Groupe de travail Réseau
Request for Comments : 3185
Catégorie : En cours de normalisation
Traduction Claude Brière de L'Isle

S. Farrell, Baltimore Technologies
S. Turner, IECA
octobre 2001

Réutilisation des clés de chiffrement de contenu dans la CMS

Résumé
Le présent document décrit une façon d'inclure un identifiant de clé dans une structure de données enveloppées de CMS (Syntaxe de message cryptographique) afin que la clé de chiffrement de contenu puisse être réutilisée pour d'autres paquets de données enveloppées.

Table des matières

1. Introduction

La CMS [RFC2630] spécifie EnvelopedData. EnvelopedData prend en charge le chiffrement de données en utilisant des techniques de gestion de clé symétriques ou asymétriques. Comme l'établissement de clés asymétriques est relativement coûteux, il est souhaitable dans certains environnements de réutiliser une clé partagée de chiffrement de contenu en utilisant des mécanismes asymétriques pour les opérations de chiffrement dans les messages suivants.

L'idée de base est ici de réutiliser la clé de chiffrement de contenu (CEK, *content-encryption key*) provenant d'un message (disons MSG1) pour déduire la clé de chiffrement de clé (KEK, *key-encryption key*) pour un message ultérieur (MSG2) en incluant une valeur de référence pour la CEK dans le message 1, et la même valeur que le KEKIdentifier *(identifiant de clé de chiffrement de clé)* pour le message 2. La CEK provenant du message 1 est appelée la "CEK référencée".

Les mots clés "DOIT", "EXIGE", "DEVRAIT", "RECOMMANDE", et "PEUT" dans ce document sont à interpréter comme décrit dans la [RFC2119].

2. Applicabilité

La présente spécification est destinée à être utilisée pour fournir une confidentialité plus efficace du choix des champs entre les homologues communicants, en particulier dans les cas où :

- L'origine est un client qui souhaite envoyer un certain nombre de champs à un serveur (le receveur) dans une seule transaction, où la CEK référencée est utilisée pour le chiffrement séparé de chaque champ.

- L'origine et le receveur sont des serveurs qui communiquent très fréquemment et utilisent cette spécification pour de simples raisons d'efficacité.

La présente spécification n'est pas destinée à s'appliquer dans tous les cas. Elle convient lorsque :

- Son utilisation est élargie ; c'est-à-dire que cette spécification ne définit pas un protocole mais simplement une astuce qui peut être utilisée dans un contexte plus large et une spécification supplémentaire sera nécessaire pour chacun de ces cas. En particulier, afin d'utiliser cette spécification, il est EXIGÉ de définir le comportement de l'origine et des receveurs lorsque une CEK référencée a été "perdue".

- Cette spécification ne convient pas pour la gestion de clé de groupe générale.

- L'API de chiffrement sous-jacente convient : il est très probable que toute API de chiffrement qui "cache" complètement les bits des clés de chiffrement à la couche de CMS va empêcher la réutilisation d'une CEK référencée (parce qu'elle n'aura pas de primitive qui permette la transformation de MSG1.CEK en MSG2.KEK).

- Les algorithmes de chiffrement de contenu et de clé ont des valeurs et des forces de clé compatibles, c'est-à-dire que si MSG1.contentEncryptionAlgorithm est un chiffrement de 40 bits et si MSG2.keyEncryptionAlgorithm exige 168 bits de matériel de clé, la présente spécification NE DEVRAIT PAS être utilisée.

On peut envisager d'autres façons d'établir le matériel de clés symétriques requis, par exemple, en établissant un schéma de clés de groupe ou en définissant un type de contenu qui contient une valeur de KEK. Bien que ce schéma soit beaucoup plus simple que la gestion de clé de groupe générique, si une mise en œuvre prend déjà en charge la gestion de clé de groupe, ce schéma n'apporte rien. Ce schéma convient aussi pour l'inclusion dans des bibliothèques de CMS (bien que l'ajout d'un nouvel état puisse poser un problème à certaines mises en œuvre) ce qui peut présenter certains avantages sur les schémas de couche application (par exemple, lorsque le contenu comporte MSG2.KEK).

3. Comment le faire

Afin de référencer la clé de chiffrement de contenu (CEK, *content-encryption key*) utilisée dans une EnvelopedData, un identifiant de clé peut être inclus dans le champ unprotectedAttrs de MSG1. Cette clé peut alors être utilisée pour déduire la clé de chiffrement de clé (KEK, *key-encryption key*) pour d'autres instances de EnvelopedData ou pour d'autres objets. Si la CEK provenant de MSG1 doit être utilisée pour déduire la KEK pour MSG2, alors MSG1 DOIT contenir un attribut unprotectedAttrs du type id-aa-CEKReference avec une seule valeur qui utilisé la syntaxe CEKReference.

MSG2.KEK est à déduire en inversant les octets de MSG1.CEK. L'inversion d'octets est destinée à éviter une attaque dans laquelle l'agresseur connaît un texte en clair et le chiffrement qui s'y rapporte (chiffré avec MSG1.CEK) qui (autrement) pourrait être directement utilisé comme une MSG2.KEK.

L'application DOIT s'assurer que les algorithmes pertinents sont compatibles. C'est-à-dire qu'un attribut CEKReference seul ne peut être utilisé que lorsque l'algorithme de chiffrement de contenu provenant de MSG1 emploie le même type de clé symétrique que l'algorithme de chiffrement de clé provenant de MSG2.

Notes :
1) Il n'y a rien pour empêcher l'inclusion d'un attribut CEKReference dans MSG2 tout comme dans MSG1. C'est-à-dire que l'origine peut "faire rouler" la CEK référencée dans tous les messages.
2) L'attribut CEKReference peut survenir avec tous les choix de RecipientInfo : ktri, kari ou kekri. Les mises en œuvre NE DOIVENT PAS supposer que CEKReference ne peut survenir que lorsque ktri ou kari sont utilisés.

IDENTIFIANT D'OBJET id-aa-CEKReference ::= { id-aa 30 } CEKReference ::= CHAINE D'OCTETS

id-aa est un identifiant d'objet défini dans la [RFC2633].

Pour permettre à l'origine de MSG1 d'indiquer la "durée de vie" de la CEK, elle PEUT aussi inclure un attribut CEKMaxDecrypts, dans le champ unprotectedAttrs de EnvelopedData. Cet attribut a une syntaxe de ENTIER (sa valeur DOIT être ≥ 1 et au plus de 2^31) et il indique au receveur le nombre maximum de messages (MSG1 exclu) qui vont utiliser la CEK référencée. Cet attribut DOIT être envoyé dans les seuls cas où un attribut CEKReference est aussi inclus.

Le receveur DEVRAIT conserver les informations de CEKReference (au minimum, l'identifiant de clé et la valeur de CEK) tant que moins de maxDecrypt messages ont été bien reçus. Les receveurs DEVRAIENT supprimer les informations de CEKReference après un délai configuré en local.

Lorsque cet attribut n'est pas présent, les origines et les receveurs DEVRAIENT se comporter comme si une valeur de un avait été envoyée.

IDENTIFIANT D'OBJET id-aa-CEKMaxDecrypts ::= { id-aa 31 } CEKMaxDecrypts ::= ENTIER

Si CEKMaxDecrypts est manquant, ou a la valeur de un, chaque CEK va alors être réutilisée une fois comme KEK pour le prochain message. Cela signifie que MSG[n].KEK est la MSG[n-1].CEK à octets inversés ; les MSG[n+1].KEK suivantes seront la MSG[n].CEK à octets inversés. Noter que MSG[n-1].CEK n'a pas d'impact sur un quelconque MSG[n+1], tant que les CEK sont générées de façon aléatoire (et non, par exemple, déduites d'une façon ou d'une autre des KEK).

4. Utiliser des algorithmes de CEK et de KEK différents

Lorsque l'algorithme de chiffrement de contenu de MSG1 et l'algorithme de chiffrement de clé de MSG2 sont le même algorithme, la KEK de MSG2 est les octets inverses de la CEK de MSG1. Cependant, en général, ces algorithmes PEUVENT différer, par exemple, en exigeant des longueurs de clé différentes. La présente section spécifie une manière générique de déduire la KEK de MSG2 dans de tels cas.

Note : Dans un certain sens, les algorithmes de CEK et de KEK ne sont jamais les "mêmes", par exemple, id-alg-CMS3DESwrap et des-ede3-cbc diffèrent. Cependant, pour les besoins de la présente spécification, tout ce dont on se soucie est que les algorithmes utilisent le même format et la même taille de matériel de clé (voir aussi les considérations pour la sécurité) et qu'ils ne diffèrent pas de façon significative en termes de "force" du résultat cryptographique. Dans ce sens, les deux algorithmes de l'exemple ci-dessus sont les "mêmes".
Les mises en œuvre PEUVENT inclure cette fonctionnalité.

L'approche de base est d'utiliser la fonction de déduction de clé PBKDF2 définie dans PKCS#5 [RFC2898], mais d'utiliser MSG1.CEK comme entrée au lieu d'un mot de passe. Le résultat de la fonction PBKDF2 est MSG2.KEK. À cette fin, un nouveau type d'attribut est défini qui permet de passer les paramètres requis.

```
IDENTIFIANT D'OBJET id-aa-KEKDerivationAlg ::= { id-aa 32 }
  KEKDerivationAlgorithm ::= SEQUENCE {
      kekAlg              AlgorithmIdentifier,
      pbkdf2Param         PBKDF2-params
  }
```

kekAlg est l'identifiant d'algorithme (et des paramètres associés, si il en est) pour l'algorithme MSG2 de chiffrement de clé. Noter qu'il n'est pas nécessaire de protéger ce champ car une modification de keyAlg ne représente qu'une attaque de déni de service.

Les paramètres de l'algorithme PBKDF2 sont à traiter comme suit :
- Le sel DOIT utiliser l'élément "spécifié" du CHOIX.
- L'origine du message choisit le compte d'itération.
- La valeur de keyLength *(longueur de clé)* est déterminée par le kekAlg et DOIT être présente.
- Le champ prf DOIT utiliser l'algorithme par défaut spécifié dans la [RFC2898] qui est algid-hmacWithSHA1 (et donc le champ prf DOIT être omis).

5. Conformité

La présente spécification ne s'applique qu'aux messages dans lesquels l'attribut CEKReference est présent. Tous les attributs spécifiés ici DEVRAIENT être ignorés sauf si ils sont présents dans un message contenant une valeur existante valide, nouvelle ou reconnue, de CEKReference. L'attribut CEKMaxDecrypts est à traiter, par le receveur, comme une indication, mais DOIT être respecté par l'origine.

L'attribut de mise en œuvre facultative KEKDerivationAlgorithm *(algorithme de déduction de KEK)* DOIT être présent dans le seul cas où l'algorithme MSG1.content-encryption est différent de l'algorithme MSG2.key-encryption, auquel cas il DOIT être présent. Les mises en œuvre qui reconnaissent cet attribut, mais qui ne prennent pas en charge la fonctionnalité DEVRAIENT ignorer l'attribut.

Ignorer des attributs comme exposé ci-dessus va conduire à des échecs de déchiffrement. Les mises en œuvre de CMS DEVRAIENT être capables de signaler la raison particulière de cet échec à l'application appelante.

6. Considérations pour la sécurité

Le chiffrement n'assure pas l'authentification, par exemple, si le contenu chiffré est essentiellement aléatoire, les receveurs NE DOIVENT PAS supposer que "connaître" une valeur de CEK référencée prouve quelque chose – n'importe qui a pu créer les données enveloppées. Ceci est pertinent aussi bien pour la sécurité (le texte en clair récupéré ne devrait pas être entièrement aléatoire) et pour éviter les dénis de service (le receveur NE DOIT PAS supposer que d'utiliser la bonne CEKReference signifie que l'origine du message est authentique).

De même, utiliser la CEKReference correcte ne signifie pas qu'un message n'a pas été répété ou inséré, et les receveurs NE DOIVENT PAS supposer que la répétition a été évitée.

Le champ maxDecrypts présente un risque potentiel d'attaque de déni de service si une valeur très grande est incluse par une source qui tente de faire que le receveur consomme de la mémoire en accumulant des CEK référencées pendant une longue période ou si l'origine n'envoie jamais le nombre indiqué de textes chiffrés. Les receveurs DEVRAIENT donc éliminer les CEK référencées lorsque la valeur de maxDecrypts est trop grande (selon la configuration locale) ou lorsque la CEK référencée a été détenue pendant une trop longue période.

Supposons que MSG1 soit envoyé à un ensemble S1 d'usagers. Dans le cas où MSG2 est seulement envoyé à un sous-ensemble d'usagers dans S1, tous les usagers de S1 vont encore être capables de déchiffrer MSG2 (car MSG2.KEK n'est calculée qu'à partir de MSG1.CEK). Les mises en œuvre devraient être conscientes que dans de tels cas, tous les membres de l'ensemble original de receveurs (S1) peuvent accéder au texte en clair de MSG2 et des messages suivants.

La raison de l'inversion des octets est la suivante : sans l'inversion des octets, un attaquant qui connaît certains des textes en clair de MSG1 (par exemple, un préfixe dans un champ) peut utiliser le bloc de texte chiffré correspondant comme prochaine CEK chiffrée, c'est-à-dire, comme MSG2.KEKRecipientInfo.encryptedKey. Maintenant, l'attaquant connaît la prochaine CEK. Cela attaque quelque chose que la présente note ne prétend pas protéger (c'est l'authentification de l'origine) mais c'est facilement évité en utilisant l'inversion des octets. L'inversion d'octet a été choisie plutôt que l'inversion des bits car celle-ci serait cause que les bits de parité de MSG1.CEK seraient utilisés comme bits de clé pour MSG2.KEK pour les algorithmes fondés sur DES. Noter que l'inversion des octets affecterait de façon similaire la parité si la vérification de parité s'étendait sur plus d'un octet, cependant, il n'y a pas d'algorithme bien connu qui fonctionne de cette façon.

Les mises en œuvre devraient être très prudentes avec ce schéma si MSG[n].KEK est utilisée pour déduire MSG[n].CEK, par exemple, si MSG[n].CEK était l'inversion d'octets de MSG[n].KEK, ce schéma pourrait alors résulter en une clé fixée utilisée de façon inattendue.

7. Références

[RFC2026] S. Bradner, "Le processus de normalisation de l'Internet -- Révision 3", (BCP0009) octobre 1996. *(Remplace* RFC1602, RFC1871) (*MàJ par* RFC3667, RFC3668, RFC3932, RFC3979, RFC3978, RFC5378)

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

[RFC2630] R. Housley, "Syntaxe de message cryptographique", juin 1999. *(Obsolète, voir RFC5652 STD70, et 3370)*

[RFC2633] B. Rmasdell, "Spécification de message S/MIME version 3", juin 1999. (*Obsolète, voir* RFC3851) *(P.S.)*

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

Adresse des auteurs

Stephen Farrell,
Baltimore Technologies,
39 Parkgate Street,
Dublin 8
IRELAND
téléphone : +353-1-881-6000
mél : stephen.farrell@baltimore.ie

Sean Turner
IECA, Inc.
9010 Edgepark Road
Vienna, VA 22182
USA
téléphone : +1.703.628.3180
mél : turners@ieca.com

Appendice A Module ASN.1

```
SMIMERcek
    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) modules(0) rcek(13) }

-- Ce module contient les définitions des attributs utilisés pour réutiliser la clé de chiffrement de contenu d'un message
-- dans des messages ultérieurs.

DEFINITIONS DES ÉTIQUETTES IMPLICITES ::=

DÉBUT
-- EXPORTE TOUT --

IMPORTE

     AlgorithmIdentifier DE
        AuthenticationFramework { joint-iso-itu-t ds(5)
            module(1) authenticationFramework(7) 3 } ;

-- La [RFC2898] utilise l'ASN.1 1993 pour définir PBKDF2-params. Comme cette spécification utilise seulement
-- l'ASN.1 1988, la définition est répétée ici pour être complet.

-- La valeur prf field DEFAUT DOIT être utilisée pour cette spécification
     IDENTIFIANT D'OBJET digestAlgorithm ::= { iso(1) member-body(2) us(840) rsadsi(113549) 2}
     IDENTIFIANT D'OBJET id-hmacWithSHA1 ::= {digestAlgorithm 7}

-- La [RFC2898] définit PBKDF2-params en utilisannt l'ASN.1 1993, qui a pour résultat le même codage que celui
-- produit par la définition ci-dessous. Voir cette définition dans la [RFC2898].

     PBKDF2-params ::= SEQUENCE {
      sel CHOIX {
         CHAINE D'OCTETS spécifiée, – DOIT ETRE UTILISÉE
         otherSource AlgorithmIdentifier, – NE PAS UTILISER CE CHAMP
      },
      iterationCount ENTIER (1..MAX),
      keyLength ENTIER (1..MAX) OPTIONEL
     }

-- id-aa est l'arc avec tous les nouveaux attributs authentifiés et non authentifiés produits par le groupe de travail S/MIME.
-- Il est aussi défini dans la [RFC2633].

IDENTIFIANT D'OBJET id-aa ::= {iso(1) member-body(2) usa(840) rsadsi(113549)
            pkcs(1) pkcs-9(9) smime(16) attributes(2)}

-- Cet attribut contient ce qui sera l'identifiant de clé pour les messages suivants
      IDENTIFIANT D'OBJET id-aa-CEKReference ::= { id-aa 30 }
      CEKReference ::= CHAINE D'OCTETS

-- Cet attribut contient une "indication" pour le receveur du nombre de fois que l'origine va utiliser la CEK réutilisée.
IDENTIFIANT D'OBJET id-aa-CEKMaxDecrypts ::= { id-aa 31 }
      CEKMaxDecrypts ::= ENTIER

-- Cet attribut spécifie la fonction de déduction de clé à utiliser lorsque l'opération d'inversion d'octets par défaut ne
-- peut pas être utilisée.

IDENTIFIANT D'OBJET id-aa-KEKDerivationAlg  ::= { id-aa 32 }
      KEKDerivationAlgorithm ::= SEQUENCE {
         kekAlg         AlgorithmIdentifier,
         pbkdf2Param     PBKDF2-params }

FIN
```

Déclaration de droits de reproduction

Remerciement

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