Groupe de travail Réseau
Request for Comments: 5649
Catégorie : Information
Traduction Claude Brière de L'Isle

R. Housley, Vigil Security
M. Dworkin, NIST
août 2009

Algorithme d'enveloppement de clé avec bourrage pour la norme de chiffrement évolué (AES)

Résumé
Le présent document spécifie une convention de bourrage à utiliser avec l'algorithme d'enveloppement de clé AES spécifié dans la RFC 3394. Cette convention élimine l'exigence que la longueur de la clé à envelopper soit un multiple de 64 bits, permettant d'envelopper une clé de n'importe quelle longueur pratique.

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

Table des matières

1. Introduction

La gestion des clés de chiffrement conduit souvent à des situations où une clé symétrique est utilisée pour chiffrer et protéger l'intégrité d'une autre clé, qui peut être soit une clé symétrique soit une clé asymétrique. L'opération est souvent appelée un enveloppement de clé.

Le présent document spécifie une extension de l'algorithme d'enveloppement de clé de la norme de chiffrement évoluée (AES, *Advanced Encryption Standard*) [AES-KW1], [RFC3394]. Sans cette extension, l'entrée à l'algorithme d'enveloppement de clé AES, appelée les données de clé, doit être une séquence de deux ou plusieurs blocs de 64 bits.

L'algorithme d'enveloppement de clé AES avec bourrage peut être utilisé pour envelopper une clé de toute taille pratique avec une clé AES. La clé de chiffrement de clé (KEK, *key-encryption key*) AES doit être de 128, 192, ou 256 bits. Les données de clé d'entrée peuvent ne faire pas plus d'un octet, ce qui va résulter en une sortie de deux blocs de 64 bits (ou 16 octets). Bien que l'algorithme d'enveloppement de clé AES ne mette pas de limite supérieure à la taille des données de clé qui peuvent être enveloppées, cette extension en met. L'utilisation d'un champ fixe de 32 bits pour porter la longueur d'octets des données de clé limite la taille de l'entrée à 2^32 octets. La plupart des systèmes vont avoir d'autres facteurs qui limitent la taille pratique des données de clé à beaucoup moins que 2^32 octets.

Un indicateur de longueur de message (MLI, *Message Length Indicator*) est défini au titre d'une "valeur initiale de remplacement" en accord avec la déclaration du paragraphe 2.2.3.2 de [AES-KW1], qui dit :"aussi, si les données de clé ne sont pas juste une clé AES, elles peuvent ne pas toujours être un multiple de 64 bits. D'autres définitions de la valeur initiale peuvent être utilisées pour traiter de tels problèmes."

2. Notation et définitions

La notation suivante est utilisée dans les descriptions de l'algorithme :

MSB(j, W) : retourne les j bits de poids fort de W
LSB(j, W) : retourne les j bits de moindre poids de W
ENC(K, B) : AES chiffre le bloc B de 128 bits en utilisant la clé K
DEC(K, B) : AES déchiffre le bloc B de 128 bits en utilisant la clé K
V1 | V2 : enchaînement de V1 et V2
K : clé de chiffrement de clé
m : nombre d'octets dans les données de clé
n : nombre de blocs de 64 bits dans les données de clé avec bourrage
Q[i] : ième octet de texte en clair dans les données de clé
P[i] : ième bloc de 64 bits de texte en clair dans les données de clé avec bourrage
C[i] : ième bloc de 64 bits de bloc de données de texte chiffré
A : registre de vérification d'intégrité de 64 bits

3. Valeur initiale de remplacement

La valeur initiale de remplacement (AIV, *Alternative Initial Value*) exigée par la présente spécification est une constante de 32 bits enchaînée à un MLI de 32 bits. La constante est (en hexadécimal) A65959A6 et occupe la moitié de poids fort de l'AIV. Noter que cela diffère des 32 bits de poids fort de l'IV par défaut du paragraphe 2.2.3.1 de [AES-KW1], de sorte qu'il n'y a pas d'ambiguïté entre les deux. Le MLI de 32 bits, qui occupe la moitié de moindre poids de l'AIV, est un entier binaire non signé égal à la longueur d'octets des données de clé en clair, dans l'ordre du réseau -- c'est-à-dire, avec l'octet de poids fort en premier. Quand le MLI n'est pas un multiple de 8, les données de clé sont bourrées sur la droite avec le moindre nombre d'octets suffisant pour faire de la longueur d'octets résultante un multiple de 8. La valeur de chaque octet de bourrage devra être 0 (huit zéros binaires).

Noter que pour un nombre donné de blocs de 64 bits de texte en clair, il y a seulement huit valeurs de MLI qui peuvent avoir ce résultat. Par exemple, les seules valeurs de MLI qui sont valides avec quatre blocs de 64 bits de texte en clair sont 32 (sans octets de bourrage) 31 (avec un octet de bourrage) 30, 29, 28, 27, 26, et 25 (avec sept octets de bourrage). Quand le processus de désenveloppement spécifié ci-dessous donne n blocs de 64 bits de données de résultat et une AIV, les huit valeurs valides pour le MLI sont 8*n, (8*n)-1, ..., et (8*n)-7. Donc, la vérification d'intégrité de l'AIV, qui est contenue dans un registre de 64 bits appelé A, exige les étapes suivantes :

1) vérifier que MSB(32,A) = A65959A6.

2) vérifier que 8*(n-1) < LSB(32,A) ≤ n. Si il en est ainsi, soit MLI = LSB(32,A).

3) Soit b = (8*n)-MLI, et ensuite, vérifier que les b octets de droite des données de résultat sont zéro.

Si les trois vérifications réussissent, alors l'AIV est valide. Si une des vérifications échoue, l'AIV est invalide et l'opération de désenveloppement doit retourner une erreur.

4. Spécification de l'enveloppement de clé AES avec algorithme de bourrage

L'enveloppement de clé AES avec algorithme de bourrage consiste en un processus d'enveloppement et un processus de désenveloppement, tous deux fondés sur le livre de code AES [AES]. Il fournit une extension à l'algorithme d'enveloppement de clé AES [AES-KW1], [RFC3394] qui élimine l'exigence que la longueur de la clé à envelopper soit un multiple de 64 bits. Les deux paragraphes qui suivent spécifient les processus d'enveloppement et de désenveloppement, appelés, respectivement, le processus étendu d'enveloppement de clé et le processus étendu de désenveloppement de clé. Ces noms distinguent ces processus de ceux spécifiés dans [AES-KW1] et la [RFC3394].

4.1 Processus étendu d'enveloppement de clé

Les entrées au processus étendu d'enveloppement de clé sont la KEK et le texte en clair à envelopper. Le texte en clair consiste en entre un et 2^32 octets, contenant les données de clé à envelopper. Le processus d'enveloppement de clé est décrit ci-dessous.

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

1) Ajout du bourrage
Si m n'est pas un multiple de 8, bourrer la chaîne d'octets du texte en clair à droite avec des octets {Q[m+1], ..., Q[r]} de zéros, où r est le plus petit multiple de 8 qui est supérieur à m. Si m est un multiple de 8, il n'y a pas de bourrage, et r = m.

Régler n = r/8, qui est la même chose que Plafond(m/8).

Pour i = 1, ..., n
    j = 8*(i-1)
    P[i] = Q[j+1] | Q[j+2] | ... | Q[j+8].

2) Enveloppement
Si le texte en clair bourré contient exactement huit octets, alors ajouter devant l'AIV comme défini à la Section 3 devant P[1] et chiffrer le bloc de 128 bits résultant en utilisant AES en mode ECB [Modes] avec la clé K (la KEK). Dans ce cas, le résultat est deux blocs de 64 bits C[0] et C[1] :

    C[0] | C[1] = ENC(K, A | P[1]).

Autrement, appliquer le processus d'enveloppement spécifié au paragraphe 2.2.1 de la [RFC3394] au texte en clair bourré {P[1], ..., P[n]} avec K (la KEK) et l'AIV comme défini à la Section 3 ci-dessus comme valeur initiale. Le résultat est n+1 blocs de 64 bits {C[0], C[1], ..., C[n]}.

4.2 Processus étendu de désenveloppement de clé

Les entrées au processus étendu de désenveloppement de clé sont la KEK et (n+1) blocs de 64 bits de texte chiffré consistant en une clé précédemment enveloppée. Si le texte chiffré est une clé enveloppée de façon valide, le processus de désenveloppement retourne alors des blocs de 64 bits de texte en clair avec bourrage, qui sont alors transposés dans cette extension en m octets de données de clé déchiffrée, comme indiqué par le MLI incorporé dans l'AIV.

Entrées : texte chiffré, (n+1) blocs de 64 bits {C[0], C[1], ..., C[n]}, et la clé, K (la KEK).
Résultats : texte en clair, m octets {Q[1], Q[2], ..., Q[m]}, ou une erreur.

1) Désenveloppement de clé
Quand n est un (n=1) le texte chiffré contient exactement deux blocs de 64 bits (C[0] et C[1]) et ils sont déchiffrés comme un seul bloc AES en utilisant AES en mode ECB [Modes] avec K (la KEK) pour récupérer l'AIV et la clé en texte en clair avec bourrage :

    A | P[1] = DEC(K, C[0] | C[1]).

Autrement, on applique les étapes 1 et 2 du processus de désenveloppement spécifié au paragraphe 2.2.2 de la [RFC3394] aux n+1 blocs de texte chiffré de 64 bits, {C[0], C[1], ..., C[n]}, et à la KEK, K. Définir les blocs de texte en clair avec bourrage , {P[1], ..., P[n]}, comme spécifié à l'étape 3 de ce processus, avec A[0] comme valeur de A. Noter que vérifier "si

A[0] est une valeur appropriée" est légèrement retardé à l'étape 2 ci-dessous car le texte en clair avec bourrage est nécessaire pour effectuer cette vérification quand l'AIV est utilisée.

2) Vérification de l'AIV
Effectuer les trois vérifications décrites à la Section 3 sur le texte en clair avec bourrage et la valeur A. Si une des vérifications échoue, retourner une erreur.

3) Suppression du bourrage
Soit m = la valeur de MLI extraite de A.
Soit P = P[1] | P[2] | ... | P[n].

Pour i = 1, ... , m
    Q[i] = LSB(8, MSB(8*i, P))

5. Identifiants d'algorithme

Certains protocoles de sécurité emploient l'ASN.1 [X.680] et emploient des identifiants d'algorithme pour désigner les algorithmes de chiffrement. Pour prendre en charge ces protocoles, l'algorithme d'enveloppement de clé AES avec bourrage a reçu les identifiants d'algorithme suivants, un pour chaque taille de KEK AES. Les identifiants d'algorithme d'enveloppement de clé AES (sans bourrage) sont aussi inclus ici pour être complet.

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

IDENTIFIANT D'OBJET id-aes128-wrap ::= { aes 5 }
IDENTIFIANT D'OBJET id-aes128-wrap-pad ::= { aes 8 }

IDENTIFIANT D'OBJET id-aes192-wrap ::= { aes 25 }
IDENTIFIANT D'OBJET id-aes192-wrap-pad ::= { aes 28 }

IDENTIFIANT D'OBJET id-aes256-wrap ::= { aes 45 }
IDENTIFIANT D'OBJET id-aes256-wrap-pad ::= { aes 48 }

Dans tous les cas, le champ de paramètre AlgorithmIdentifier DOIT être absent.

6. Exemples d'enveloppement de clé bourré

Les exemples de cette section ont été générés en utilisant la mise en œuvre fondée sur l'indice de l'algorithme d'enveloppement de clé AES avec l'approche de bourrage spécifiée à la Section 4 du présent document. Toutes les valeurs sont montrées en hexadécimal.

Le premier exemple enveloppe 20 octets de données de clé avec une KEK de 192 bits.

KEK : 5840df6e29b02af1 ab493b705bf16ea1 ae8338f4dcc176a8
Clé : c37b7e6492584340 bed1220780894115 5068f738
Enveloppe : 138bdeaa9b8fa7fc 61f97742e72248ee 5ae6ae5360d1ae6a5f54f373fa543b6a

Le second exemple enveloppe 7 octets de données de clé avec une KEK de 192 bits.

KEK : 5840df6e29b02af1 ab493b705bf16ea1 ae8338f4dcc176a8
Clé : 466f7250617369
Enveloppe : afbeb0f07dfbf541 9200f2ccb50bb24f

7. Considérations sur la sécurité

Les mises en œuvre doivent protéger la clé de chiffrement de clé (KEK). La compromission de la KEK peut aboutir à la divulgation de toutes les clés qui ont été enveloppées avec la KEK, ce qui peut conduire à la compromission de tout le trafic protégé avec ces clés enveloppées.

La KEK doit être au moins aussi bonne que le matériel de chiffrement qu'elle protège.

Si la KEK et la clé enveloppée sont associées à des algorithmes de chiffrement différents, la sécurité effective fournie aux données protégées avec la clé enveloppée est déterminée par le plus faible des deux algorithmes. Si, par exemple, les données sont chiffrées avec AES à 128 bits et cette clé AES est enveloppée avec une clé AES de 256 bits, au plus 128 bits de protection sont alors fournis aux données. Si, dans un autre exemple, une clé AES de 128 bits est utilisée pour envelopper une clé privée RSA de 4096 bits, au plus une protection de 128 bits est alors fournie à toutes les données qui dépendent de cette clé privée. Donc, les mises en œuvre doivent s'assurer que les algorithmes de chiffrement de clé sont au moins aussi forts que les autres algorithmes de chiffrement employés dans l'ensemble d'un système.

Les algorithmes d'enveloppe de clé AES et d'enveloppe de clé AES avec bourrage utilisent des constantes différentes dans la valeur initiale. L'utilisation de valeurs différentes assure que le receveur de données de clé avec bourrage ne peut pas réussir à les désenvelopper comme des données de clé sans bourrage, ou vice versa. Cela reste vrai quand les données de clé sont enveloppées en utilisant l'algorithme d'enveloppe de clé AES avec bourrage mais qu'aucun bourrage n'est nécessaire.

L'algorithme d'enveloppe de clé AES avec bourrage fournit presque la même quantité de protection de l'intégrité que l'algorithme d'enveloppe de clé AES.

Une technique de bourrage précédente était spécifiée pour envelopper les clés de code d'authentification de message haché (HMAC, *Hashed Message Authentication Code*) avec AES [RFC3537]. La technique du présent document est plus générale ; elle n'est pas limitée à l'enveloppement des clés HMAC.

Dans la conception de certains modules de chiffrement de haute sécurité, il est souhaitable de séparer le matériel de clés de chiffrement des autres données. L'utilisation d'un mécanisme de chiffrement spécifique seulement pour la protection du matériel de chiffrement de clé peut aider à atteindre ce but. L'enveloppement de clé AES et l'enveloppement de clé AES avec bourrage sont de tels mécanismes. Les concepteurs de systèmes ne devraient pas utiliser ces algorithmes pour chiffrer autres chose que du matériel de chiffrement de clés.

8. Références

8.1 Références normatives

[AES] National Institute of Standards and Technology, FIPS Pub 197, "Advanced Encryption Standard (AES)", 26 novembre 2001.

[AES-KW1] National Institute of Standards and Technology, "AES Key Wrap Specification", 17 novembre 2001. http://csrc.nist.gov/groups/ST/toolkit/documents/kms/AES_key_wrap.pdf

[Modes] Dworkin, M., "Recommendation for Block Cipher Modes of Operation -- Methods and Techniques", NIST Special Publication 800-38A, 2001.

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

[X.680] Recommandation UIT-T X.680 | ISO/CEI 8824-1:2002. "Technologie de l'information - Notation de syntaxe abstraite numéro un (ASN.1) : spécification de la notation de base".<https://www.itu.int/rec/T-REC-X.680>

8.2 Références pour information

[RFC3537] J. Schaad, R. Housley, "Enveloppement d'une clé de code d'authentification de message haché (HMAC) dans

une clé de la norme de triple chiffrement de données (3DES) ou de la norme de chiffrement évolué (AES)", mai 2003. *(P.S.)*

[RFC3565] J. Schaad, "Utilisation de l'algorithme de chiffrement de la norme de chiffrement évolué (AES) dans la syntaxe de message cryptographique (CMS)", juillet 2003. *(P.S.)*

[RFC5911] P. Hoffman, J. Schaad, "Nouveaux modules ASN.1 pour la syntaxe de message cryptographique (CMS) et S/MIME", juin 2010. *(Information)*

[X.681] Recommandation UIT-T X.681 | ISO/CEI 8824-2:2015, "Technologie de l'information - Notation de syntaxe abstraite numéro un (ASN.1) : spécification de l'objet d'information", août 2015, <https://www.itu.int/rec/T-REC-X.681>.

[X.682] Recommandation UIT-T X.682 | ISO/CEI 8824-3:2015, "Technologie de l'information - Notation de syntaxe abstraite numéro un (ASN.1) : spécification des contraintes", août 2015, <https://www.itu.int/rec/T-REC-X.682>.

[X.683] Recommandation UIT-T X.683 | ISO/CEI 8824-4:2015, "Technologie de l'information - Notation de syntaxe abstraite numéro un (ASN.1) : paramètres des spécifications d'ASN.1", août 2015, <https://www.itu.int/rec/T-REC-X.683>.

9. Remerciements

C'est à Paul Timmel qu'on doit le MLI et la technique de bourrage décrite dans ce document.

Appendice A. Modules ASN.1

Cet appendice inclut deux modules ASN.1. Le premier utilise la syntaxe de 1988, et le second utilise la syntaxe de 2002.

L'appendice A.1 donne les définitions normatives d'ASN.1 pour les identifiants d'algorithme inclus dans la présente spécification qui utilisent l'ASN.1 comme défini dans [X.680] avec la syntaxe 1988 d'ASN.1.

L'appendice A.2 donne les définitions normatives d'ASN.1 pour les identifiants d'algorithme inclus dans la présente spécification qui utilisent l'ASN.1 comme défini dans [X.680], [X.681], [X.682], et [X.683] avec la syntaxe ASN1 de 2002. Cet appendice contient les mêmes informations que l'appendice A.1 ; cependant, l'appendice A.1 a la préséance en cas de conflit. Les objets d'algorithme de chiffrement de contenu et d'enveloppe de clé sont définis dans la [RFC5911].

Les identifiants d'algorithme id-aes128-wrap, id-aes192-wrap, et id-aes256-wrap sont définis dans la [RFC3565].

A.1 Module ASN.1 1988

```
AESKeyWrapWithPad-88 { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) modules(0) 47 }

ÉTIQUETTES IMPLICITES DE DÉFINITIONS ::=

DÉBUT

-- EXPORTE TOUT --

-- IMPORTE RIEN --

-- Identifiants d'objet d'information AES --

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

-- Identifiants d'objet d'algorithme d'enveloppement de clé AES avec bourrage à utiliser sans le champ Paramètre
IDENTIFIANT D'OBJET id-aes128-wrap-pad ::= { aes 8 }
```

```
IDENTIFIANT D'OBJET id-aes192-wrap-pad ::= { aes 28 }
IDENTIFIANT D'OBJET id-aes256-wrap-pad ::= { aes 48 }

FIN
```

A.2 Module ASN.1 2002

```
AESKeyWrapWithPad-02 { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) modules(0) 48 }

ÉTIQUETTES IMPLICITES DE DÉFINITIONS ::=

DÉBUT

-- EXPORTE TOUT --

IMPORTE
    AlgorithmIdentifier{}, CONTENT-ENCRYPTION, KEY-WRAP, SMIME-CAPS
    FROM AlgorithmInformation-2009  -- [RFC5911]
      { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
        id-mod-algorithmInformation-02(58) };

  AES-ContentEncryption CONTENT-ENCRYPTION ::= {
    cea-aes128-wrap-pad |
    cea-aes192-wrap-pad |
    cea-aes256-wrap-pad,
    ... }

  AES-KeyWrap KEY-WRAP ::= {
    kwa-aes128-wrap-pad |
    kwa-aes192-wrap-pad |
    kwa-aes256-wrap-pad,
    ... }

  SMimeCaps SMIME-CAPS ::= {
    cea-aes128-wrap-pad.&smimeCaps |
    cea-aes192-wrap-pad.&smimeCaps |
    cea-aes256-wrap-pad.&smimeCaps |
    kwa-aes128-wrap-pad.&smimeCaps |
    kwa-aes192-wrap-pad.&smimeCaps |
    kwa-aes256-wrap-pad.&smimeCaps,
    ... }

-- Identifiant d'objet AES

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

-- Algorithmes de chiffrement de contenu

cea-aes128-wrap-pad CONTENT-ENCRYPTION ::= {
    IDENTIFIER id-aes128-wrap-pad
    PARAMS ARE absent
    SMIME-CAPS { IDENTIFIED BY id-aes128-wrap-pad } }

cea-aes192-wrap-pad CONTENT-ENCRYPTION ::= {
    IDENTIFIER id-aes192-wrap-pad
    PARAMS ARE absent
    SMIME-CAPS { IDENTIFIED BY id-aes192-wrap-pad } }

cea-aes256-wrap-pad CONTENT-ENCRYPTION ::= {
```

```
    IDENTIFIER id-aes256-wrap-pad
    PARAMS ARE absent
    SMIME-CAPS { IDENTIFIED BY id-aes256-wrap-pad } }

-- Algorithmes d'enveloppe de clé

kwa-aes128-wrap-pad KEY-WRAP ::= {
    IDENTIFIER id-aes128-wrap-pad
    PARAMS ARE absent
    SMIME-CAPS { IDENTIFIED BY id-aes128-wrap-pad } }

IDENTIFIANT D'OBJET id-aes128-wrap-pad ::= { aes 8 }

kwa-aes192-wrap-pad KEY-WRAP ::= {
    IDENTIFIER id-aes192-wrap-pad
    PARAMS ARE absent
    SMIME-CAPS { IDENTIFIED BY id-aes192-wrap-pad } }

IDENTIFIANT D'OBJET  id-aes192-wrap-pad ::= { aes 28 }

kwa-aes256-wrap-pad KEY-WRAP ::= {
    IDENTIFIER id-aes256-wrap-pad
    PARAMS ARE absent
    SMIME-CAPS { IDENTIFIED BY id-aes256-wrap-pad } }

IDENTIFIANT D'OBJET id-aes256-wrap-pad ::= { aes 48 }

FIN
```

Adresse des auteurs

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

Morris Dworkin
National Institute of Standards and Technology
100 Bureau Drive, Mail Stop 8930
Gaithersburg, MD 20899-8930
USA
mél : dworkin@nist.gov