Groupe de travail Réseau
Request for Comments : 4010
Catégorie : En cours de normalisation
Traduction Claude Brière de L'Isle

J. Park, S. Lee, J. Kim & J. Lee
KISA
février 2005

Utilisation de l'algorithme de chiffrement SEED en syntaxe de message cryptographique (CMS)

Résumé
Le présent document spécifie les conventions d'utilisation de l'algorithme de chiffrement SEED pour un chiffrement avec la syntaxe de message cryptographique (CMS, *Cryptographic Message Syntax*).

SEED est ajouté à l'ensemble des algorithmes facultatifs de chiffrement symétrique en CMS en fournissant deux classes d'identifiants d'objet (OID, *object identifier*) uniques. Une classe d'OID définit les algorithmes de chiffrement de contenu et l'autre définit les algorithmes de chiffrement de clé.

1. Introduction

Le présent document spécifie les conventions d'utilisation des algorithmes de chiffrement SEED [RFC4009], [TTASSEED] pour le chiffrement avec la syntaxe de message cryptographique (CMS, *Cryptographic Message Syntax*) [RFC3852]. Les identifiants d'objet (OID) pertinents et les étapes de traitement sont fournis afin que SEED puisse être utilisé dans la spécification de CMS (RFC 3852, RFC 3370) pour le chiffrement du contenu et des clés.

1.1 SEED

SEED est un algorithme de chiffrement symétrique développé par l'agence coréenne de sécurité de l'information (KISA, *Korea Information Security Agency*) et un groupe d'experts depuis 1998. La taille de bloc d'entrée/sortie et la longueur de clé de SEED est de 128 bits. SEED a la structure de Feistel à 16 tours. Une entrée de 128 bits est divisée en deux blocs de 64 bit s et le bloc de 64 bits de droite est entré dans la fonction circulaire, avec une sous clé de 64 bits générée à partir du programmeur de clé.

SEED est facilement mis en œuvre dans divers logiciels et matériels parce qu'il prend moins de mémoire à mettre en œuvre que les autres algorithmes et génère des clés sans dégrader la sécurité de l'algorithme. En particulier, il peut être effectivement adopté dans un environnement de calcul de ressources restreintes, comme des appareils mobiles et des cartes à mémoire.

SEED est robuste contre les attaques connues incluant la cryptanalyse différentielle (DC, *Differential cryptanalysis*), la cryptanalyse linéaire (LC, *Linear cryptanalysis*), et les attaques de clé qui s'y rapportent. SEED est passé par de larges procédures d'examen public. Il a été évalué et est considéré comme cryptographiquement sûr par des organisations crédibles telles que l'ISO/CEI JTC 1/SC 27 et le comité d'évaluation et de recherche cryptographique (CRYPTREC, *Cryptography Research and Evaluation Committee*) du Japon [ISOSEED], [CRYPTREC].

SEED est un standard national d'associations industrielles [TTASSEED] et est largement utilisé en Corée du Sud pour le commerce électronique et les services financiers qui fonctionnent sur les communications par réseaux filaires et sans fil.

1.2 Terminologie

Dans le présent document, les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS",

"DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" sont à interpréter comme décrit dans la [RFC2119].

2. Identifiants d'objet pour chiffrement de contenu et de clé

La présente section donne les OID et les informations de traitement nécessaires pour que SEED soit utilisé pou le chiffrement du contenu et des clés dans CMS. SEED est ajouté à l'ensemble des algorithmes facultatifs de chiffrement symétrique dans la CMS en fournissant deux classes d'identifiants d'objet (OID, *object identifier*) uniques. Une classe d'OID définit les algorithmes de chiffrement de contenu et l'autre définit les algorithmes de chiffrement de clé. Donc, un agent de CMS peut appliquer SEED soit pour le chiffrement du contenu , soit pour le chiffrement de clé en choisissant l'identifiant d'objet correspondant, en fournissant les paramètres requis, et en lançant le code de programme.

2.1 OID pour chiffrement du contenu

SEED est ajouté à l'ensemble des algorithmes de chiffrement symétrique de contenu défini dans la [RFC3370]. L'algorithme de chiffrement de contenu SEED en mode de chaînage de bloc de chiffrement (CBC, *Cipher Block Chaining*) a l'identifiant d'objet suivant :

IDENTIFIANT D'OBJET id-seedCBC ::= { iso(1) member-body(2) korea(410) kisa(200004) algorithm(1) seedCBC(4) }

Le champ de paramètres AlgorithmIdentifier DOIT être présent, et le champ Paramètres DOIT contenir la valeur d'initialisation (IV) :

SeedCBCParameter ::= SeedIV -- Valeur d'initialisation

SeedIV ::= CHAÎNE D'OCTETS (TAILLE(16))

Le texte source est bourré conformément au paragraphe 6.3 de la [RFC3852].

2.2 OID pour chiffrement de clé

Les procédures d'enveloppement/désenveloppement de clé utilisées pour chiffrer/déchiffrer une clé de chiffrement de contenu (CEK, *content-encryption key*) SEED avec une clé de chiffrement de clé (KEK, *key-encryption key*) SEED sont spécifiées à la Section 3. La génération et la distribution des clés de chiffrement de clé sortent du domaine d'application du présent document.

L'algorithme de chiffrement de clé SEED a l'identifiant d'objet suivant :

IDENTIFIANT D'OBJET id-npki-app-cmsSeed-wrap ::= { iso(1) member-body(2) korea(410) kisa(200004) npki-app(7)
smime(1) alg(1) cmsSEED-wrap(1) }

Le paramètre associé à cet identifiant d'objet DOIT être absent, parce que la procédure d'enveloppement de clé elle-même définit comment et quand utiliser une IV.

3. Algorithme d'enveloppement de clé

L'enveloppement et le désenveloppement de clé SEED est fait conformément à l'algorithme d'enveloppement de clé AES [RFC3394].

3.1. Notation et définitions

La notation suivante est utilisée dans la description des algorithmes d'enveloppement de clé :

SEED(K, W) : chiffre W en utilisant le livre de code SEED avec la clé K.
SEED-1(K, W) : déchiffre W en utilisant le livre de code SEED avec la clé K.
MSB(j, W) : retourne les j bits de poids fort de W.
LSB(j, W) : retourne les j bits de moindre poids de W.

B1 ^ B2 : le OUX au bit près de B1 et B2.
B1 | B2 : enchaîne B1 et B2.
K : clé de chiffrement de clé K.
n : nombre de blocs de données de clé de 64 bits.
s : nombre d'étapes dans le processus d'enveloppement, s = 6n.
P[i] : ième bloc de données de clé en texte source (de 64 bits) où i = 0, 1, 2, ..., n.
C[i] : ième bloc de données de texte chiffré (de 64 bits) où i = 0, 1, 2, ..., n.
A : registre de 64 bits de vérification d'intégrité.
R[i] : dispositif de registres de 64 bits où i = 0, 1, 2, ..., n.
A[t], R[i][t] : contenus des registres A et R[i] après l'étape de chiffrement t.
IV : valeur initiale de 64 bits utilisée durant le processus d'enveloppement.

Dans l'algorithme d'enveloppement de clé, la fonction d'enchaînement sera utilisée pour enchaîner des quantités de 64 bits pour former l'entrée de 128 bits au livre de code SEED. Les fonctions d'extraction seront utilisées pour partager le résultat de 128 bits provenant du livre de code SEED en deux quantités de 64 bits.

3.2 Enveloppement de clé SEED

L'enveloppement de clé avec SEED est identique à celui du paragraphe 2.2.1 de la [RFC3394] en remplaçant "AES" par "SEED".

L'entrée du processus d'enveloppement de clé est la KEK et le texte source à envelopper. Le texte source consiste en blocs de 64 bits contenant les données de clé qui sont enveloppées. Le processus d'enveloppement de clé est décrit ci-dessous :

Entrées : texte source, n valeurs de 64 bits {P[1], P[2], ..., P[n]}, et clé, K (la KEK).
Résultats : texte chiffré, (n+1) valeurs de 64 bits {C[0], C[1], ..., C[n]}.

1) Initialiser les variables.
Régler A[0] à une valeur initiale (voir le paragraphe 3.4)
  Pour i = 1 à n
    R[0][i] = P[i]

2) Calculer les valeurs intermédiaires.
  Pour t = 1 à s, où s = 6n
    A[t] = MSB(64, SEED(K, A[t-1] | R[t-1][1])) ^ t
    Pour i = 1 à n-1 R[t][i] = R[t-1][i+1]
    R[t][n] = LSB(64, SEED(K, A[t-1] | R[t-1][1]))

3) Sortir les résultats.
  Régler C[0] = A[s]
  Pour i = 1 à n
    C[i] = R[s][i]

Une autre description de l'algorithme d'enveloppement de clé implique d'indexer plutôt que de glisser. Cette approche permet de calculer la clé enveloppée en place, évitant la rotation dans la description précédente. Cela produit un résultat identique et est plus facilement mis en œuvre dans le logiciel.

Entrées : texte source, n valeurs de 64bits {P[1], P[2], ..., P[n]}, et la clé, K (la KEK).
Résultats : texte chiffré, (n+1) valeurs de 64 bits {C[0], C[1], ..., C[n]}.

1) Initialiser les variables.
  Régler A = IV, une valeur initiale (voir le paragraphe 3.4)
  Pour i = 1 à n
    R[i] = P[i]

2) Calculer les valeurs intermédiaires.
  Pour j = 0 à 5
    Pour i=1 à n
      B = SEED(K, A | R[i])
      A = MSB(64, B) ^ t où t = (n*j)+i
      R[i] = LSB(64, B)

3) Sortir les résultats.
Régler C[0] = A
Pour i = 1 à n
C[i] = R[i]

3.3 Désenveloppement de clé SEED

Le désenveloppement de clé avec SEED est identique à celui du paragraphe 2.2.2 de la [RFC3394], en remplaçant "AES" par "SEED".

Les entrées du processus de désenveloppement sont la KEK et (n+1) blocs de 64 bits de texte chiffré consistant en la clé précédemment enveloppée. Il retourne n blocs de texte source consistant en les n blocs de 64 bits des données de clé déchiffrée.

Entrées : texte chiffré, (n+1) valeurs de 64 bits {C[0], C[1], ..., C[n]}, et la clé, K (la KEK).
Résultats : texte source, n valeurs de 64 bits {P[1], P[2], ..., P[n]}.

1) Initialiser les variables.
Régler A[s] = C[0] où s = 6n
Pour i = 1 à n
R[s][i] = C[i]

2) Calculer les valeurs intermédiaires.
Pour t = s à 1
A[t-1] = MSB(64, SEED-1(K, ((A[t] ^ t) | R[t][n]))
R[t-1][1] = LSB(64, SEED-1(K, ((A[t]^t) | R[t][n]))
Pour i = 2 à n
R[t-1][i] = R[t][i-1]

3) Sortir les résultats.
Si A[0] est une valeur initiale appropriée (voir au paragraphe 3.4),
Alors
Pour i = 1 à n
P[i] = R[0][i]
Autrement
Retourner une erreur

L'algorithme de désenveloppement peut aussi être spécifié comme une opération fondée sur l'indexation, permettant de faire les calculs à la place. Là encore, cela produit le même résultat que l'approche du glissement de registre.

Entrées : texte chiffré, (n+1) valeurs de 64 bits {C[0], C[1], ..., C[n]}, et la clé, K (la KEK).
Résultats : texte source, n valeurs de 64 bits {P[0], P[1], ..., P[n]}.

1) Initialiser les variables.
Régler A = C[0]
Pour i = 1 à n
R[i] = C[i]

2) Calculer les valeurs intermédiaires.
Pour j = 5 à 0
Pour i = n à 1
B = SEED-1(K, (A ^ t) | R[i]) où t = n*j+i
A = MSB(64, B)
R[i] = LSB(64, B)

3) Sortir le résultat.
Si A est une valeur initiale appropriée (voir au paragraphe 3.4),
Alors
Pour i = 1 à n
P[i] = R[i]
Autrement
Retourner une erreur

3.4 Intégrité des données clé --valeur initiale

La valeur initiale (IV) se réfère à la valeur allouée à A[0] dans la première étape du processus d'enveloppement. Cette valeur est utilisée pour obtenir une vérification d'intégrité sur les données de clé. Dans l'étape finale du processus de désenveloppement, la valeur récupérée de A[0] est comparée à la valeur attendue de A[0]. Si elles correspondent, la clé est acceptée comme valide, et l'algorithme de désenveloppement la retourne. Si elles ne correspondent pas, la clé est alors rejetée, et l'algorithme de désenveloppement retourne une erreur.

Les propriétés exactes réalisées par cette vérification d'intégrité dépendent de la définition de la valeur initiale. Des applications différentes peuvent invoquer des propriété assez différentes ; par exemple, si il est nécessaire de déterminer l'intégrité des données de clé tout au long de son cycle de vie ou juste quand elles sont désenveloppées. La présente spécification définit une valeur initiale par défaut qui prend en charge l'intégrité des données de clé durant la période où elle est enveloppée (au paragraphe 3.4.1). Des dispositions sont aussi fournies pour la prise en charge d'autres valeurs initiales (au paragraphe 3.4.2).

3.4.1 Valeur initiale par défaut

La valeur initiale (IV) par défaut est définie comme étant la constante hexadécimale : A[0] = IV = A6A6A6A6A6A6A6A6

L'utilisation d'une constante comme IV prend en charge une forte vérification d'intégrité sur les données de clé durant la période où elles sont enveloppées. Si le désenveloppement produit A[0] = A6A6A6A6A6A6A6A6, les chances que les données de clé soient corrompues est de 2^-64. Si le désenveloppement produit A[0] = toute autre valeur, le désenveloppement doit alors retourner une erreur et ne pas retourner de données de clé.

3.4.2 Autres valeurs initiales

Lorsque l'enveloppement de clé est utilisé au titre d'un plus grand protocole ou système de gestion de clés, la portée désirée pour l'intégrité des données peut être plus que juste les données de clé, et la durée souhaitée peut être plus que juste la période où elles sont enveloppées. Aussi, si les données de clé ne sont pas juste une clé SEED, elles peuvent n'être pas toujours un multiple de 64 bits. D'autres définitions de la valeur initiale peuvent être utilisées pour traiter de tels problèmes. Selon la [RFC3394], le NIST définira d'autres valeurs initiales dans de futures publications de gestion de clés lorsque elles seront nécessaires. Pour traiter un ensemble de solutions de remplacement qui pourrait évoluer avec le temps, les mises en œuvre d'enveloppement de clé non spécifiques d'une application devront avoir une certaine souplesse dans la façon dont la valeur initiale est établie et vérifiée.

4. Attribut SMIMECapabilities

Un client S/MIME DEVRAIT annoncer l'ensemble de fonctions cryptographiques qu'il prend en charge en utilisant l'attribut de capacités S/MIME. Cet attribut donne une liste partielle des OID des fonctions cryptographiques et DOIT être signé par le client. Les OID des fonctions DEVRAIENT être logiquement séparés en catégories fonctionnelles et DOIVENT être ordonnés selon leur préférence.

Le paragraphe 2.5.2 de la [RFC3851] définit l'attribut SMIMECapabilities signé (défini comme SEQUENCE de SEQUENCES SMIMECapability) à utiliser pour spécifier une liste partielle d'algorithmes que le logiciel qui annonce les SMIMECapabilities peut prendre en charge.

Si un client S/MIME est obligé de prendre en charge le chiffrement symétrique avec SEED, l'attribut de capacités DOIT contenir l'OID SEED spécifié ci-dessus dans la catégorie des algorithmes symétriques. Le paramètre associé à cet OID DOIT être SeedSMimeCapability.

SeedSMimeCapabilty ::= NULL

La SEQUENCE SMIMECapability qui représente SEED DOIT être codée en DER comme la chaîne hexadécimale suivante : 30 0C 06 08 2A 83 1A 8C 9A 44 01 04 05 00

Lorsque un agent envoyeur crée un message chiffré, il doit décider quel type d'algorithme de chiffrement utiliser. En général, le processus de décision implique des informations obtenues des listes de capacités incluses dans les messages reçus du receveur, ainsi que d'autres informations, comme des accords privés, les préférences de l'utilisateur et les restrictions légales. Si la politique locale exige l'utilisation de SEED pour le chiffrement symétrique, les deux client

S/MIME envoyeur et receveur doivent le prendre en charge, et SEED doit être configuré comme algorithme symétrique préféré.

5. Considérations sur la sécurité

Le présent document spécifie l'utilisation de SEED pour chiffrer le contenu d'un message de CMS et pour chiffrer la clé symétrique utilisée pour chiffrer le contenu du message de CMS, les autres mécanismes étant les mêmes que ceux existants. Donc, les considérations sur la sécurité décrites dans les spécifications de CMS, [RFC3852], [RFC3370] et l'algorithme d'enveloppement de clé AES [RFC3394] peuvent être appliquées au présent document. Aucun problème de sécurité n'a été trouvé sur SEED [CRYPTREC].

6. Références

6.1 Références normatives

[TTASSEED] Telecommunications Technology Association (TTA), South Korea, "128-bit Symmetric Block Cipher (SEED)", TTAS.KO- 12.0004, septembre 1998 (en coréen)
http://www.tta.or.kr/English/new/main/index.htm

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

[RFC3370] R. Housley, "Algorithmes de syntaxe de message cryptographique (CMS)", août 2002. *(P.S.)*

[RFC3851] B. Ramsdell, "Spécification du message d'extensions de messagerie Internet multi-objets/sécurisé (S/MIME) version 3.1", juillet 2004. *(Remplacée par* RFC5751)

[RFC3394] J. Schaad, R. Housley, "Algorithme d'enveloppe de clés pour la norme de chiffrement évoluée (AES)", septembre 2002. *(Information)*

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

6.2 Références pour information

[RFC4009] J. Park et autres, "Algorithme de chiffrement SEED", février 2005. (*Obsolète, voir* RFC4269) *(Information)*

[ISOSEED] ISO/CEI JTC1/SC 27 N 256r1, "National Body contributions on NP 18033 Encryption algorithms in response to document SC 27 N 2563", octobre 2000.

[CRYPTREC] Information-technology Promotion Agency (IPA), Japan, CRYPTREC. "SEED Evaluation Report", février 2002. http://www.kisa.or.kr

Appendice A. Modules ASN.1

SeedEncryptionAlgorithmInCMS
{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9) smime(16) modules(0) id-mod-cms-seed(25) }

ÉTIQUETTES IMPLICITES DE DÉFINITIONS ::=

DÉBUT

IDENTIFIANT D'OBJET id-seedCBC ::= { iso(1) member-body(2) korea(410) kisa(200004) algorithm(1) seedCBC(4) }

-- Valeur d'initialisation (IV)

```
   SeedCBCParameter ::= SeedIV
   SeedIV ::= CHAÎNE D'OCTETS (TAILLE(16))

-- Identifiants d'algorithme d'enveloppement de clé SEED - Parameter est absent.

IDENTIFIANT D'OBJET id-npki-app-cmsSeed-wrap ::=
       { iso(1) member-body(2) korea(410) kisa(200004) npki-app(7) smime(1) alg(1) cmsSEED-wrap(1) }

-- Paramètre de capacité S/MIME SEED

 SeedSMimeCapability ::= NULL

FIN
```

Adresse des auteurs

Jongwook Park
Korea Information Security Agency
78, Garak-Dong, Songpa-Gu, Seoul, 138-803
REPUBLIC OF KOREA
téléphone : +82-2-405-5432
mél : khopri@kisa.or.kr

Sungjae Lee
Korea Information Security Agency
REPUBLIC OF KOREA

téléphone : +82-2-405-5243
Fax : +82-2-405-5499
mél : sjlee@kisa.or.kr

Jeeyeon Kim
Korea Information Security Agency
REPUBLIC OF KOREA
téléphone : +82-2-405-5238
Fax : +82-2-405-5499
mél : jykim@kisa.or.kr

Jaeil Lee
Korea Information Security Agency
téléphone : +82-2-405-5300
Fax : +82-2-405-5499
mél : jilee@kisa.or.kr

Déclaration complète de droits de reproduction

norme. Prière d'adresser les informations à l'IETF à ietf- ipr@ietf.org.

Remerciement

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