Groupe de travail Réseau
Request for Comments : 2898
Catégorie : Information

B. Kaliski, RSA Laboratories
septembre 2000
Traduction Claude Brière de L'Isle

Spécification de cryptographie fondée sur le mot de passe PKCS n° 5 version 2.0

Statut de ce mémoire
Le présent mémoire apporte des informations pour la communauté de l'Internet. Il ne spécifie aucune sorte de norme de l'Internet. La distribution du présent mémoire n'est soumise à aucune restriction.

Résumé
Le présent mémoire représente une réédition de PKCS n° 5 v2.0 de la série des normes de cryptographie à clé publique (PKCS, *Cryptographie à clés publiques Standard*) des RSA Laboratories, et le contrôle de son amendement est conservé au sein du processus PKCS. Le corps de ce document, sauf la section des considérations pour la sécurité est tiré directement de cette spécification.
Le présent document formule des recommandations pour la mise en œuvre de la cryptographie fondée sur le mot de passe, et couvre les fonctions de déduction de clé, les schémas de chiffrement, les schémas d'authentification de message, et la syntaxe ASN.1 qui identifie les techniques.
Les recommandations sont destinées à une application générale au sein des systèmes informatiques et de communications, et à ce titre ont une grande souplesse. Elles sont particulièrement destinées à la protection d'informations sensibles telles que les clés privées, comme dans PKCS n° 8 [25]. Il est prévu que les applications standard et les profils de mise en œuvre fondés sur ces spécifications puissent inclure des contraintes supplémentaires.
Les autres techniques cryptographiques fondées sur les mots de passe, telles que les protocoles d'authentification d'entité de clé fondée sur le mot de passe et d'établissement de clés [4] [5] [26] sortent du domaine d'application du présent document. Les lignes directrices pour le choix des mots de passe sont aussi en dehors de son domaine d'application.

Table des matières

1. Introduction

Le présent document apporte des recommandations pour la mise en œuvre de la cryptographie fondée sur le mot de passe, couvrant les aspects suivants :
- fonctions de déduction de clé
- chiffrement
- schémas d'authentification de message
- syntaxe ASN.1 d'identification des techniques

Les recommandations sont destinées aux applications générales dans les systèmes informatiques et de communications, et comportent à ce titre une grande souplesse. Elles sont principalement destinées à la protection des informations sensibles comme les clés privées, comme dans PKCS n° 8 [25]. Il est prévu que les normes d'application et les profils de mise en œuvre fondés sur ces spécifications peuvent inclure des contraintes supplémentaires.

Les autres techniques cryptographiques fondées sur les mots de passe, telles que les protocoles d'authentification d'entité de clé fondée sur le mot de passe et d'établissement de clés [4] [5] [26] sortent du domaine d'application du présent document. Les lignes directrices pour le choix des mots de passe sont aussi en dehors de son domaine d'application.

Le présent document subroge PKCS n° 5 version 1.5 [24], mais comporte des techniques compatibles.

2. Notation

C	texte chiffré, une chaîne d'octets
c	compte d'itération, un entier positif
DK	clé déduite, une chaîne d'octets
dkLen	longueur en octets de la clé déduite, un entier positif
EM	message codé, une chaîne d'octets
Hash	fonction de hachage sous-jacente
hLen	longueur en octets du résultat de la fonction pseudo aléatoire, un entier positif
l	longueur en blocs de la clé déduite, un entier positif
IV	vecteur d'initialisation, une chaîne d'octets
K	clé de chiffrement, une chaîne d'octets
KDF	fonction de déduction de clé
M	message, une chaîne d'octets
P	mot de passe, une chaîne d'octets
PRF	fonction pseudo aléatoire sous-jacente
PS	chaîne de bourrage, une chaîne d'octets
psLen	longueur en octets de la chaîne de bourrage, un entier positif
S	sel, une chaîne d'octets
T	code d'authentification de message, une chaîne d'octets
T_1, ..., T_l, U_1, ..., U_c	valeurs intermédiaires, une chaîne d'octets

01, 02, ..., 08 octets avec les valeurs 1, 2, ..., 8

\oux ou exclusif au bit près de chaînes de deux octets

|| || opérateur de longueur d'octet

|| opérateur d'enchaînement

<i..j> opérateur d'extraction de sous-chaîne : extrait les octets de i à j, $0 \le i \le j$

3. Généralités

Dans de nombreuses applications de cryptographie à clés publiques, la sécurité de l'utilisateur dépend en fin de compte d'une ou deux valeurs de texte secret ou mots de passe. Comme un mot de passe n'est pas directement applicable comme clé à un système de chiffrement conventionnel, un certain traitement du mot de passe est cependant nécessaire pour effectuer avec lui les opérations cryptographiques. De plus, comme les mots de passe sont souvent choisis à partir d'un espace relativement restreint, il faut veiller particulièrement à ce que le traitement défende contre les attaques de recherche.

Une approche générale de la cryptographie fondée sur le mot de passe, comme décrite par Morris et Thompson [8] pour la protection des tableaux de mots de passe, est de combiner un mot de passe avec un sel pour produire une clé. Le sel peut être vu comme un indice dans un grand ensemble de clés déduites du mot de passe, et il n'a pas besoin d'être gardé secret. Bien qu'il soit possible à un agresseur de construire un tableau des mots de passe possibles (c'est ce qu'on appelle une "attaque de dictionnaire") construire un tableau des clés possibles sera difficile, car il y aura de nombreuses clés possibles pour chaque mot de passe. Un agresseur sera donc limité par les recherches sur tous les mots de passe différents pour chaque sel.

Une autre approche de la cryptographie fondée sur le mot de passe est de construire des techniques de déduction de clé qui sont relativement coûteuses, ce qui accroît le coût d'une recherche exhaustive. Une façon de le faire est d'inclure un compte d'itérations dans la technique de déduction de clé, qui indique combien de fois itérer certaines fonctions sous-jacentes par lesquelles les clés sont déduites. Un nombre modeste d'itérations, disons 1000, ne va probablement pas être un fardeau trop lourd pour les parties légitimes lors du calcul d'une clé, mais sera d'un poids significatif pour les agresseurs.

Sel et compte d'itération formaient la base du chiffrement fondé sur le mot de passe dans PKCS n° 5 v1.5, et adoptée ici aussi pour les diverses opérations cryptographiques. Donc, la déduction de clé fondée sur le mot de passe telle que définie ici est une fonction d'un mot de passe, d'un sel, et d'un compte d'itérations, où ces deux dernières quantités n'ont pas besoin d'être gardées secrètes.

À partir d'une fonction de déduction de clé fondée sur le mot de passe, on définit directement les schémas de chiffrement fondé sur le mot de passe et d'authentification de message. Comme dans PKCS n° 5 v1.5, les schémas de chiffrement fondé sur le mot de passe sont ici fondés sur un schéma sous-jacent de chiffrement conventionnel, où la clé pour le schéma conventionnel est déduite du mot de passe. De même, le schéma d'authentification de message fondé sur le mot de passe se fonde sur un schéma sous-jacent conventionnel. Cette approche en deux couches rend les techniques fondées sur le mot de passe modulaires en termes de techniques sous-jacentes sur lesquelles s'appuyer.

Il est prévu que les fonctions de déduction de clé fondées sur le mot de passe puissent trouver d'autres applications que les simples schémas de chiffrement et d'authentification de message définis ici. Par exemple, on peut déduire un ensemble de clés avec une seule application d'une fonction de déduction de clé, plutôt que de déduire chaque clé avec une application séparée de la fonction. Les clés dans l'ensemble seraient obtenues comme sous-chaînes du résultat de la fonction de déduction de clé. Cette approche peut être employée au titre de l'établissement de clés dans un protocole en mode session. Une autre application est la vérification de mot de passe, où le résultat de la fonction de déduction de clé est mémorisé (avec le sel et le compte d'itération) pour les besoins de vérifications de mot de passe ultérieures.

Tout au long du présent document, un mot de passe est considéré comme une chaîne d'octets de longueur arbitraire dont l'interprétation comme chaîne de texte n'est pas spécifiée. Dans l'intérêt de l'interopérabilité, il est cependant recommandé que les applications suivent des règles communes de codage de texte. ASCII et UTF-8 [27] sont deux possibilités. (ASCII est un sous-ensemble de UTF-8.)

Bien que le choix d'un mot de passe sorte du domaine d'application du présent document, des lignes directrices ont été publiées dans [17] dont on se trouvera bien de tenir compte.

4. Sel et compte d'itération

Comme sel et compte d'itération sont au cœur des techniques définies dans le présent document, ils méritent quelques précisions.

4.1 Sel

Dans une cryptographie fondée sur le mot de passe, le sel a traditionnellement servi au besoin de produire un grand ensemble de clés correspondant à un certain mot de passe, parmi lequel il en est choisi un au hasard conformément au sel. Une clé individuelle est choisie dans l'ensemble en appliquant une fonction de déduction de clé KDF, telle que

DK = KDF (P, S)

où DK est la clé déduite, P est le mot de passe, et S est le sel. Cela présente deux avantages :

1. Il est difficile à un agresseur de pré calculer toutes les clés qui correspondent à un dictionnaire de mots de passe, ou même les clés les plus probables. Si le sel est long de 64 bits, par exemple, il y aura 2^64 clés pour chaque mot de passe. Un agresseur en est donc réduit à chercher les mots de passe après qu'une opération fondée sur le mot de passe a été effectuée et que le sel est connu.

2. Il n'est pas vraisemblable que la même clé soit choisie deux fois. Là encore, si le sel est long de 64 bits, les chances de "collision" entre clés ne deviennent pas significatives tant qu'environ 2^32 clés ont été produites, conformément au paradoxe de l'anniversaire. Cela répond à certaines des inquiétudes sur les interactions entre plusieurs utilisations de la même clé, qui peut s'appliquer pour certaines techniques de chiffrement et d'authentification.

Dans un chiffrement fondé sur le mot de passe, la partie qui chiffre un message peut avoir l'assurance que ces avantages se réalisent en choisissant simplement un sel grand et suffisamment aléatoire lorsque elle déduit une clé de chiffrement d'un mot de passe. Une personne qui génère un code d'authentification de message peut obtenir une telle assurance de façon similaire.

La partie qui déchiffre un message ou vérifie un code d'authentification de message, ne peut cependant pas être sûre qu'un sel fourni par une autre partie a en fait été généré de façon aléatoire. Il est possible, par exemple, que le sel ait été copié d'une autre opération fondée sur le mot de passe, afin de tenter d'exploiter les interactions entre plusieurs utilisations de la même clé. Par exemple, supposons que deux parties légitimes échangent un message chiffré, où la clé de chiffrement est une clé de 80 bits déduite d'un mot de passe partagé avec un sel. Un agresseur pourrait tirer le sel de ce chiffrement et le fournir à une des parties bien qu'il ait été pour une clé de 40 bits. Si la partie révèle le résultat du déchiffrement avec la clé de 40 bits, l'agresseur peut être capable de résoudre la clé de 40 bits. Dans le cas où cette clé de 40 bits est la première moitié de la clé de 80 bits, l'agresseur peut alors facilement résoudre les 40 bits restants de la clé de 80 bits.

Pour se défendre contre de telles attaques, l'interaction entre plusieurs utilisations de la même clé devrait être analysée soigneusement, ou le sel devrait contenir des données qui font explicitement la distinction entre les différentes opérations. Par exemple, le sel pourrait avoir un octet supplémentaire, non aléatoire, qui spécifie si la clé déduite est pour chiffrement, pour authentification de message, ou pour quelque autre opération.

Sur la base de ce qui vient d'être dit, on fait la recommandation suivante pour le choix du sel :

1. Si il n'y a pas de souci d'interactions entre plusieurs utilisations de la même clé (ou d'un préfixe de cette clé) avec les techniques de chiffrement et d'authentification fondées sur le mot de passe prises en charge pour un certain mot de passe, le sel peut alors être généré au hasard et il n'y a pas besoin de vérifier qu'il est d'un format particulier par la partie qui reçoit le sel. Il devrait être long d'au moins huit octets (64 bits).

2. Autrement, le sel devrait contenir des données qui font explicitement la distinction entre les différentes opérations et les différentes longueurs de clés, en plus d'une partie aléatoire qui est d'au moins huit octets de long, et ces données devraient être vérifiées ou régénérées par la partie qui reçoit le sel. Par exemple, le sel pourrait avoir un octet supplémentaire non aléatoire qui spécifie l'objet de la clé déduite. Autrement, ce pourrait être le codage d'une structure qui spécifie des informations détaillées sur la clé dérivée, comme la technique de chiffrement ou d'authentification et un numéro de séquence parmi les différentes clés déduites du mot de passe. Le format particulier des données supplémentaires est laissé au choix de l'application.

Note : Si un générateur de nombres aléatoire ou de nombres pseudo aléatoires n'est pas disponible, une solution de remplacement déterministe pour générer le sel (ou sa partie aléatoire) est d'appliquer une fonction de déduction de

clé fondée sur le mot de passe au mot de passe et au message M à traiter. Par exemple, le sel pourrait être calculé avec une fonction de déduction de clé comme S = KDF (P, M). Cette approche n'est pas recommandée si le message M est connu pour appartenir à un petit espace de messages (par exemple, "Oui" ou "Non") cependant, même là, il n'y aura qu'un petit nombre de sels possibles.

4.2 Compte d'itération

Un compte d'itérations sert traditionnellement pour augmenter le coût de production des clés à partir d'un mot de passe, augmentant par là la difficulté des attaques. Pour les méthodes exposées dans ce document, un minimum de 1000 itérations est recommandé. Cela va augmenter significativement le coût d'une recherche exhaustive sur les mots de passe, sans impact notable sur le coût de déduction des clés individuelles.

5. Fonctions de déduction de clé

Une fonction de déduction de clé produit une clé dérivée à partir d'une clé de base et d'autres paramètres. Dans une fonction de déduction de clé fondée sur le mot de passe, la clé de base est un mot de passe et les autres paramètres sont une valeur de sel et un compte d'itérations, comme mentionné à la Section 3.

La principale application des fonctions de déduction de clé fondées sur le mot de passe définies ici est dans les schémas de chiffrement décrits à la Section 6 et le schéma d'authentification de message de la Section 7. D'autres applications sont certainement possibles, d'où la définition indépendante de ces fonctions.

Deux fonctions sont spécifiées dans cette section : PBKDF1 et PBKDF2.
PBKDF2 est recommandée pour les nouvelles applications; PBKDF1 n'est incluse que pour la compatibilité avec les applications existantes, et n'est pas recommandée pour les nouvelles applications.

Une application typique des fonctions de déduction de clé définies ici pourrait inclure les étapes suivantes :
1. Choisir un sel S et un compte d'itérations c, comme mentionné à la Section 4.
2. Choisir une longueur en octets pour la clé déduite, dkLen.
3. Appliquer la fonction de déduction de clé au mot de passe, au sel, au compte d'itérations et à la longueur de clé pour produire une clé déduite.
4. Sortir la clé dérivée.

On peut déduire n'importe quel nombre de clés d'un mot de passe en faisant varier le sel, comme décrit à la Section 3.

5.1 PBKDF1

PBKDF1 applique une fonction de hachage, MD2 [6], MD5 [19] ou SHA-1 [18], pour déduire les clés. La longueur de la clé déduite est limitée par la longueur du résultat de la fonction de hachage, qui est de 16 octets pour MD2 et MD5 et de 20 octets pour SHA-1. PBKDF1 est compatible avec le processus de déduction de clé décrit dans PKCS n° 5 v1.5.

PBKDF1 n'est recommandé que pour la compatibilité avec les applications existantes car les clés qu'il produit peuvent n'être pas assez grandes pour certaines applications.

PBKDF1 (P, S, c, dkLen)

Options	Hachage	fonction de hachage sous-jacente
Entrée	P	mot de passe, une chaîne d'octets
	S	sel, chaîne de huit octets
	c	compte d'itérations, entier positif
	dkLen	longueur prévue de la clé déduite en octets, entier positif, au plus de 16 pour MD2 ou MD5 et de 20 pour SHA-1
Résultat	DK	clé déduite, une chaîne d'octets de dkLen

Étapes :

1. Si dkLen > 16 pour MD2 et MD5, ou si dkLen > 20 pour SHA-1, sortir "clé déduite trop longue" et arrêter.

2. Appliquer la fonction de hachage sous-jacente Hash pendant c itérations à l'enchaînement du mot de passe P et su sel S, puis extraire les dkLen premiers octets pour produire une clé déduite DK :
   T_1 = Hash (P || S) ,
   T_2 = Hash (T_1) ,
   ...
   T_c = Hash (T_{c-1}) ,
   DK = Tc<0..dkLen-1>

3. Sortir la clé déduite DK.

5.2 PBKDF2

PBKDF2 applique une fonction pseudo aléatoire (voir un exemple à l'Appendice B.1) pour déduire les clés. La longueur de la clé déduite n'est pas limité par nature. (Cependant, l'espace maximum de recherche efficace pour la clé déduite peut être limité par la structure de la fonction pseudo aléatoire sous-jacente. Voir les précisions données à l'Appendice B.1.) PBKDF2 est recommandé pour les nouvelles applications.

PBKDF2 (P, S, c, dkLen)

Options	PRF	fonction pseudo aléatoire sous-jacente (hLen note la longueur en octets du résultat de la fonction pseudo aléatoire)
Entrées :	P	mot de passe, une chaîne d'octets
	S	sel, une chaîne d'octets
	c	compte d'itérations, un entier positif
	dkLen	longueur prévue en octets de la clé déduite, un entier positif, d'au plus (2^32 - 1) * hLen
Résultat :	DK	clé déduite, chaîne d'octets de dkLen.

Étapes :

1. Si dkLen > (2^32 - 1) * hLen, afficher "clé déduite trop longue" et arrêter.

2. Soit l le nombre de blocs de hLen octets dans la clé déduite, en arrondissant à l'entier supérieur, et soit r le nombre d'octets dans le dernier bloc :
   l = CEIL (dkLen / hLen) ,
   r = dkLen - (l - 1) * hLen .
   Ici, CEIL (x) est la fonction "plafond", c'est-à-dire, le plus petit entier supérieur ou égal à x.

3. Pour chaque bloc de la clé déduite, on applique la fonction F définie ci-dessous au mot de passe P, au sel S, au compte d'itérations c, et à l'indice de bloc pour calculer le bloc :
   T_1 = F (P, S, c, 1) ,
   T_2 = F (P, S, c, 2) ,
   ...
   T_l = F (P, S, c, l) ,
   où la fonction F est définie comme la somme des ou exclusifs de la première itération de c de la fonction pseudo aléatoire sous-jacente PRF appliquée au mot de passe P et de l'enchaînement du sel et du bloc indice i :
   F (P, S, c, i) = U_1 \oux U_2 \oux ... \oux U_c
   où
   U_1 = PRF (P, S || INT (i)) ,
   U_2 = PRF (P, U_1) ,
   ...
   U_c = PRF (P, U_{c-1}) .

   Ici, INT (i) est un codage en quatre octets de l'entier i, octet de poids fort en premier.

4. Enchaîner les blocs et extraire les premiers dkLen octets pour produire une clé déduite DK :

   DK = T_1 || T_2 || ... || T_l<0..r-1>

5. Sortir la clé déduite DK.

Note. La construction de la fonction F suit une approche "ceinture et bretelles". L'itération U_i est calculée par récurrence pour retirer un degré de parallélisme à un agresseur ; ils sont traités par l'opération de OU exclusif pour réduire la crainte que la récurrence ne dégénère en un petit ensemble de valeurs.

6. Schémas de chiffrement

Un schéma de chiffrement, dans l'établissement symétrique, consiste en une opération de chiffrement et une opération de déchiffrement, où l'opération de chiffrement produit un texte chiffré avec une clé à partir d'un message, et l'opération de déchiffrement récupère le message à partir du texte chiffré à l'aide de la même clé. Dans un schéma de chiffrement fondé sur le mot de passe, la clé est un mot de passe.

Une application typique de schéma de chiffrement fondé sur un mot de passe est une méthode de protection à clé privée, où le message contient des informations de clé privée, comme dans PKCS n° 8. Les schémas de chiffrement définis ici seraient des algorithmes de chiffrement convenables dans ce contexte.

Deux schémas sont spécifiés dans cette section : PBES1 et PBES2. PBES2 est recommandé pour de nouvelles applications ; PBES1 n'est inclus que pour la compatibilité avec les applications existantes, et n'est pas recommandé pour les nouvelles applications.

6.1 PBES1

PBES1 combine la fonction de PBKDF1 (paragraphe 5.1) avec un chiffrement de bloc sous-jacent, qui devra être DES [15] ou RC2(tm) [21] en mode CBC [16]. PBES1 est compatible avec le schéma de chiffrement de PKCS n° 5 v1.5.

PBES1 n'est recommandé que pour la compatibilité avec les applications existantes, car il n'accepte que deux schémas de chiffrement sous-jacents qui ont une taille de clé (56 ou 64 bits) qui peut n'être pas assez grande pour certaines applications.

6.1.1 Opération de chiffrement

L'opération de chiffrement pour PBES1 comporte les étapes suivantes, qui chiffrent un message M avec un mot de passe P pour produire un texte chiffré C :

1. Choisir un sel S de huit octets et un compte d'itérations c, comme mentionné à la Section 4.

2. Applique la fonction PBKDF1 de déduction de clé (paragraphe 5.1) au mot de passe P, au sel S, et au compte d'itérations c pour produire une clé déduite DK d'une longueur de 16 octets:

   DK = PBKDF1 (P, S, c, 16) .

3. Séparer la clé déduite DK en une clé de chiffrement K consistant en les huit premiers octets of DK et un vecteur d'initialisation IV consistant en les huit octets suivants :
   K = DK<0..7> ,
   IV = DK<8..15> .

4. Enchaîner M et une chaîne de bourrage PS pour former un message codé EM :
   EM = M || PS ,
   où la chaîne de bourrage PS consiste en 8-(||M|| mod 8) octets, chacun de valeur 8-(||M|| mod 8). La chaîne de bourrage PS va satisfaire à l'une des déclarations suivantes :
   PS = 01, if ||M|| mod 8 = 7 ;
   PS = 02 02, if ||M|| mod 8 = 6 ;
   ...
   PS = 08 08 08 08 08 08 08 08, si ||M|| mod 8 = 0.
   La longueur en octets du message codé sera un multiple de huit et il sera possible de récupérer le message M sans ambiguïté à partir du message codé. (Cette règle de bourrage est tirée de la RFC1423 [3].)

5. Chiffrer le message codé EM avec le bloc de chiffrement sous-jacent (DES ou RC2) en mode de chaînage de bloc de chiffrement avec la clé de chiffrement K avec le vecteur d'initialisation IV pour produire le texte chiffré C. Pour DES, la clé K devra être considérée comme un codage sur 64 bits d'une clé DES de 56 bits en ignorant les bits de parité (voir

dans [9]). Pour RC2, les "bits de clé efficace" devront être 64 bits.

6. Sortir le texte chiffré C.

Le sel S et le compte d'itérations c peuvent être portés par la partie qui effectue le déchiffrement dans une valeur d'identifiant d'algorithme (voir à l'Appendice A.3).

6.1.2 Opération de déchiffrement

L'opération de déchiffrement pour PBES1 comporte les étapes suivantes, qui déchiffrent un texte chiffré C sous un mot de passe P pour récupérer un message M :

1. Obtenir le sel S de huit octets et le compte d'itérations c.

2. Appliquer la fonction de déduction de clé PBKDF1 (paragraphe 5.1) au mot de passe P, au sel S, et au compte d'itérations c pour produire une clé déduite DK d'une longueur de 16 octets :
   DK = PBKDF1 (P, S, c, 16)

3. Séparer la clé déduite DK en deux clés de chiffrement K consistant en les huit premiers octets de DK et un vecteur d'initialisation IV consistant en les huit octets suivants :
   K = DK <0..7> ,
   IV = DK <8..15> .

4. Déchiffrer le texte chiffré C avec le bloc de chiffrement sous-jacent (DES ou RC2) en mode de chaînage de bloc de chiffrement avec la clé de chiffrement K et le vecteur d'initialisation IV pour recouvrer un message codé EM. Si la longueur en octets du texte chiffré C n'est pas un multiple de huit, afficher "erreur de déchiffrement" et arrêter.

5. Séparer le message codé EM en un message M et une chaîne de bourrage PS :
   EM = M || PS ,
   où la chaîne de bourrage PS consiste en un certain nombre d'octets psLen ayant chacun la valeur psLen, où psLen est entre 1 et 8. Si il n'est pas possible de séparer le message codé EM de cette manière, afficher "erreur de déchiffrement" et arrêter.

6. Sortir le message récupéré M.

6.2 PBES2

PBES2 combine une fonction de déduction de clés fondée sur le mot de passe, qui devra être PBKDF2 (paragraphe 5.2) pour cette version de PKCS n° 5, avec un schéma de chiffrement sous-jacent (voir des exemples à l'Appendice B.2). La longueur de clé et tous les autres paramètres du schéma de chiffrement sous-jacents dépendent de ce schéma.

PBES2 est recommandé pour les nouvelles applications.

6.2.1 Opération de chiffrement

L'opération de chiffrement pour PBES2 comporte les étapes suivantes, qui chiffrent un message M sous un mot de passe P pour produire un texte chiffré C, en appliquant une fonction de déduction de clé KDF choisie et un schéma choisi de chiffrement sous-jacent :

1. Choisir un sel S et un compte d'itérations c, comme expliqué à la Section 4.

2. Choisir la longueur en octets, dkLen, pour la clé déduite pour le schéma choisi de chiffrement sous-jacent.

3. Appliquer la fonction de déduction de clé choisie au mot de passe P, au sel S, et au compte d'itérations c pour produire une clé déduite DK de longueur dkLen octets :
   DK = KDF (P, S, c, dkLen) .

4. Chiffrer le message M avec le schéma choisi de chiffrement sous-jacent avec la clé déduite DK pour produire un texte chiffré C. (Cette étape peut impliquer le choix de paramètres tels qu'un vecteur d'initialisation et un bourrage, selon le schéma sous-jacent.)

5. Sortir le texte chiffré C.

Le sel S, le compte d'itérations c, la longueur de clé dkLen, et les identifiants pour la fonction de déduction de clé et le schéma de chiffrement sous-jacent peuvent être convoyés à la partie qui effectue le déchiffrement dans une valeur d'identifiant d'algorithme (voir à l'Appendice A.4).

6.2.2 Opération de déchiffrement

L'opération de déchiffrement pour PBES2 comporte les étapes suivantes, qui déchiffrent le texte chiffré C sous un mot de passe P pour récupérer un message M :

1. Obtenir le sel S pour l'opération.

2. Obtenir le compte d'itérations c pour la fonction de déduction de clé.

3. Obtenir la longueur de clé en octets, dkLen, pour la clé déduite pour le schéma de chiffrement sous-jacent.

4. Appliquer la fonction de déduction de clé choisie au mot de passe P, au sel S, et au compte d'itérations c pour produire une clé déduite DK de longueur dkLen octets :
   DK = KDF (P, S, c, dkLen) .

5. Déchiffrer le texte chiffré C avec le schéma de chiffrement sous-jacent sous la clé déduite DK pour recouvrer un message M. Si la fonction de déchiffrement affiche "erreur de déchiffrement", afficher alors "erreur de déchiffrement" et arrêter.

6. Sortir le message M récupéré.

7. Schémas d'authentification de message

Un schéma d'authentification de message consiste en une opération de génération d'un MAC (code d'authentification de message) et en une opération de vérification de MAC, où l'opération de génération de MAC produit un code d'authentification de message à partir d'un message sous une clé, et l'opération de vérification de MAC vérifie le code d'authentification de message sous la même clé. Dans un schéma d'authentification de message fondé sur le mot de passe, la clé est un mot de passe.

Une schéma est spécifié dans cette section : PBMAC1.

7.1 PBMAC1

PBMAC1 combine une fonction de déduction de clé fondée sur un mot de passe, qui devra être PBKDF2 (paragraphe 5.2) pour cette version de PKCS n° 5, avec un schéma d'authentification de message sous-jacent (voir un exemple à l'Appendice B.3). La longueur de clé et tous autres paramètres pour le schéma d'authentification de message sous-jacent dépendent du schéma.

7.1.1 Génération du MAC

L'opération de génération du MAC pour PBMAC1 comporte les étapes suivantes, qui traitent un message M sous un mot de passe P pour générer un code d'authentification de message T, en appliquant une fonction de déduction de clé KDF choisie et un schéma d'authentification de message choisi :

1. Choisir un sel S et un compte d'itérations c, comme indiqué à la Section 4.

2. Choisir une longueur de clé en octets, dkLen, pour la clé déduite pour la fonction d'authentification de message sous-jacente.

3. Applique la fonction de déduction de clé choisie au mot de passe P, au sel S, et au compte d'itérations c pour produire une clé déduite DK de longueur dkLen octets :

   DK = KDF (P, S, c, dkLen) .

4. Traiter le message M avec le schéma d'authentification de message sous-jacent sous la clé déduite DK pour générer un code d'authentification de message T.

5. Sortir le code d'authentification de message T.

Le sel S, le compte d'itérations c, la longueur de clé dkLen, et les identifiants pour la fonction de déduction de clé et le schéma d'authentification de message sous-jacent peuvent être convoyés à la partie qui effectue la vérification dans une valeur d'identifiant d'algorithme (voir à l'Appendice A.5).

7.1.2 Vérification du MAC

L'opération de vérification de MAC pour PBMAC1 comporte les étapes suivantes, qui traitent un message M sous un mot de passe P pour vérifier un code d'authentification de message T :

1. Obtenir le sel S et le compte d'itérations c.

2. Obtenir la longueur de clé en octets, dkLen, pour la clé déduite pour le schéma d'authentification de message sous-jacent.

3. Appliquer la fonction de déduction de clé choisie au mot de passe P, au sel S, et au compte d'itérations c pour produire une clé déduite DK de longueur dkLen octets :
   DK = KDF (P, S, c, dkLen) .

4. Traiter le message M avec le schéma d'authentification de message sous-jacent sous la clé déduite DK pour vérifier le code d'authentification de message T.

5. Si le code d'authentification de message se vérifie, afficher "correct" ; autrement, afficher "incorrect."

8. Considérations pour la sécurité

La cryptographie fondée sur le mot de passe est généralement limitée quant à la sécurité qu'elle peut fournir, en particulier pour les méthodes telles que celles définies dans le présent document où une recherche hors ligne du mot de passe est possible. Bien que l'utilisation du sel et du compte d'itérations puisse augmenter la complexité des attaques (voir les recommandations de la Section 4) il est essentiel que les mots de passe soient bien choisis, et les directives pertinentes (par exemple, [17]) devraient être prises en compte. Il est aussi important que les mots de passe soient bien protégés si ils sont mémorisés.

En général, des clés différentes devraient être déduites d'un mot de passe pour des utilisations différentes afin de minimiser la possibilité d'interactions indésirables. Pour le chiffrement fondé sur le mot de passe avec un seul algorithme, un sel aléatoire est suffisant pour assurer que des clés différentes seront produites. Dans certaines autres situations, comme mentionné à la Section 4, un sel structuré est nécessaire. Les recommandations de la Section 4 devraient donc être prises en compte lors du choix d'une valeur de sel.

9. Adresse de l'auteur

Burt Kaliski
RSA Laboratories
20 Crosby Drive
Bedford, MA 01730 USA
mél : bkaliski@rsasecurity.com

Appendice A. Syntaxe ASN.1

Cette section définit la syntaxe ASN.1 pour les fonctions de déduction de clé, les schémas de chiffrement, le schéma d'authentification de message, et les techniques de prise en charge. L'application prévue pour ces définitions inclut PKCS n° 8 et autres syntaxes de gestion de clé, de données chiffrées, et de données protégées en intégrité. (Divers aspects de l'ASN.1 sont spécifiés dans plusieurs normes ISO/CET [9], [10], [11], [12], [13], [14].)

L'identifiant d'objet pkcs-5 identifie l'arc de l'arborescence d'OID duquel sont dérivés les OID spécifiques de PKCS n° 5 de cette section :

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

A.1 PBKDF1

Aucun identifiant d'objet n'est donné pour PBKDF1, car les identifiants d'objet pour PBES1 sont suffisants pour les applications existantes et PBKDF2 est recommandé pour les nouvelles applications.

A.2 PBKDF2

L'identifiant d'objet id-PBKDF2 identifie la fonction de déduction de clé PBKDF2 (paragraphe 5.2).

IDENTIFIANT D'OBJET id-PBKDF2 ::= {pkcs-5 12}

Le champ Paramètres associé à cet OID dans un AlgorithmIdentifier devra avoir le type PBKDF2-params :

```
PBKDF2-params ::= SEQUENCE {
    sel   CHOIX { CHAÎNE D'OCTETS spécifiée, otherSource AlgorithmIdentifier {{PBKDF2-SaltSources}} },
    iterationCount          ENTIER (1..MAX),
    keyLength    ENTIER (1..MAX) OPTIONNEL,
    prf AlgorithmIdentifier {{PBKDF2-PRFs}} DEFAUT
    algid-hmacWithSHA1 }
```

Les champs de type PKDF2-params ont les significations suivantes :

- sel spécifie la valeur de sel, ou la source de la valeur de sel. Il devra être une chaîne d'octets ou un identifiant d'algorithme avec un OID dans l'ensemble PBKDF2-SaltSources, réservé pour les futures versions de PKCS n° 5.

L'approche de la source du sel est destinée à indiquer comment la valeur de sel doit être générée comme fonction de paramètres dans l'identifiant d'algorithme, les données d'application ou les deux. Par exemple, elle peut indiquer que la valeur de sel est produite à partir du codage d'une structure qui spécifie des informations détaillées sur la clé déduite comme suggéré au^paragraphe 4.1. Certaines des informations peuvent être portées ailleurs, par exemple, dans l'identifiant d'algorithme de chiffrement. Cependant, de telles facilités ne seront fournies que dans une version future de PKCS n° 5.

Dans la présente version, une application peut tirer les avantages mentionnés au paragraphe 4.1 en choisissant une interprétation particulière de la valeur de sel dans l'alternative spécifiée.

IDENTIFIANT-D'ALGORITHME PBKDF2-SaltSources ::= { ... }

- iterationCount spécifie le compte d'itérations. Le compte d'itérations maximum permis dépend de la mise en œuvre. Il est prévu que les profils de mise en œuvre fixent d'autres contraintes aux limites.

- keyLength, un champ facultatif, est la longueur en octets de la clé déduite. La longueur maximum de clé permise dépend de la mise en œuvre ; il est prévu que des profils de mise en œuvre fixent d'autres contraintes. Le champ est fourni par convention ; la longueur de clé n'est pas protégée cryptographiquement. Si il y a des soucis d'interaction entre les opérations de différentes longueurs de clé pour un certain sel (voir au paragraphe 4.1) le sel devrait distinguer les différentes longueurs de clé.

- prf identifie la fonction pseudo aléatoire sous-jacente. Il devra être un identifiant d'algorithme avec un OID dans l'ensemble PBKDF2-PRFs, qui pour cette version de PKCS n° 5 devra consister en id-hmacWithSHA1 (voir l'Appendice B.1.1) et tous autres OID définis par l'application.

IDENTIFIANT-D'ALGORITHME PBKDF2-PRFs ::= { {NUL IDENTIFIÉ PAR id-hmacWithSHA1}, ... }

La fonction pseudo aléatoire par défaut est HMAC-SHA-1 :

algid-hmacWithSHA1 AlgorithmIdentifier {{PBKDF2-PRFs}} ::= {algorithm id-hmacWithSHA1, parameters NULL :

NULL}

A.3 PBES1

Différents identifiants d'objet identifient le schéma de chiffrement PBES1 (paragraphe 6.1) conformément à la fonction de hachage sous-jacente dans la fonction de déduction de clé et au chiffrement de bloc sous-jacent, comme résumé dans le tableau suivant :

Fonction de hachage	Chiffrement de bloc	OID
MD2	DES	pkcs-5.1
MD2	RC2	pkcs-5.4
MD5	DES	pkcs-5.3
MD5	RC2	pkcs-5.6
SHA-1	DES	pkcs-5.10
SHA-1	RC2	pkcs-5.11

IDENTIFIANT D'OBJET pbeWithMD2AndDES-CBC ::= {pkcs-5 1}
IDENTIFIANT D'OBJET pbeWithMD2AndRC2-CBC ::= {pkcs-5 4}
IDENTIFIANT D'OBJET pbeWithMD5AndDES-CBC ::= {pkcs-5 3}
IDENTIFIANT D'OBJET pbeWithMD5AndRC2-CBC ::= {pkcs-5 6}
IDENTIFIANT D'OBJET pbeWithSHA1AndDES-CBC ::= {pkcs-5 10}
IDENTIFIANT D'OBJET pbeWithSHA1AndRC2-CBC ::= {pkcs-5 11}

Pour chaque OID, le champ Paramètres associé à l'OID dans un identifiant d'algorithme devra avoir le type PBEParameter :

```
PBEParameter ::= SEQUENCE {
    sel             CHAÎNE D'OCTETS (TAILLE(8)),
    iterationCount  ENTIER }
```

Le champs de type PBEParameter ont les significations suivantes :
- sel spécifie la valeur de sel, une chaîne de huit octets.
- iterationCount spécifie le compte d'itérations.

A.4 PBES2

L'identifiant d'objet id-PBES2 identifie le schéma de chiffrement PBES2 (paragraphe 6.2).

IDENTIFIANT D'OBJET id-PBES2 ::= {pkcs-5 13}

Le champ Paramètres associé à cet OID dans un identifiant d'algorithme devra avoir le type PBES2-params :

```
PBES2-params ::= SEQUENCE {
    keyDerivationFunc    AlgorithmIdentifier {{PBES2-KDFs}},
    encryptionScheme     AlgorithmIdentifier {{PBES2-Encs}} }
```

Les champs de type PBES2-params ont les significations suivantes :

- keyDerivationFunc identifie la fonction de déduction de clé sous-jacente. Ce devra être un identifiant d'algorithme avec un OID dans l'ensemble PBES2-KDFs, qui pour cette version de PKCS n° 5 doit être id-PBKDF2 (Appendice A.2).

  IDENTIFIANT D'ALGORITHME PBES2-KDFs -::= { {PBKDF2-params IDENTIFIÉ PAR id-PBKDF2}, ... }

- encryptionScheme identifie le schéma de chiffrement sous-jacent. Ce devra être un identifiant d'algorithme avec un OID dans l'ensemble PBES2-Encs, dont la définition est laissée à l'application. Des exemples de schémas de chiffrement sous-jacents sont donnés à l'Appendice B.2.

  IDENTIFIANT D'ALGORITHME PBES2-Encs ::= { ... }

A.5 PBMAC1

L'identifiant d'objet id-PBMAC1 identifie le schéma d'authentification de message PBMAC1 (paragraphe 7.1).

IDENTIFIANT D'OBJET id-PBMAC1 ::= {pkcs-5 14}

Le champ Paramètres associé à cet OID dans un identifiant d'algorithme devra avoir le type PBMAC1-params :

```
PBMAC1-params ::=  SEQUENCE {
    keyDerivationFunc       AlgorithmIdentifier {{PBMAC1-KDFs}},
    messageAuthScheme       AlgorithmIdentifier {{PBMAC1-MACs}} }
```

Le champ keyDerivationFunc a la même signification que le champ correspondant de PBES2-params (Appendice A.4) sauf que l'ensemble des OID est PBMAC1-KDFs.

IDENTIFIANT-D'ALGORITHME PBMAC1-KDFs ::= { {PBKDF2-params IDENTIFIÉ PAR id-PBKDF2}, ... }

Le champ messageAuthScheme identifie le schéma sous-jacent d'authentification de message. Il devra avoir un identifiant d'algorithme avec un OID dans l'ensemble PBMAC1-MACs, dont la définition est laissée à l'application. Des exemples de schéma de chiffrement sous-jacent sont données à l'Appendice B.3.

IDENTIFIANT-D'ALGORITHME PBMAC1-MACs ::= { ... }

Appendice B. Techniques de prise en charge

Cette section donne plusieurs exemples de fonctions et schémas sous-jacents qui prennent en charge les schémas fondés sur le mot de passe des Sections 5, 6 et 7.

Bien que les techniques de prise en charge soient appropriées pour que les applications les mettent en œuvre, aucune d'elles n'est exigée. On espère, cependant, que des profils pour PKCS n° 5 seront développés qui spécifieront des techniques de prise en charge particulières.

Cette section donne aussi des identifiants d'objets pour les techniques de prise en charge. Les identifiants d'objet digestAlgorithm et encryptionAlgorithm identifient les arcs d'où sont déduits certains OID d'algorithmes référencés dans cette section :

IDENTIFIANT D'OBJET digestAlgorithm ::= {rsadsi 2}
IDENTIFIANT D'OBJET encryptionAlgorithm ::= {rsadsi 3}

B.1 Fonctions pseudo aléatoires

Un exemple de fonction pseudo aléatoire pour PBKDF2 (paragraphe 5.2) est HMAC-SHA-1.

B.1.1 HMAC-SHA-1

HMAC-SHA-1 est la fonction pseudo aléatoire qui correspond au code d'authentification de message HMAC [7] fondé sur la fonction de hachage SHA-1 [18]. La fonction pseudo aléatoire est celle par laquelle est calculé le code authentification de message, avec un résultat de pleine longueur. (Le premier argument à la fonction PRF pseudo aléatoire sert de "clé" de HMAC, et le second sert de "texte" de HMAC. Dans le cas de PBKDF2, la "clé" est donc le mot de passe et le "texte" est le sel.) HMAC-SHA-1 a une longueur de clé variable et une valeur de résultat de 20 octets (160 bits).

Bien que la longueur de la clé pour HMAC-SHA-1 soit par nature non limitée, l'espace de recherche effectif pour les résultats de la fonction pseudo aléatoire peut être limité par la structure de la fonction. En particulier, lorsque la clé est plus longue que 512 bits, HMAC-SHA-1 va d'abord la couper à 160 bits. Donc, même si une longue clé déduite comportant plusieurs résultats de fonction pseudo aléatoire est produite à partir d'une clé, l'espace effectif de recherche pour la clé déduite sera au plus de 160 bits. Bien que la limitation spécifique pour les autres tailles de clé dépende des détails de la construction HMAC, on devrait supposer, pour rester prudent, que l'espace effectif de recherche est aussi limité à 160 bits pour les autres tailles de clé.

(La limitation à 160 bits ne devrait généralement pas poser de limitation pratique dans le cas de la cryptographie fondée sur le mot de passe, car l'espace de recherche pour un mot de passe a peu de chances de dépasser 160 bits.)

L'identifiant d'objet id-hmacWithSHA1 identifie la fonction pseudo aléatoire HMAC-SHA-1 :

IDENTIFIANT D'OBJET id-hmacWithSHA1 ::= {digestAlgorithm 7}

Le champ Paramètres associé à cet OID dans un identifiant d'algorithme devra avoir le type NUL. Cet identifiant d'objet est employé dans l'ensemble d'objets PBKDF2-PRFs (Appendice A.2).

Note : Bien que HMAC-SHA-1 ait été conçu comme un code d'authentification de message, sa preuve de sécurité est déjà modifiée pour s'accommoder des exigences d'une fonction pseudo aléatoire, avec des hypothèses plus fortes.

Une fonction de hachage peut aussi satisfaire aux exigences d'une fonction pseudo aléatoire avec certaines hypothèses. Par exemple, l'application directe d'une fonction de hachage à l'enchaînement de la "clé" et du "texte" peut être appropriée, pourvu que le "texte" ait une structure appropriée pour empêcher certaines attaques. HMAC-SHA-1 est cependant préférable, parce qu'il traite "clé" et "texte" comme des arguments distincts et n'exige pas que "texte" ait de structure.

B.2 Schémas de chiffrement

Les exemples de fonctions pseudo aléatoires pour PBES2 (paragraphe 6.2) sont DES-CBC-Pad, DES-EDE2-CBC-Pad, RC2-CBC-Pad, et RC5-CBC-Pad.

L'identifiant d'objet donné dans cette section est destiné à être employé dans l'ensemble d'objets PBES2-Encs (Appendice A.4).

B.2.1 DES-CBC-Pad

DES-CBC-Pad est un DES à une seule clé [15] en mode CBC [16] avec l'opération de bourrage de la RFC1423 (voir au paragraphe 6.1.1). DES-CBC-Pad a une clé de chiffrement de huit octets et un vecteur d'initialisation de huit octets. La clé est considérée comme un codage de 64 bits d'une clé DES de 56 bits en ignorant les bits de parité.

L'identifiant d'objet desCBC (défini dans les accords de l'atelier de mise en œuvre du NIST/OSI) identifie le schéma de chiffrement DES-CBC-Pad :

IDENTIFIANT D'OBJET desCBC ::= {iso(1) identified-organization(3) oiw(14) secsig(3) algorithms(2) 7}

Le champ Paramètres associé à cet OID dans un identifiant d'algorithme devra avoir le type CHAÎNE D'OCTETS (TAILLE(8)) spécifiant le vecteur d'initialisation pour le mode CBC.

B.2.2 DES-EDE3-CBC-Pad

DES-EDE3-CBC-Pad est un triple-DES à trois clés en mode CBC [1] avec l'opération de bourrage de la RFC1423. DES-EDE3-CBC-Pad a une clé de chiffrement de 24 octets et un vecteur d'initialisation de huit octets. La clé est considérée comme l'enchaînement de trois clés de huit octets, chacune d'elles étant un codage de 64 bits d'une clé DES de 56 bits avec les bits de parité ignorés.

L'identifiant d'objet des-EDE3-CBC identifie le schéma de chiffrement DES-EDE3-CBC-Pad :

IDENTIFIANT D'OBJET des-EDE3-CBC ::= {encryptionAlgorithm 7}

Le champ Paramètres associé à cet OID dans un identifiant d'algorithme devra avoir le type CHAÎNE D'OCTETS (TAILLE(8)) spécifiant le vecteur d'initialisation pour le mode CBC.

Note : Un OID pour DES-EDE3-CBC sans bourrage est donné dans ANSI X9.52 [1] ; celui qui est donné ici est préféré car il spécifie le bourrage.

B.2.3 RC2-CBC-Pad

RC2-CBC-Pad est l'algorithme de chiffrement RC2(tm) [21] en mode CBC avec l'opération de bourrage de la RFC1423.

RC2-CBC-Pad a une longueur de clé variable, de un à 128 octets, un paramètre séparé "bits de clé efficace" de un à 1024 bits qui limite l'espace de recherche effectif indépendamment de la longueur de la clé, et un vecteur d'initialisation de huit octets.

L'identifiant d'objet rc2CBC identifie le schéma de chiffrement RC2-CBC-Pad :

IDENTIFIANT D'OBJET rc2CBC ::= {encryptionAlgorithm 2}

Le champ Paramètres associé à cet OID dans un identifiant d'algorithme devra avoir le type RC2-CBC-Parameter :

```
RC2-CBC-Parameter ::= SEQUENCE {
    rc2ParameterVersion  ENTIER OPTIONNEL,
    iv                   CHAÎNE D'OCTETS (TAILLE(8)) }
```

Les champs de type RC2-CBCParameter ont les significations suivantes :

- rc2ParameterVersion est un codage breveté de RSA Security Inc. des "bits de clé efficaces" pour RC2. Les codages suivants sont définis :

Bits de clé efficaces	Codage
40	160
64	120
128	58
b > 256	b

Si le champ rc2ParameterVersion est omis, les "bits de clé efficaces" prennent par défaut la valeur 32. (Ceci pour la rétro compatibilité avec certaines très anciennes mises en œuvre.)

- iv est le vecteur d'initialisation de huit octets.

B.2.4 RC5-CBC-Pad

RC5-CBC-Pad est l'algorithme de chiffrement RC5(tm) [20] en mode CBC avec une généralisation de l'opération de bourrage de la RFC1423. Ce schéma est pleinement spécifié dans [2]. RC5-CBC-Pad a une longueur de clé variable, de 0 à 256 octets, et accepte les deux tailles de bloc de 64 bits et de 128 bits. Pour la première, il a un vecteur d'initialisation de huit octets, et pour le second, un vecteur d'initialisation de 16 octets. RC5-CBC-Pad a aussi un nombre variable de "tours" dans l'opération de chiffrement, de 8 à 127.

Note : La généralisation de l'opération de bourrage est la suivante. Pour RC5 avec une taille de bloc de 64 bits, la chaîne de bourrage est comme défini dans la RFC1423. Pour RC5 avec la taille de bloc de 128 bits, la chaîne de bourrage consiste en 16-(||M|| mod 16) octets ayant chacun la valeur 16-(||M|| mod 16).

L'identifiant d'objet rc5-CBC-PAD [2] identifie le schéma de chiffrement RC5-CBC-Pad :

IDENTIFIANT D'OBJET rc5-CBC-PAD ::= {encryptionAlgorithm 9}

Le champ Paramètres associé à cet OID dans un identifiant d'algorithme devra avoir le type RC5-CBC-Parameters :

```
RC5-CBC-Parameters ::= SEQUENCE {
    version          ENTIER {v1-0(16)} (v1-0),
    rounds           ENTIER (8..127),
    blockSizeInBits  ENTIER (64 | 128),
    iv               CHAÎNE D'OCTETS OPTIONNEL }
```

Les champs de type RC5-CBC-Parameters ont les significations suivantes :
- version est la version de l'algorithme, qui devra être v1-0.
- rounds est le nombre de tours dans l'opération de chiffrement, qui devra être entre 8 et 127.
- blockSizeInBits est la taille de bloc en bits, qui devra être 64 ou 128.
- iv est le vecteur d'initialisation, une chaîne d'octets de huit octets pour RC5 à 64 bits et une chaîne de 16 octets pour RC5 à 128 bits. La chaîne par défaut est faite d'octets à zéro pour la longueur appropriée.

B.3 Schémas d'authentification de message

Un exemple de schéma d'authentification de message pour PBMAC1 (paragraphe 7.1) est HMAC-SHA-1.

B.3.1 HMAC-SHA-1

HMAC-SHA-1 est le schéma d'authentification de message HMAC [7] fondé sur la fonction de hachage SHA-1 [18]. HMAC-SHA-1 a une longueur de clé variable et un code d'authentification de message de 20 octets (160 bits).

L'identifiant d'objet id-hmacWithSHA1 (voir l'Appendice B.1.1) identifie le schéma d'authentification de message HMAC-SHA-1. (L'identifiant d'objet est le même pour la fonction pseudo aléatoire et le schéma d'authentification de message ; la distinction se fait par le contexte.) Cet identifiant d'objet est destiné à être utilisé dans l'ensemble d'objets PBMAC1-Macs (Appendice A.5).

Appendice C. Module ASN.1

Pour les besoins de référence, la syntaxe ASN.1 de l'Appendice A est présentée ici comme module ASN.1.

```
-- Module ASN.1 PKCS n° 5 v2.0
-- Révisé le 25 mars 1999
-- La conformité de ce module à la norme ASN.1 a été vérifiée par les outils ASN.1 OSS.

PKCS5v2-0 {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-5(5) modules(16) pkcs5v2-0(1)}

DEFINITIONS ::= BEGIN

-- Identifiants des objets de base

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

-- Types et classes de base

AlgorithmIdentifier { IDENTIFIANT D'ALGORITHME:InfoObjectSet } ::=
    SEQUENCE {
      algorithm IDENTIFIANT D'ALGORITHME.&id({InfoObjectSet}),
      parameters IDENTIFIANT D'ALGORITHME.&Type({InfoObjectSet}
      {@algorithm}) OPTIONNEL
  }

IDENTIFIANT D'ALGORITHME ::= IDENTIFIANT DE TYPE

-- PBKDF2

IDENTIFIANT D'ALGORITHME PBKDF2Algorithms ::= { {PBKDF2-params IDENTIFIÉ PAR id-PBKDF2}, ...}

id-PBKDF2 IDENTIFIANT D'OBJET ::= {pkcs-5 12}

algid-hmacWithSHA1 AlgorithmIdentifier {{PBKDF2-PRFs}} ::= {algorithm id-hmacWithSHA1, parameters NULL :
                                                              NULL}

PBKDF2-params ::= SEQUENCE {
      sel          CHOIX { CHAÎNE D'OCTETS spécifiée, otherSource AlgorithmIdentifier {{PBKDF2-SaltSources}} },
      iterationCount      ENTIER (1..MAX),
      keyLength           ENTIER (1..MAX) OPTIONNEL,
      prf AlgorithmIdentifier {{PBKDF2-PRFs}} DEFAUT
      algid-hmacWithSHA1
  }

IDENTIFIANT D'ALGORITHME PBKDF2-SaltSources ::= { ... }
```

```
IDENTIFIANT D'ALGORITHME PBKDF2-PRFs ::= { {NULL IDENTIFIED BY id-hmacWithSHA1}, ... }

-- PBES1

IDENTIFIANT D'ALGORITHME PBES1Algorithms ::= {
    {PBEParameter IDENTIFIÉ PAR pbeWithMD2AndDES-CBC}  |
    {PBEParameter IDENTIFIÉ PAR pbeWithMD2AndRC2-CBC}  |
    {PBEParameter IDENTIFIÉ PAR pbeWithMD5AndDES-CBC}  |
    {PBEParameter IDENTIFIÉ PAR pbeWithMD5AndRC2-CBC}  |
    {PBEParameter IDENTIFIÉ PAR pbeWithSHA1AndDES-CBC} |
    {PBEParameter IDENTIFIÉ PAR pbeWithSHA1AndRC2-CBC},
    ...
  }

IDENTIFIANT D'OBJET pbeWithMD2AndDES-CBC  ::= {pkcs-5 1}
IDENTIFIANT D'OBJET pbeWithMD2AndRC2-CBC ::= {pkcs-5 4}
IDENTIFIANT D'OBJET pbeWithMD5AndDES-CBC ::= {pkcs-5 3}
IDENTIFIANT D'OBJET pbeWithMD5AndRC2-CBC ::= {pkcs-5 6}
IDENTIFIANT D'OBJET pbeWithSHA1AndDES-CBC ::= {pkcs-5 10}
IDENTIFIANT D'OBJET pbeWithSHA1AndRC2-CBC ::= {pkcs-5 11}

PBEParameter ::= SEQUENCE {
    sel                 CHAÎNE D'OCTETS (TAILLE(8)),
    iterationCount      ENTIER
  }

-- PBES2

IDENTIFIANT D'ALGORITHME PBES2Algorithms ::= { {PBES2-params IDENTIFIÉ PAR id-PBES2}, ...}

IDENTIFIANT D'OBJET  id-PBES2 ::= {pkcs-5 13}

PBES2-params ::= SEQUENCE {
    keyDerivationFunc    AlgorithmIdentifier {{PBES2-KDFs}},
    encryptionScheme     AlgorithmIdentifier {{PBES2-Encs}}
  }

IDENTIFIANT D'ALGORITHME PBES2-KDFs ::= { {PBKDF2-params IDENTIFIÉ PAR id-PBKDF2}, ... }

IDENTIFIANT D'ALGORITHME PBES2-Encs ::= { ... }

-- PBMAC1

IDENTIFIANT D'ALGORITHME PBMAC1Algorithms ::= { {PBMAC1-params IDENTIFIÉ PAR id-PBMAC1}, ...}

id-PBMAC1 IDENTIFIANT D'OBJET ::= {pkcs-5 14}

PBMAC1-params ::=  SEQUENCE {
                                keyDerivationFunc       AlgorithmIdentifier {{PBMAC1-KDFs}},
                                messageAuthScheme     AlgorithmIdentifier {{PBMAC1-MACs}}
  }

IDENTIFIANT D'ALGORITHME PBMAC1-KDFs ::= { {PBKDF2-params IDENTIFIÉ PAR id-PBKDF2}, ... }

IDENTIFIANT D'ALGORITHME PBMAC1-MACs ::= { ... }

-- Techniques de prise en charge

IDENTIFIANT D'OBJET digestAlgorithm ::= {rsadsi 2}
IDENTIFIANT D'OBJET encryptionAlgorithm ::= {rsadsi 3}
```

```
IDENTIFIANT D'ALGORITHME SupportingAlgorithms ::= {
                                        {NULL IDENTIFIÉ PAR id-hmacWithSHA1} |
                                        {CHAÎNE D'OCTETS (TAILLE(8)) IDENTIFIÉ PAR desCBC} |
                                        {CHAÎNE D'OCTETS (TAILLE(8)) IDENTIFIÉ PAR des-EDE3-CBC} |
                                        {RC2-CBC-Parameter IDENTIFIÉ PAR rc2CBC} |
                                        {RC5-CBC-Parameters IDENTIFIÉ PAR rc5-CBC-PAD},
                                        ...
                                        }

IDENTIFIANT D'OBJET  id-hmacWithSHA1 ::= {digestAlgorithm 7}

IDENTIFIANT D'OBJET desCBC ::= {iso(1) identified-organization(3) oiw(14) secsig(3) algorithms(2) 7}    - de OIW

IDENTIFIANT D'OBJET des-EDE3-CBC ::= {encryptionAlgorithm 7}

IDENTIFIANT D'OBJET rc2CBC ::= {encryptionAlgorithm 2}

RC2-CBC-Parameter ::= SEQUENCE {
    rc2ParameterVersion     ENTIER OPTIONNEL,
    iv                      CHAÎNE D'OCTETS (TAILLE(8))
  }

IDENTIFIANT D'OBJET rc5-CBC-PAD ::= {encryptionAlgorithm 9}

RC5-CBC-Parameters ::= SEQUENCE {
    version                 ENTIER {v1-0(16)} (v1-0),
    rounds                  ENTIER (8..127),
    blockSizeInBits         ENTIER (64 | 128),
    iv                      CHAÎNE D'OCTETS OPTIONNEL
  }

FIN
```

Considérations de propriété intellectuelle

RSA Security ne fait pas de revendication de brevet sur les constructions générales décrites dans le présent document, bien que les techniques sous-jacentes spécifiques puissent être couvertes. Parmi les techniques sous-jacentes, l'algorithme de chiffrement RC5 (Appendice B.2.4) est protégé par les brevets U.S. 5 724 428 [22] et 5 835 600 [23].

RC2 et RC5 sont des marques déposées de RSA Security.

Licence de copier le présent document est accordée à condition qu'il soit identifié comme la norme de cryptographie à clé publique (PKCS) de RSA Security Inc. dans tout matériel mentionnant ou faisant référence au présent document.

RSA Security ne fait pas de déclaration concernant les revendications de propriété intellectuelle par d'autres parties. Une telle détermination est de la responsabilité de l'utilisateur.

Historique des révisions

Versions 1.0-1.3
Les versions de 1.0 à 1.3 ont été distribuées aux participants aux réunions de RSA Data Security Inc sur les normes de cryptographie à clé publique en février et mars 1991.

Version 1.4
La version 1.4 faisait partie de la publication initiale du 3 juin 1991 de PKCS. La version 1.4 a été publiée comme document de l'atelier de mise en œuvre NIST/OSI SEC-SIG-91-20.

Version 1.5
La version 1.5 incorporait plusieurs amendements rédactionnels, y compris des mises à jour de références et l'ajout d'un historique des révisions.

Version 2.0
La version 2.0 incorpore des changements rédactionnels majeurs en termes de structure de document, et introduit le schéma de chiffrement PBES2, le schéma d'authentification de message PBMAC1, et les fonctions de déduction de clé indépendantes fondées sur le mot de passe. Cette version continue de prendre en charge le processus de chiffrement de la version 1.5.

Références

[1] American National Standard X9.52 - 1998, "Triple Data Encryption Algorithm Modes of Operation". Document de travail du comité de normalisation X9, 27 juillet 1998.

[2] R. Baldwin et R. Rivest, "Algorithmes RC5, RC5-CBC, RC5-CBC-Pad, et RC5-CTS", RFC2040, octobre 1996. *(Information)*

[3] D. Balenson, D., "Amélioration de la confidentialité pour la messagerie électronique Internet : Partie III -- Algorithmes, modes et identifiants", RFC1423, février 1993. *(Historique)*

[4] S.M. Bellovin et M. Merritt. "Encrypted key exchange: Password-based protocols secure against dictionary attacks". Dans Proceedings of the 1992 IEEE Computer Society Conference on Research in Security et Privacy, pages 72-84, IEEE Computer Society, 1992.

[5] D. Jablon. "Strong mot de passe-only authenticated key exchange". ACM Computer Communications Review, octobre 1996.

[6] B. Kaliski, "Algorithme de résumé de message MD2", RFC1319, avril 1992. *(Historique, Information)*

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

[8] Robert Morris et Ken Thompson. "Password security: A case history". Communications de ACM, 22(11):594-597, novembre 1979.

[9] ISO/CEI 8824-1:1995, "Information technology - Abstract Syntax Notation One (ASN.1) - Specification of basic notation". 1995.

[10] ISO/CEI 8824-1:1995/Amd.1:1995, "Information technology – Abstract Syntax Notation One (ASN.1) - Specification of basic notation - Amendment 1 - Rules of extensibility". 1995.

[11] ISO/CEI 8824-2:1995 "Information technology - Abstract Syntax Notation One (ASN.1) - Information object specification". 1995.

[12] ISO/CEI 8824-2:1995/Amd.1:1995 "Information technology - Abstract Syntax Notation One (ASN.1) - Information object specification - Amendment 1 - Rules of extensibility". 1995.

[13] ISO/CEI 8824-3:1995 "Information technology - Abstract Syntax Notation One (ASN.1) - Constraint specification". 1995.

[14] ISO/CEI 8824-4:1995 "Information technology - Abstract Syntax Notation One (ASN.1) - Parameterization of ASN.1 specifications". 1995.

[15] National Institute of Standards et Technology (NIST). FIPS PUB 46-2: "Data Encryption Standard". 30 décembre 1993.

[16] National Institute of Standards et Technology (NIST). FIPS PUB 81: "DES Modes of Operation". 2 décembre 1980.

[17] National Institute of Standards et Technology (NIST). FIPS PUB 112: "Password Usage". 30 mai 1985.

[18] National Institute of Standards et Technology (NIST). FIPS PUB 180-1: "Secure Hash Standard". avril 1994.

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

[20] R.L. Rivest. "The RC5 encryption algorithm". Dans Proceedings of the Second International Workshop on Fast Software Encryption, pages 86-96, Springer-Verlag, 1994.

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

[22] R.L. Rivest. "Block-Encryption Algorithm with Data-Dependent Rotations". Brevet U.S. n° 5,724,428, 3 mars 1998.

[23] R.L. Rivest. "Block Encryption Algorithm with Data-Dependent Rotations". Brevet U.S. n° 5,835,600, 10 novembre 1998.

[24] RSA Laboratories. "PKCS #5: Password-Based Encryption Standard". Version 1.5, novembre 1993.

[25] RSA Laboratories. "PKCS #8: Private-Key Information Syntax Standard. Version 1.2", novembre 1993.

[26] T. Wu. "The Secure Remote Password protocol". Dans Proceedings of the 1998 Internet Society Network et Distributed System Security Symposium, pages 97-111, Internet Society, 1998.

[27] F. Yergeau, "UTF-8, un format de transformation de la norme ISO 10646", RFC2279, janvier 1998. (*Obsolète, voir* RFC3629) *(D.S.)*

Information de contact sur PKCS

Les normes de cryptographie à clé publique sont des spécifications produites par RSA Laboratories en coopération avec les développeurs de systèmes sécurisés du monde entier dans le but d'accélérer le déploiement de la cryptographie à clé publique. D'abord publiés en 1991 à la suite de réunions avec un petit groupe d'adeptes précoces de la technologie des clés publiques, les documents PKCS sont devenus largement référencés et mis en œuvre. Les contributions provenant des séries PKCS sont devenues des parties de nombreuses normes formelles et standard de faits, incluant les documents ANSI X9, PKIX, SET, S/MIME, et SSL.

D'autres développements de PKCS se font au travers de discussions sur des listes de diffusion et d'ateliers de travail occasionnels, et des suggestions d'amélioration sont toujours bienvenues. Pour en savoir plus, contacter :

PKCS Editor
RSA Laboratories
20 Crosby Drive
Bedford, MA 01730
USA
pkcs-editor@rsasecurity.com
http://www.rsalabs.com/pkcs/

Déclaration de droits de reproduction

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