.1 DSA

L'algorithme de signature DSA est défini dans la publication FIPS 186 [DSS]. DSA DOIT être utilisé avec l'algorithme de résumé de message SHA-1.

L'identifiant d'algorithme pour les clés publiques de sujet DSA dans les certificats est :

IDENTIFIANT D'OBJET id-dsa ::= { iso(1) member-body(2) us(840) x9-57 (10040) x9cm(4) 1 }

La validation de signature DSA exige trois paramètres, habituellement appelés p, q, et g. Lorsque l'identifiant d'algorithme id-dsa est utilisé, le champ de paramètres AlgorithmIdentifier est facultatif. S'il est présent, le champ de paramètres AlgorithmIdentifier DOIT contenir les trois valeurs de paramètre DSA codées en utilisant le type Dss-Parms. S'il est absent, la clé publique DSA sujette utilise les mêmes paramètres DSA que le producteur du certificat.

```
Dss-Parms ::= SEQUENCE {
    p   ENTIER,
    q   ENTIER,
    g   ENTIER  }
```

Lorsque l'identifiant d'algorithme id-dsa est utilisé, la clé publique DSA, appelé commuément Y, DOIT être codée comme un ENTIER. Le résultat de ce codage est porté dans la clé publique de sujet de certificat.

```
Dss-Pub-Key ::= ENTIER                                  -- Y
```

L'identifiant d'algorithme pour DSA avec les valeurs de signature SHA-1 est :

IDENTIFIANT D'OBJET id-dsa-with-sha1 ::= { iso(1) member-body(2) us(840) x9-57 (10040) x9cm(4) 3 }

Lorsque l'identifiant d'algorithme id-dsa-with-sha1 est utilisé, le champ Paramètres d'AlgorithmIdentifier DOIT être absent.

Lors de la signature, l'algorithme DSA génère deux valeurs, communément appelées r et s. Pour transférer ces deux valeurs comme une seule signature, elles DOIVENT être codées en utilisant le type Dss-Sig-Value :

```
Dss-Sig-Value ::= SEQUENCE {
    r   ENTIER,
    s   ENTIER }
```

3.2 RSA

L'algorithme de signature RSA (PKCS n° 1 v1.5) est défini dans la [RFC2437]. La RFC2437 spécifie l'utilisation de l'algorithme de signature RSA avec les algorithmes de résumé de message SHA-1 et MD5.

L'identifiant d'algorithme pour les clés publiques sujettes RSA dans les certificats est :

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

Lorsque l'identifiant d'algorithme rsaEncryption est utilisé, le champ Paramètres AlgorithmIdentifier DOIT contenir NUL.

Lorsque l'identifiant d'algorithme rsaEncryption est utilisé, la clé publique RSA, qui est composée d'un module et d'un

exposant public, DOIT être codée en utilisant le type RSAPublicKey. Le résultat de ce codage est porté dans la clé publique de sujet de certificat.

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

Les mises en œuvre de CMS qui incluent l'algorithme de signature RSA (PKCS n° 1 v1.5) DOIVENT aussi mettre en œuvre l'algorithme de résumé de message SHA-1. De telles mises en œuvre DEVRAIENT aussi prendre en charge l'algorithme de résumé de message MD5.

L'identifiant d'algorithme rsaEncryption est utilisé pour identifier les valeurs de signature RSA (PKCS n° 1 v1.5) sans considération de l'algorithme de résumé de message employé. Les mises en œuvre de CMS qui incluent l'algorithme de signature RSA (PKCS n° 1 v1.5) DOIVENT prendre en charge l'identifiant d'algorithme de valeur de signature rsaEncryption, et les mises en œuvre de CMS PEUVENT prendre en charge les identifiants d'algorithme de valeur de signature RSA (PKCS n° 1 v1.5) qui spécifient à la fois l'algorithme de signature RSA (PKCS n° 1 v1.5) et l'algorithme de résumé de message.

L'identifiant d'algorithme pour RSA (PKCS n° 1 v1.5) avec les valeurs de signature SHA-1 est :

```
IDENTIFIANT D'OBJET sha1WithRSAEncryption ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
                                                pkcs-1(1) 5 }
```

L'identifiant d'algorithme pour RSA (PKCS n° 1 v1.5) avec les valeurs de signature MD5 est :

```
IDENTIFIANT D'OBJET md5WithRSAEncryption  ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
                                                pkcs-1(1) 4 }
```

Lorsque les identifiants d'algorithme de valeur de signature rsaEncryption, sha1WithRSAEncryption, ou md5WithRSAEncryption sont utilisés, le champ de paramètres AlgorithmIdentifier DOIT être NUL.

Lors de la signature, l'algorithme RSA génère une seule valeur, et cette valeur est utilisée directement comme valeur de signature.

4. Algorithmes de gestion de clé

CMS s'accommode des techniques générales de gestion de clés suivantes : accord de clé, transport de clé, clés de chiffrement de clés symétriques distribuées au préalable, et mots de passe.

4.1 Algorithmes d'accord de clé

Cette section spécifie les conventions employées par les mises en œuvre de CMS qui prennent en charge l'accord de clés en utilisant le Diffie-Hellman éphémère-statique X9.42 (X9.42 E-S D-H) et le Diffie-Hellman statique-statique X9.42 (X9.42 S-S D-H).

Lorsque un algorithme d'accord de clés est utilisé, un algorithme de chiffrement de clé est aussi nécessaire. Donc, lorsque l'accord de clé est pris en charge, un algorithme de chiffrement de clé DOIT être fourni pour chaque algorithme de chiffrement de contenu. Les algorithmes d'enveloppe de clé pour le Triple-DES et RC2 sont décrits dans la [RFC3217].

Pour l'accord de clé de clé de chiffrement de clés RC2, 128 bits DOIVENT être générés comme entrée au processus d'expansion de clé utilisé pour calculer la clé RC2 efficace [RFC2268].

Les identifiants d'algorithme d'accord de clé sont situés dans les champs EnvelopedData RecipientInfos KeyAgreeRecipientInfo keyEncryptionAlgorithm et AuthenticatedData RecipientInfos KeyAgreeRecipientInfo keyEncryptionAlgorithm.

Les identifiants d'algorithme d'enveloppe de clé sont situés dans les paramètres KeyWrapAlgorithm au sein des champs EnvelopedData RecipientInfos KeyAgreeRecipientInfo keyEncryptionAlgorithm et AuthenticatedData RecipientInfos KeyAgreeRecipientInfo keyEncryptionAlgorithm.

Les clés de chiffrement de contenu enveloppées sont situées dans le champ EnvelopedData RecipientInfos

KeyAgreeRecipientInfo RecipientEncryptedKeys encryptedKey. Les clés d'authentification de message enveloppées sont situées dans le champ AuthenticatedData RecipientInfos KeyAgreeRecipientInfo RecipientEncryptedKeys encryptedKey.

4.1.1 Diffie-Hellman X9.42 éphémère-statique

L'accord de clé Diffie-Hellman éphémère-statique est défini dans la [RFC2631]. Lorsque on utilise le Diffie-Hellman éphémère-statique, le champ EnvelopedData RecipientInfos KeyAgreeRecipientInfo est utilisé comme suit :

version DOIT être 3.

originator DOIT être la solution originatorKey. Le champ d'algorithme originatorKey DOIT contenir l'identifiant d'objet dh-public-number avec les paramètres absents. Le champ originatorKey publicKey DOIT contenir la clé publique éphémère de l'envoyeur. L'identifiant d'objet dh-public-number est :

IDENTIFIANT D'OBJET dh-public-number ::= { iso(1) member-body(2) us(840) ansi-x942(10046) number-type(2) 1 }

ukm peut être présent ou absent. Les mises en œuvre de CMS DOIVENT accepter que ukm soit absent, et les mises en œuvre de CMS DEVRAIENT accepter que ukm soit présent. Lorsqu'il est présent, ukm est utilisé pour s'assurer qu'une clé de chiffrement de clé différente est générée lorsque la clé privée éphémère pourrait être utilisée plus d'une fois.

keyEncryptionAlgorithm DOIT être l'identifiant d'algorithme id-alg-ESDH. Le champ paramètre d'identifiant d'algorithme pour id-alg- ESDH est KeyWrapAlgorithm, et ce paramètre DOIT être présent. KeyWrapAlgorithm note l'algorithme de chiffrement symétrique utilisé pour chiffrer la clé de chiffrement de contenu avec la paire de clés de chiffrement de clé générée en utilisant l'algorithme d'accord de clés Diffie-Hellman éphémère-statique de X9.42. Les algorithmes Triple-DES et RC2 d'enveloppe de clé sont décrits dans la [RFC3217]. L'identifiant d'algorithme id-alg-ESDH et la syntaxe des paramètres sont :

IDENTIFIANT D'OBJET id-alg-ESDH ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) alg(3) 5 }

KeyWrapAlgorithm ::= AlgorithmIdentifier

recipientEncryptedKeys contient un identifiant et une clé chiffrée pour chaque receveur. RecipientEncryptedKey KeyAgreeRecipientIdentifier DOIT contenir soit le issuerAndSerialNumber qui identifie le certificat du receveur, soit le RecipientKeyIdentifier qui contient l'identifiant de clé de sujet provenant du certificat du receveur. Dans les deux cas, le certificat du receveur contient la clé publique statique du receveur. RecipientEncryptedKey EncryptedKey DOIT contenir la clé de chiffrement de contenu chiffrée avec la paire de clés de chiffrement de clé Diffie-Hellman éphémère-statique X9.42 générée en utilisant l'algorithme spécifié par KeyWrapAlgorithm.

4.1.2 Diffie-Hellman X9.42 statique-statique

L'accord de clé Diffie-Hellman statique-statique est défini dans la [RFC2631]. Lorsque on utilise Diffie-Hellman statique-statique, les champs EnvelopedData RecipientInfos KeyAgreeRecipientInfo et AuthenticatedData RecipientInfos KeyAgreeRecipientInfo sont utilisés comme suit :

version DOIT être 3.

originator DOIT être la solution issuerAndSerialNumber ou la solution subjectKeyIdentifier. Dans les deux cas, le certificat de l'origine contient la clé publique de l'envoyeur. La [RFC3279] spécifie la syntaxe et les valeurs des paramètres de AlgorithmIdentifier qui sont inclus dans le certificat de l'origine. Le champ d'informations de la clé publique sujette du certificat de l'origine DOIT contenir l'identifiant d'objet :

IDENTIFIANT D'OBJET dh-public-number ::= { iso(1) member-body(2) us(840) ansi-x942(10046) number-type(2) 1 }

ukm DOIT être présent. Le ukm assure qu'une clé de chiffrement de clé différente est générée pour chaque message entre le même envoyeur et receveur.

keyEncryptionAlgorithm DOIT être l'identifiant d'algorithme id-alg-SSDH. Le champ de paramètre d'identifiant d'algorithme pour id-alg-SSDH est KeyWrapAlgorihtm, et ce paramètre DOIT être présent. KeyWrapAlgorithm note l'algorithme de chiffrement symétrique utilisé pour chiffrer la clé de chiffrement de contenu avec la paire de clés de

chiffrement de clé générée en utilisant l'algorithme d'accord de clé Diffie-Hellman statique-statique X9.42. Les algorithmes d'enveloppe de clé Triple-DES et RC2 sont décrits dans la [RFC3217]. L'identifiant d'algorithme et la syntaxe de paramètre de id-alg-SSDH sont :

IDENTIFIANT D'OBJET id-alg-SSDH ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) alg(3) 10 }

KeyWrapAlgorithm ::= AlgorithmIdentifier

recipientEncryptedKeys contient un identifiant et une clé chiffrée pour chaque receveur. RecipientEncryptedKey KeyAgreeRecipientIdentifier DOIT contenir soit le issuerAndSerialNumber qui identifie le certificat du receveur, soit le RecipientKeyIdentifier qui contient l'identifiant de clé sujette venant du certificat du receveur. Dans les deux cas, le certificat du receveur contient la clé publique statique du receveur. RecipientEncryptedKey EncryptedKey DOIT contenir la clé de chiffrement de contenu chiffrée avec la clé de chiffrement de clé générée en paire par le Diffie-Hellman statique-statique X9.42 en utilisant l'algorithme spécifié par le KeyWrapAlgortihm.

4.2 Algorithmes de transport de clé

Ce paragraphe spécifie les conventions employées par les mises en œuvre de CMS qui prennent en charge le transport de clé en utilisant RSA (PKCS n° 1 v1.5).

Les identifiants d'algorithme de transport de clé sont situés dans le champ EnvelopedData RecipientInfos KeyTransRecipientInfo keyEncryptionAlgorithm.

Les clés chiffrées de chiffrement de contenu de transport de clé sont situées dans le champ EnvelopedData RecipientInfos KeyTransRecipientInfo encryptedKey.

4.2.1 RSA (PKCS n° 1 v1.5)

L'algorithme de transport de clé RSA est le schéma de chiffrement RSA défini dans la [RFC2313] ; le type de bloc est 02, et le message à chiffrer est la clé de chiffrement de contenu. L'identifiant d'algorithme pour RSA (PKCS n° 1 v1.5) est :

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

Le champ de paramètres AlgorithmIdentifier DOIT être présent, et le champ paramètres DOIT contenir NUL.

Lorsque on utilise une clé de chiffrement de contenu Triple-DES, les mises en œuvre de CMS DOIVENT ajuster les bits de parité pour chaque clé DES comprenant la clé Triple-DES avant le chiffrement RSA.

L'utilisation du chiffrement RSA (PKCS n° 1 v1.5) comme défini dans la [RFC2313], pour assurer la confidentialité, a une faiblesse connue. La faiblesse relève principalement de l'usage dans des applications interactives plutôt que dans des environnements à remise différée. Des informations complémentaires et des propositions de contre mesures sont exposées dans la section sur les considérations pour la sécurité (Section 9) de ce document et dans la [RFC3218].

Noter que le même schéma de chiffrement RSA est aussi défini dans la [RFC2437]. Dans la RFC2437, ce schéma de chiffrement RSA est appelé RSAES-PKCS1-v1_5.

4.3 Algorithmes de clé symétrique de chiffrement de clé

Ce paragraphe spécifie les conventions employées par les mises en œuvre de CMS qui prennent en charge la gestion de clé de chiffrement de clé symétrique utilisant les clés de chiffrement de clé Triple-DES ou RC2. Lorsque RC2 est pris en charge, les clés RC2 à 128 bits DOIVENT être utilisées comme clés de chiffrement de clé, et elles DOIVENT être utilisées avec le paramètre RC2ParameterVersion réglé à 58. Une mise en œuvre de CMS PEUT accepter des algorithmes mixtes de chiffrement de clé et de chiffrement de contenu. Par exemple, une clé RC2 de chiffrement de contenu de 40 bits PEUT être enveloppée avec une clé de chiffrement de clé Triple-DES de 168 bits ou une clé de chiffrement de clé RC2 de 128 bits.

Les identifiants d'algorithme d'enveloppe de clé sont situés dans les champs EnvelopedData RecipientInfos KEKRecipientInfo keyEncryptionAlgorithm et AuthenticatedData RecipientInfos KEKRecipientInfo keyEncryptionAlgorithm.

Les clés de chiffrement de contenu enveloppées sont situées dans le champ EnvelopedData RecipientInfos KEKRecipientInfo encryptedKey. Les clés d'authentification de message enveloppées sont situées dans le champ

AuthenticatedData RecipientInfos KEKRecipientInfo encryptedKey.

Le résultat d'un algorithme d'accord de clés est une clé de chiffrement de clé, et cette clé de chiffrement de clé est utilisée pour chiffrer la clé de chiffrement de contenu. Pour prendre en charge l'accord de clé, les identifiants d'algorithme d'enveloppe de clé sont situés dans le paramètre KeyWrapAlgorithm des champs EnvelopedData RecipientInfos KeyAgreeRecipientInfo keyEncryptionAlgorithm et AuthenticatedData RecipientInfos KeyAgreeRecipientInfo keyEncryptionAlgorithm. Cependant, seuls les algorithmes d'accord de clés qui fournissent par nature l'authentification devraient être utilisés avec AuthenticatedData. Les clés de chiffrement de contenu enveloppées sont situées dans le champ EnvelopedData RecipientInfos KeyAgreeRecipientInfo RecipientEncryptedKeys encryptedKey, les cles enveloppées d'authentification de message sont situées dans le champ AuthenticatedData RecipientInfos KeyAgreeRecipientInfo RecipientEncryptedKeys encryptedKey.

4.3.1 Enveloppe de clé Triple-DES

Une mise en œuvre de CMS PEUT accepter des algorithmes mixtes de chiffrement de clé et de chiffrement de contenu. Par exemple, une clé de chiffrement de contenu RC2 de 128 bits PEUT être enveloppée avec une clé de chiffrement de clé Triple-DES de 168 bits.

L'identifiant d'algorithme de chiffrement de clé Triple-DES est :

IDENTIFIANT D'OBJET id-alg-CMS3DESwrap ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) alg(3) 6 }

Le champ Paramètres de AlgorithmIdentifier DOIT être NUL.

L'algorithme d'enveloppe de clé utilisé pour chiffrer une clé de chiffrement de contenu Triple-DES avec une clé de chiffrement de clé Triple-DES est spécifié au paragraphe 3.1 de la [RFC3217]. L'algorithme correspondant de développement de clé est spécifié au paragraphe 3.2 de la [RFC3217].

La distribution hors bande de la clé de chiffrement de clé Triple-DES utilisée pour chiffrer la clé de chiffrement de contenu Triple-DES sort du domaine d'application du présent document.

4.3.2 Enveloppe de clé RC2

Une mise en œuvre de CMS PEUT accepter des algorithmes mixtes de chiffrement de clé et de chiffrement de conten. Par exemple, une clé de chiffrement de contenu RC2 à 128 bits PEUT être enveloppée avec une clé de chiffrement de clé Triple-DES de 168 bits. De même, une clé de chiffrement de contenu RC2 à 40 bits PEUT être enveloppée avec une clé de chiffrement de clé RC2 de 128 bits.

Le chiffrement de clé RC2 a l'identifiant d'algorithme :

IDENTIFIANT D'OBJET id-alg-CMSRC2wrap ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) alg(3) 7 }

Le champ de paramètre AlgorithmIdentifier DOIT être RC2wrapParameter :

RC2wrapParameter ::= RC2ParameterVersion

RC2ParameterVersion ::= ENTIER

Les bits de clé efficaces RC2 (la taille de clé) supérieurs à 32 et inférieurs à 256 sont codés en RC2ParameterVersion. Pour les bits de clé efficaces de 40, 64, et 128, les valeurs de rc2ParameterVersion sont respectivement de 160, 120, et 58. Ces valeurs ne sont pas simplement la longueur de clé RC2. Noter que la valeur 160 doit être codée avec deux octets (00 A0), parce que le codage sur un octet (A0) représente un nombre négatif.

Les clés RC2 de 128 bits DOIVENT être utilisées comme clé de chiffrement de clés, et elles DOIVENT être utilisées avec le paramètre RC2ParameterVersion réglé à 58.

L'algorithme d'enveloppe de clé utilisé pour chiffrer une clé de chiffrement de contenu RC2 avec une clé de chiffrement de clé RC2 est spécifié au paragrapohe 4.1 de la [RFC3217]. L'algorithme correspondant de développement de clé est spécifié au paragraphe 4.2 de la [RFC3217].

La distribution hors bande de clé de chiffrement de clé RC2 utilisée pour chiffrer la clé de chiffrement de contenu RC2 sort du domaine d'application du présent document.

4.4 Algorithmes de déduction de clé

Ce paragraphe spécifie les conventions employées par les mises en œuvre de CMS qui prennent en charge la gestion de clé fondée sur le mot de passe en utilisant PBKDF2.

Les algorithmes de déduction de clé sont utilisés pour convertir un mot de passe en une clé de chiffrement de clé au titre de la technique de gestion de clé fondée sur le mot de passe.

Les identifiantd d'algorithme de déduction de clé sont situés dans les champs EnvelopedData RecipientInfos PasswordRecipientInfo keyDerivationAlgorithm et AuthenticatedData RecipientInfos PasswordRecipientInfo keyDerivationAlgorithm.

La clé de chiffrement de clé qui est déduite du mot de passe est utilisée pour chiffrer la clé de chiffrement de contenu.

Les clés de chiffrement de contenu chiffrées avec des clés de chiffrement de clé déduites d'un mot de passe sont situées dans le champ EnvelopedData RecipientInfos PasswordRecipientInfo encryptedKey. Les clés d'authentification de message chiffrées avec des clés de chiffrement de clé déduites d'un mot de passe sont situées dans le champ AuthenticatedData RecipientInfos PasswordRecipientInfo encryptedKey.

4.4.1 PBKDF2

L'algorithme de déduction de clé PBKDF2 est spécifié dans la [RFC2898]. KeyDerivationAlgorithmIdentifer identifie l'algorithme de déduction de clé, et tout paramètre associé utilisé pour déduire la clé de chiffrement de clé du mot de passe fourni par l'utilisateur. L'identifiant d'algorithme pour l'algorithme PBKDF2 de déduction de clé est :

IDENTIFIANT D'OBJET id-PBKDF2 ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-5(5) 12 }

Le champ du paramètre AlgorithmIdentifier DOIT être PBKDF2-params :

```
PBKDF2-params ::= SEQUENCE {
    sel CHOIX { CHAINE D'OCTETS spécifiée , autre source de AlgorithmIdentifier },
    iterationCount ENTIER (1..MAX),
    keyLength ENTIER (1..MAX) OPTIONNEL,
    prf AlgorithmIdentifier
        DEFAUT { algorithme hMAC-SHA1, paramètres NUL } }
```

Au sein de PBKDF2-params, le sel DOIT utiliser la CHAINE D'OCTETS spécifiée.

5. Algorithmes de chiffrement du contenu

Cette section spécifie les conventions employées par les mises en œuvre de CMS qui prennent en charge le chiffrement de contenu en utilisant le Triple-DES à trois clé en mode CBC, le Triple-DES à deux clés en mode CBC, ou RC2 en mode CBC.

Les identifiants d'algorithme de chiffrement de contenu sont situés dans les champs EnvelopedData EncryptedContentInfo contentEncryptionAlgorithm et EncryptedData EncryptedContentInfo contentEncryptionAlgorithm.

Les algorithmes de chiffrement de contenu sont utilisés pour chiffrer les contenus situés dans le champ EnvelopedData EncryptedContentInfo encryptedContent et le champ EncryptedData EncryptedContentInfo encryptedContent.

5.1 Triple-DES CBC

L'algorithme Triple-DES est décrit dans ANSI X9.52 [3DES]. Le Triple-DES se compose de trois opérations de DES [DES] à la suite : chiffrement, déchiffrement, et chiffrement. Le Triple-DES à trois clés utilise une clé différente pour chaque opération DES. Le Triple-DES à deux clés utilise une clé pour les deux opérations de chiffrement et une clé différente pour l'opération de déchiffrement. Les mêmes identifiants d'algorithme sont utilisés pour le Triple-DES à deux et à trois clés. L'identifiant d'algorithme pour le Triple-DES en mode de chaînage de bloc de chiffrement (CBC, *Cipher*

Block Chaining) est :

IDENTIFIANT D'OBJET des-ede3-cbc ::= { iso(1) member-body(2) us(840) rsadsi(113549) encryptionAlgorithm(3) 7 }

Le champ de paramètres AlgorithmIdentifier DOIT être présent, et le champ paramètres doit contenir un CBCParameter :

CBCParameter ::= IV

IV ::= CHAINE D'OCTETS -- exactement 8 octets

5.2 RC2 CBC

L'algorithme RC2 est décrit dans la [RFC2268]. L'identifiant d'algorithme pour RC2 en mode CBC est :

IDENTIFIANT D'OBJET rc2-cbc::= { iso(1) member-body(2) us(840) rsadsi(113549) encryptionAlgorithm(3) 2 }

Le champ de paramètres AlgorithmIdentifier DOIT être présent, et le champ paramètres DOIT contenir un RC2CBCParameter :

```
RC2CBCParameter ::= SEQUENCE {
    rc2ParameterVersion  ENTIER,
    iv  CHAINE D'OCTETS  }                    -- exactement 8 octets
```

Les bits de clé efficaces RC2 (la taille de clé) supérieure à 32 et inférieure à 256 sont codés dans la rc2ParameterVersion. Pour les bits de clé efficaces de 40, 64, et 128, les valeurs de rc2ParameterVersion sont respectivement de 160, 120, et 58. Ces valeurs ne sont pas simplement la longueur de clé RC2. Noter que la valeur 160 doit être codée par deux octets (00 A0) car le codage sur un octet (A0) représente un nombre négatif.

6. Algorithmes de code d'authentification de message

Cette section spécifie les conventions employées par les mises en œuvre de CMS qui prennent en charge le HMAC avec code d'authentification de message (MAC) SHA-1.

Les identifiants d'algorithme de MAC sont situés dans le champ AuthenticatedData macAlgorithm.

Les valeurs de MAC sont situées dans le champ mac AuthenticatedData.

6.1 HMAC avec SHA-1

L'algorithme HMAC avec SHA-1 est décrit dans la [RFC2104]. L'identifiant d'algorithme pour HMAC avec SHA-1 est :

IDENTIFIANT D'OBJET hMAC-SHA1 ::= { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) 8 1 2 }

Il y a deux codages possibles pour le champ de paramètres AlgorithmIdentifier de HMAC avec SHA-1. Les deux solutions proviennent du fait que lorsque la syntaxe de 1988 pour le type AlgorithmIdentifier a été traduite dans la syntaxe 1997, le OPTIONNEL associé aux paramètres AlgorithmIdentifier a été perdu. Le OPTIONNEL a été retrouvé grâce à 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 peuvent coder les paramètres avec un NUL alors que d'autres les omettent entièrement.

Le champ de paramètres AlgorithmIdentifier est OPTIONNEL. S'il est présent, le champ paramètres DOIT contenir un NUL. Les mises en œuvre DOIVENT accepter le AlgorithmIdentifiers HMAC avec SHA-1 où les paramètres sont absents. Les mises en œuvre DOIVENT accepter les AlgorithmIdentifiers HMAC avec SHA-1 dont les paramètres sont à NUL. Les mises en œuvre DEVRAIENT générer des AlgorithmIdentifiers HMAC avec SHA-1 avec des paramètres absents.

7. Module ASN.1

```
CryptographicMessageSyntaxAlgorithms
      { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) modules(0) cmsalg-2001(16) }

DEFINITIONS DES ÉTIQUETTES IMPLICITES ::=
DÉBUT

-- EXPORTS Tout
-- Les types et valeurs définis dans ce module sont exportés pour utilisation dans les autres modules ASN.1.
-- D'autres applications peuvent les utiliser pour leurs propres besoins.

IMPORTS
-- Imports de la [RFC3280], Appendice A.1
        AlgorithmIdentifier
          DE PKIX1Explicit88 { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7)
             mod(0) pkix1-explicit(18) } ;

-- Identifiaants d'algorithme

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

IDENTIFIANT D'OBJET  md5 ::= { iso(1) member-body(2) us(840) rsadsi(113549) digestAlgorithm(2) 5 }

IDENTIFIANT D'OBJET  id-dsa ::=  { iso(1) member-body(2) us(840) x9-57(10040) x9cm(4) 1 }

IDENTIFIANT D'OBJET  id-dsa-with-sha1 ::=  { iso(1) member-body(2) us(840) x9-57(10040) x9cm(4) 3 }

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

IDENTIFIANT D'OBJET  md5WithRSAEncryption ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-
                                                1(1) 4 }

IDENTIFIANT D'OBJET  sha1WithRSAEncryption ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-
                                                1(1) 5 }

IDENTIFIANT D'OBJET  dh-public-number ::= { iso(1) member-body(2) us(840) ansi-x942(10046) number-type(2) 1 }

IDENTIFIANT D'OBJET  id-alg-ESDH ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16)
                                       alg(3) 5 }

IDENTIFIANT D'OBJET  id-alg-SSDH ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16)
                                       alg(3) 10 }

IDENTIFIANT D'OBJET  id-alg-CMS3DESwrap ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
                                              smime(16) alg(3) 6 }

IDENTIFIANT D'OBJET  id-alg-CMSRC2wrap ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
                                             smime(16) alg(3) 7 }

IDENTIFIANT D'OBJET  des-ede3-cbc ::= { iso(1) member-body(2) us(840) rsadsi(113549) encryptionAlgorithm(3) 7 }

IDENTIFIANT D'OBJET  rc2-cbc ::= { iso(1) member-body(2) us(840) rsadsi(113549) encryptionAlgorithm(3) 2 }

IDENTIFIANT D'OBJET  hMAC-SHA1 ::= { iso(1) identified-organization(3) dod(6) internet(1) security(5)
                                     mechanisms(5) 8 1 2 }

IDENTIFIANT D'OBJET  id-PBKDF2 ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-5(5) 12 }
```

```
-- Types de clé publique

    Dss-Pub-Key ::= ENTIER                                          -- Y

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

    DHPublicKey ::= ENTIER                                          -- y = g^x mod p

-- Types de valeur de signature

    Dss-Sig-Value ::= SEQUENCE {
        r                                ENTIER,
        s                               ENTIER }

-- Types de paramètre d'identifiant d'algorithme

    Dss-Parms ::= SEQUENCE {
        p                                ENTIER,
        q                               ENTIER,
        g                               ENTIER }

    DHDomainParameters ::= SEQUENCE {
        p                                   ENTIER,                 -- nombre premier impair, p =
        jq +1
        g                               ENTIER,         -- générateur, g
        q                               ENTIER,         -- facteur de p-1
        j                               ENTIER OPTIONNEL,    -- facteur de sous-groupe
      validationParms                         ValidationParms OPTIONNEL }

    ValidationParms ::= SEQUENCE {
        germe                                CHAINE BINAIRE,
        pgenCounter                     ENTIER }

    KeyWrapAlgorithm ::= AlgorithmIdentifier

    RC2wrapParameter ::= RC2ParameterVersion

    RC2ParameterVersion ::= ENTIER

    CBCParameter ::= IV

    IV ::= CHAINE D'OCTETS          -- exactement 8 octets

    RC2CBCParameter ::= SEQUENCE {
        rc2ParameterVersion ENTIER,
        iv CHAINE D'OCTETS  }  -- exactly 8 octets

    PBKDF2-params ::= SEQUENCE {
        sel                                  CHOIX { CHAINE D'OCTETS spécifiée, Identifiant
        d'algorithme d'autre source },
        iterationCount                  ENTIER (1..MAX),
        keyLength                       ENTIER (1..MAX) OPTIONNEL,
        prf                             AlgorithmIdentifier
    DEFAUT { algorithme hMAC-SHA1, paramètres NUL } }

FIN                                     -- de CryptographicMessageSyntaxAlgorithms
```

8. Références

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

[DES] American National Standards Institute. ANSI X3.106, "American National Standard for Information Systems - Data Link Encryption". 1983.

[DSS] National Institute of Standards et Technology. FIPS Pub 186,"Digital Signature Standard". 19 mai 1994.

[MODES] National Institute of Standards et Technology. FIPS Pub 81 "DES Modes of Operation". 2 décembre 1980.

[RFC1321] R. Rivest, "Algorithme de résumé de message MD5", avril 1992. *(Information)*

[RFC1750] D. Eastlake, 3rd et autres, "Recommandations d'aléa pour la sécurité", décembre 1994. *(Info., remplacée par la RFC 4086)*

[RFC2104] H. Krawczyk, M. Bellare et R. Canetti, "HMAC : Hachage de clés pour l'authentification de message", février 1997.

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

[RFC2268] R. Rivest, "Description de l'algorithme de chiffrement RC2(r)", mars 1998. *(Information)*

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

[RFC2437] B. Kaliski et J. Staddon, "PKCS n° 1 : Spécifications de la cryptographie RSA version 2.0", octobre 1998. *(Obsolète, voir RFC3447) (Information)*

[RFC2630] R. Housley, "Syntaxe de message cryptographique", juin 1999. *(Obsolète, voirRFC5652565256525652 , STD70)*

[RFC2631] E. Rescorla, "Méthode d'accord de clé Diffie-Hellman", juin 1999. *(P.S.)*

[RFC2898] B. Kaliski, "PKCS n° 5 : Spécification de la cryptographie fondée sur un mot de passe, version 2.0", septembre 2000. *(Info.)*

[RFC3217] R. Housley, "Enveloppe de clé en Triple-DES et RC2", décembre 2001. *(Information)*

[RFC3218] E. Rescorla, "Empêcher l'attaque du million de messages sur la syntaxe de message cryptographique", janvier 2002. *(Information)*

[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.

[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 la RFC5280)*

[RFC3369] R. Housley, "Syntaxe de message cryptographique (CMS)", août 2002. (*Obsolète, voir* RFC5652, *STD70.)*

[SHA1] National Institute of Standards et Technology. FIPS Pub 180-1 "Secure Hash Standard". 17 avril 1995.

[X.208-88] Recommandation CCITT X.208 "Spécification de la notation de syntaxe abstraite n° 1 (ASN.1)". 1988.

[X.209-88] Recommandation CCITT. X.209: "Spécification des règles de codage de base pour la notation de syntaxe abstraite n° 1 (ASN.1)". 1988.

9. Considérations pour la sécurité

La CMS donne une méthode pour la signature numérique des données, pour les résumés de données, le chiffrement des données, et l'authentification des données. Le présent document identifie les conventions pour l'utilisation de plusieurs algorithmes de chiffrement à utiliser avec la CMS.

Les mises en œuvre doivent protéger la clé privée du signataire. La compromission de la clé privée du signataire permet l'usurpation d'identité.

Les mises en œuvre doivent protéger la clé privée de gestion de clé, la clé de chiffrement de clé, et la clé de chiffrement de contenu. La compromission de la clé privée de gestion de clé ou de la clé de chiffrement de clé peut résulter en la divulgation de tous les contenus protégés avec cette clé. De même, la compromission de la clé de chiffrement de contenu peut résulter en la divulgation du contenu chiffré associé.

Les mises en œuvre doivent protéger la clé privée de gestion de clé et la clé d'authentification de message. La compromission de la clé privée de gestion de clé permet l'usurpation des données authentifiées. De même, la compromission de la clé d'authentification de message peut résulter en une indétectable modification du contenu authentifié.

La technique de gestion de clé employée pour distribuer les clés d'authentification de message doit elle-même fournir l'authentification, autrement, le contenu est livré avec l'intégrité en provenance d'une source inconnue. Ni RSA [RFC2313], [RFC2437], ni le Diffie-Hellman éphémère-statique [RFC2631] ne fournissent l'authentification d'origine des données nécessaire. Le Diffie-Hellman statique-statique [RFC2631] fournit bien l'authentificationd'origine des données nécessaire lorsque les clés publiques de l'origine et du receveur sont toutes deux liées aux identités appropriées dans les certificats X.509 [RFC3280].

Lorsque plus de deux parties partagent la même clé d'authentification de message, l'authentification de l'origine des données n'est pas fournie. Toute partie qui connait la clé d'authentification de message peut calculer un MAC valide, donc le contenu pourrait avoir été généré par n'importe laquelle des parties.

Les mises en œuvre doivent générer au hasard les clés de chiffrement de contenu, les clés d'authentification de message, les vecteurs d'initialisation (IV), les valeurs à utilisation unique (comme la valeur k lors de la génération d'une signature DSA) et le bourrage. Par ailleurs, la génération de paires de clés publique/privée s'apuie sur des nombres aléatoires. L'utilisation de générateurs de nombres pseudo aléatoires (PRNG, *pseudo-random number generator*) inadéquats pour générer de telles valeurs cryptographiques peut résulter en une sécurité faible ou inexistante. Un agresseur peut trouver beaucoup plus facile de reproduire m'environnement de PRNG qui a produit les clés, pour chercher dans le petit nombre de possibilités résultant, plutôt qu'une recherche en force brute sur la totalité de l'espace de clés. La génération de nombres aléatoires de qualité est difficile. La [RFC1750] offre d'importantes lignes directrices dans ce domaine, et l'appendice 3 de FIPS Pub 186 [DSS] donne une technique de PRNG de qualité.

Lorsque on utilise des algorithmes d'accord de clés ou de clé de chiffrement de clé symétrique à distribution préalable, une clé de chiffrement de clé est utilisée pour chiffrer la clé de chiffrement de contenu. Si les algorithmes de chiffrement de clé et de chiffrement de contenu sont différents, la sécurité effective est déterminée par le plus faible des deux algorithmes. Si, par exemple, le contenu est chiffré avec Triple-DES à 168 bits et si la clé de chiffrement de contenu Triple-DES est enveloppée avec une clé RC2 de 40 bits, une protection d'au plus 40 bits est alors fournie. Une recherche triviale pour déterminer la valeur de la clé RC2 de 40 bits peut découvrir la clé Triple-DES, et alors la clé Triple-DES peut être utilisée pour déchiffrer le contenu. Donc, les mises en œuvre doivent s'assurer que les algorithmes de chiffrement de clé sont aussi forts ou plus forts que les algorithmes de chiffrement de contenu.

La [RFC3217] spécifie les algorithmes d'enveloppe de clé utilisés pour chiffrer une clé de chiffrement de contenu Triple-DES avec une clé de chiffrement de clé Triple-DES [3DES] ou pour chiffrer une clé de chiffrement de contenu RC2 avec une clé de chiffrement de clé RC2 [RFC2268]. Les algorithmes d'enveloppe de clé font usage du mode CBC [MODES]. Ces algorithmes d'enveloppe de clé ont été révisés pour être utilisés avec Triple-DES et RC2. Ils n'ont pas été révisés pour être utilisés avec d'autres modes cryptographiques ou d'autres algorithmes de chiffrement. Donc, si une mise en œuvre de CMS souhaite prendre en charge des méthodes de chiffrement en plus de Triple-DES ou RC2, des algorithmes d'enveloppe de clé supplémentaires devront être définis pour prendre en charge ces chiffrements supplémentaires.

Les développeurs devraient être conscients que les algorithmes cryptographiques deviennent plus faibles avec le temps. Alors que de nouvelles techniques de cryptanalyse sont développées et que s'améliorent les performance des ordinateurs, le facteur de travail pour casser un algorithme cryptographique particulier va diminuer. Les mises en œuvre d'algorithmes cryptographiques devraient donc être modulaires pour permettre que de nouveaux algorithmes soient insérés directement. C'est-à-dire que les mises en œuvre devraient être prêtes à mettre à jour régulièrement leur ensemble d'algorithmes.

Les utilisateurs de la CMS, en particulier ceux qui employent la CMS pour soutenir des applications interactives, devraient être conscients que RSA (PKCS n° 1 v1.5) tel que spécifié dans la [RFC2313], est vulnérable à des attaques adaptives de texte chiffré choisi lorsque il est appliqué à des fins de chiffrement. L'exploitation de cette faiblesse identifiée, qui révèle le résultat d'un déchiffrement RSA particulier, requiert l'accès à un oracle qui va répondre à un grand nombre de textes chiffrés (sur la base des résultats actuellement disponibles, des centaines de milliers ou plus) qui sont construits de façon adaptive en réponse à des réponses reçues préalablement, ce qui donne des informations sur la réussite ou l'échec des tentatives de déchiffrement. Il en résulte que l'attaque paraît significativement moins faisable pour les environnements S/MIME à remise différée que pour les protocoles directement interactifs. Lorsque les constructions de CMS sont appliquées comme une couche de chiffrement intermédiaire au sein d'un environnement de questions réponses interactives, l'exploitation pourrait en être plus réalisable.

Une version mise à jour de PKCS n° 1 a été publiée, PKCS n° 1 Version 2.0 [RFC2437]. Ce document mis à jour remplace la RFC2313. PKCS n° 1 Version 2.0 préserve la prise en charge du format de bourrage de chiffrement défini dans PKCS n° 1 Version 1.5 [RFC2313], et il définit aussi une nouvelle solution de remplacement. Pour résoudre la vulnérabilité au texte chiffré choisi adaptatif, PKCS n° 1 Version 2.0 spécifie et recommande l'utilisation du bourrage de chiffrement asymétrique optimal (OAEP, *Optimal Asymmetric Encryption Padding*) lorsque le chiffrement RSA est utilisé pour fournir la confidentialité. Les concepteurs de protocoles et systèmes qui emploient la CMS dans des environnements interactifs devraient envisager l'usage de OAEP, ou devraient s'assurer que les informations qui pourraient révéler le succès ou l'échec de tentatives d'opérations de déchiffrement de PKCS n° 1 Version 1.5 ne sont pas fournies. La prise en charge de OAEP sera vraisemblablement ajoutée à une future version de la spécification de l'algorithme de CMS. Voir dans la [RFC3218] d'autres informations sur la façon de déjouer la vulnérabilité au texte chiffré choisi adaptatif dans les mises en œuvre de PKCS n° 1 Version 1.5.

10. Remerciements

Le présent document est le résultat des contributions de nombreuxprofessionnels. Je remercie de leur dur labeur tous les membres du groupe de travail S/MIME de l'IETF. Des remerciements particuliers à Rich Ankney, Simon Blake-Wilson, Tim Dean, Steve Dusse, Carl Ellison, Peter Gutmann, Bob Jueneman, Stephen Henson, Paul Hoffman, Scott Hollenbeck, Don Johnson, Burt Kaliski, John Linn, John Pawling, Blake Ramsdell, Francois Rousseau, Jim Schaad, et Dave Solo pour leurs efforts et leur soutien.

11 Adresse de l'auteur

Russell Housley
RSA Laboratories
918 Spring Knoll Drive
Herndon, VA 20170
mél : rhousley@rsasecurity.com

12. Déclaration complète de droits de reproduction

ENGINEERING TASK FORCE déclinent toutes garanties, exprimées ou implicites, y compris mais non limitées à toute garantie que l'utilisation des informations ci encloses ne violent aucun droit ou aucune garantie implicite de commercialisation ou d'aptitude à un objet particulier.

Remerciement

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