Groupe de travail Réseau
Request for Comments : 3962
Catégorie : En cours de normalisation

K. Raeburn, MIT
février 2005
Traduction Claude Brière de L'Isle

Chiffrement de la norme de chiffrement évolué (AES) pour Kerberos 5

Résumé
L'Institut national des normes et technologies des États-Unis d'Amérique (NIST, *United States National Institute of Standards and Technology*) a choisi une nouvelle norme de chiffrement évolué (AES, *Advanced Encryption Standard*), qui est significativement plus rapide et (on l'espère) plus sûre que le vieil algorithme de la norme de chiffrement des données (DES, *Data Encryption Standard*). Le présent document est la spécification des ajouts de cet algorithme à la suite de système de chiffrement Kerberos.

1. Introduction

Le présent document définit les types de chiffrement et de somme de contrôle pour Kerberos 5 utilisant l'algorithme AES récemment choisi par le NIST. Ces nouveaux types prennent en charge le chiffrement de bloc de 128 bits et les tailles de clé de 128 ou 256 bits.

En utilisant le "profil simplifié" de la [RFC3961], on peut définir une paire de schémas de chiffrement et de somme de contrôle. AES est utilisé avec soustraction de texte chiffré pour éviter l'expansion de message, et SHA-1 [SHA1] est la fonction de somme de contrôle associée.

2. Conventions utilisées dans ce document

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" en majuscules dans ce document sont à interpréter comme décrit dans le BCP 14, [RFC2119].

3. Représentation de clé du protocole

Le profil de la [RFC3961] traite les clés et les chaînes d'octets aléatoires comme conceptuellement différentes. Mais comme l'espace de clés AES est dense, on peut utiliser comme clé toute chaîne binaire de longueur appropriée. On utilise la représentation d'octet pour la clé décrite dans [AES], où le premier bit de la chaîne binaire est le bit de poids fort du premier octet de la représentation de chaîne d'octets.

4. Génération de clé à partir de phrases de passe ou de données aléatoires

Étant donné le format des clés ci-dessus, on peut générer des clés à partir de la quantité appropriée de données aléatoires (128 ou 256 bits) en copiant simplement la chaîne d'entrée.

Pour générer une clé de chiffrement à partir d'une phrase de passe et d'une chaîne de sel, on utilise la fonction PBKDF2 de

PKCS n° 5 v2.0 ([RFC2898]), avec les paramètres indiqués ci-dessous, pour générer une clé intermédiaire (de la même longueur que la clé finale désirée) qui est alors passée dans la fonction DK avec la chaîne ASCII "kerberos" de huit octets comme on le fait pour des3-cbc-hmac-sha1-kd dans la [RFC3961]. (Dans les termes de la [RFC3961], la fonction PBKDF2 produit une "chaîne d'octets aléatoire", donc l'application de la fonction aléatoire à clé bien que ce soit effectivement une simple opération d'identité.) La clé résultante est la clé à long terme d'utilisateur à utiliser avec l'algorithme de chiffrement en question.

```
tkey = random2key(PBKDF2(passphrase, salt, iter_count, keylength))
key = DK(tkey, "kerberos")
```

La fonction pseudo aléatoire utilisée par PBKDF2 sera un HMAC SHA-1 de la phrase de passe et du sel, comme décrit dans l'Appendice B.1 de PKCS#5.

Le nombre d'itérations est spécifié par les paramètres chaîne à clé fournis. La chaîne de paramètres est de quatre octets indiquant un nombre non signé en ordre gros boutien. C'est le nombre d'itérations à effectuer. Si la valeur est 00 00 00 00, le nombre d'itérations à effectuer est de 4 294 967 296 (2**32). (Donc le compte minimum d'itérations exprimable est 1.)

Pour des environnements où un matériel plus lent est la norme, les mises en œuvre de protocoles comme Kerberos peuvent souhaiter limiter le nombre d'itérations pour empêcher une réponse usurpée, fournie par un attaquant, de consommer des quantités de temps de CPU du côté du client ; si une telle limite est mise en œuvre, elle DEVRAIT ne pas être inférieure à 50 000. Même pour des environnements de matériel rapide, 4 millions d'itérations vont vraisemblablement prendre un très long temps ; des limites bien plus grandes peuvent être appliquées, et il peut être avisé que des mises en œuvre permettent l'interruption de cette opération par l'utilisateur si l'environnement le permet.

Si les paramètres chaîne à clé ne sont pas fournis, la valeur utilisée est 00 00 10 00 (4 096 en décimal, qui indique 4 096 itérations).

Noter que ceci n'est pas une exigence, ni même une recommandation, car cette valeur est à utiliser dans une "pré authentification optimiste" (par exemple, pour tenter une pré authentification fondée sur l'horodatage en utilisant la clé à long terme de l'utilisateur sans avoir d'abord communiqué avec le KDC) en l'absence d'informations supplémentaires, ou comme valeur par défaut pour des sites qui l'utilisent comme clé à long terme pour leurs principaux dans leur base de données Kerberos. C'est simplement l'interprétation de l'absence du champ Paramètre chaîne à clé lorsque le KDC a eu l'opportunité de le fournir.

Un échantillon de valeurs d'essai est donné à l'Appendice B.

5. Soustraction de texte chiffré

Le chaînage de bloc de chiffrement est utilisé pour chiffrer les messages, avec la valeur initiale mémorisée dans l'état de chiffrement. À la différence des systèmes de chiffrement Kerberos précédents, on utilise la soustraction de texte chiffré pour traiter le bloc final éventuellement partiel du message.

La soustraction de texte chiffré est décrite aux pages 195-196 de [AC], et à la section 8 de la [RFC2040] ; elle présente l'avantage qu'aucune expansion de message n'est faite durant le chiffrement de messages de taille arbitraire comme c'est normalement fait en mode CBC avec du bourrage. Des erreurs de la [RFC2040] sont mentionnées à l'Appendice A qui est considéré comme faisant partie de la technique de soustraction de texte chiffré utilisée ici.

La soustraction de texte chiffré, comme définie dans la [RFC2040], suppose que plus d'un bloc de texte source est disponible. Si exactement un bloc est à chiffrer, ce bloc est simplement chiffré avec AES (aussi connu sous le nom de mode ECB). Les entrées plus petites que un bloc sont bourrées en fin pour donner une taille de un bloc ; les valeurs des bits de bourrage ne sont pas spécifiées. (Les mises en œuvre PEUVENT utiliser le bourrage tout à zéro, mais les protocoles NE DOIVENT PAS s'appuyer sur le caractère déterministe du résultat. Les mises en œuvre PEUVENT utiliser un bourrage aléatoire, mais les protocoles NE DOIVENT PAS croire que le résultat ne sera pas déterministe. Noter que dans la plupart des cas, le profil de chiffrement Kerberos va ajouter un confondeur aléatoire qui est indépendant de ce bourrage.)

Pour la cohérence, la soustraction de texte chiffré est toujours utilisée pour les deux derniers blocs des données à chiffrer, comme dans la [RFC2040]. Si la longueur des données est un multiple de la taille de bloc, ceci est équivalent au pur mode CBC avec l'échange des deux derniers blocs de texte chiffré.

Une valeur d'essai est donnée à l'Appendice B.

La valeur initiale empruntée à un chiffrement pour l'utiliser dans un chiffrement suivant est l'avant dernier bloc du résultat du chiffrement ; c'est la forme chiffrée du dernier bloc de texte source. Lors du déchiffrement, l'avant dernier bloc du texte chiffré fourni est retransformé en prochaine valeur d'initialisation. Si un seul bloc de texte chiffré est disponible (déchiffrer un bloc, ou chiffrer un bloc ou moins) ce seul bloc est alors exécuté à la place.

6. Paramètres de profil d'algorithme Kerberos

Voici un résumé des paramètre à utiliser avec le profil simplifié de l'algorithme décrit dans la [RFC3961] :

Format de la clé de protocole	Chaîne binaire de 128 ou 256
Fonction chaîne à clé	PBKDF2+DK avec un compte d'itérations variable (voir ci-dessus
Paramètres de chaîne à clé par défaut	00 00 10 00
Longueur du germe de génération de clé	taille de clé
Fonction aléatoire à clé	fonction d'identité
Fonction de hachage, H	SHA-1
Taille de résultat HMAC, h	12 octets (96 bits)
Taille de bloc de message, m	1 octet
Fonction de chiffrement/déchiffrement, E et D	AES en mode CBC-CTS (taille de bloc de chiffrement 16 octets), avec du prochain au dernier bloc (dernier bloc si il n'y en a qu'un) comme ivec de style CBC.

Utiliser ce profil avec chaque taille de clé donne deux définitions de chaque algorithme de chiffrement et de somme de contrôle.

7. Numéros alloués

Les numéros de type de chiffrement suivants sont alloués :

Types de chiffrement

Nom de type	**Valeur de etype**	**Taille de clé**
aes128-cts-hmac-sha1-96	17	128
aes256-cts-hmac-sha1-96	18	256

Les numéros de type de somme de contrôle sont alloués :

Types de comme de contrôle

Nom du type	**Valeur du sumtype**	**Longueur**
hmac-sha1-96-aes128	15	96
hmac-sha1-96-aes256	16	96

Ces types de somme de contrôle seront utilisés avec les types de chiffrement correspondants définis ci-dessus.

8. Considérations pour la sécurité

Ce nouvel algorithme n'a pas été en service depuis assez longtemps pour recevoir les décennies d'intenses analyses qu'a subi DES. Il est possible qu'existent certaines faiblesses qui n'ont pas été découvertes par les cryptographes qui analysent ces algorithmes avant et durant le processus de sélection d'AES.

L'utilisation de la fonction HMAC a des inconvénients pour certaines longueurs de phrase de passe. Par exemple, une phrase de passe plus longue que la taille de bloc de la fonction de hachage (64 octets, pour SHA-1) est hachée en une plus petite taille (20 octets) avant d'appliquer l'algorithme HMAC principal. Cependant, l'entropie est généralement éparse dans les phrases de passe, en particulier dans les longues, de sorte que ce peut n'être pas un problème dans les rares cas d'utilisateurs qui ont de longues phrases de passe.

Aussi, générer une clé de 256 bits à partir d'une phrase de passe de toute longueur peut être décevant, car l'entropie effective dans une clé déduite d'une phrase de passe ne peut pas être presque aussi grande étant données les propriétés de la fonction de chaîne à clé décrite ici.

Le compte d'itérations dans PBKDF2 paraît utile principalement comme un multiplicateur constant pour la quantité de travail requise pour un attaquant qui utilise la méthode de la force brute. Malheureusement, il multiplie aussi, de la même quantité, le travail nécessaire par un utilisateur légitime avec un mot de passe valide. Donc, le facteur de travail imposé à un attaquant (qui peut avoir de nombreux postes de travail puissants à sa disposition) doit être mis en balance avec le facteur de travail imposé à l'utilisateur légitime (qui peut avoir un PDA ou un téléphone cellulaire) ; la puissance de calcul disponible de chaque côté augmente avec le temps qui passe aussi. La meilleure façon de faire face à l'attaque en force brute est le mécanisme de pré authentification qui assure une meilleure protection de la clé à long terme de l'utilisateur. L'utilisation de tels mécanismes sort du domaine d'application du présent document.

Si un site souhaite utiliser ce moyen de protection contre l'attaque en force brute, le compte d'itérations devrait être choisi sur la base des facilités disponibles à l'attaquant et à l'utilisateur légitime, et de la quantité de travail que l'attaquant va être obligé d'effectuer pour acquérir la clé ou le mot de passé.

Par exemple : les essais de l'auteur sur un système Pentium 4 à 2 GHz ont indiqué qu'en une seconde, près de 90 000 itérations peuvent être faites produisant une clé de 256 bits. Cela a été fait en utilisant la mise en œuvre d'assemblage SHA-1 à partir de OpenSSL, et une version préliminaire du code PBKDF2 pour le paquetage Kerberos du MIT, sur un seul système. On n'a pas tenté de faire de hachages multiples en parallèle, et on suppose qu'un attaquant qui le ferait réussirait au moins 100 000 itérations par seconde – arrondi à 2**17, pour faciliter le calcul. Pour faire simple, on suppose aussi que le coût de l'étape finale de chiffrement AES est négligeable.

Paul Leach estime [LEACH] qu'un dictionnaire de cassage de mot de passe peut avoir de l'ordre de 2**21 entrées, avec des majuscules, la ponctuation, et autres variations qui contribuent peut-être à un facteur de 2**11, donnant une estimation de 2**32.

Donc, pour un compte d'itérations connu N et une chaîne de sel connue, un attaquant avec un certain nombre d'ordinateurs comparables à celui de l'auteur aurait besoin en gros de N*2**15 secondes de CPU pour convertir la totalité du dictionnaire plus les variations en clés.

Un attaquant qui utilise une douzaine d'ordinateurs pendant un mois aura en gros 2**25 secondes de CPU disponibles. Ainsi, utiliser 2**12 (4 096) itérations signifierait qu'un attaquant avec une douzaine de tels ordinateurs dédiés à une attaque en force brute contre une seule clé (en fait, toute clé déduite d'un mot de passe partageant le même sel et le même compte d'itérations) traiterait toutes les variantes des entrées du dictionnaire en quatre mois, et en moyenne, trouverait probablement le mot de passe de l'utilisateur en deux mois.

Donc, si cette forme d'attaque est un problème, les usagers devraient changer leur mot de passe au bout de quelques mois, et un compte d'itération supérieur de quelques ordres de magnitude devrait être choisi. Peut être plusieurs ordres de grandeur, car de nombreux utilisateurs vont tendre à utiliser les mots de passe les plus courts et les plus simples (dans la mesure où ils le peuvent, étant données les vérifications de qualité des mots de passe d'un site) que l'attaquant va probablement essayer en premier.

Comme cette estimation se fonde sur la puissance de CPU disponible actuellement, les comptes d'itérations utilisés pour ce mode de défense devraient être augmentés avec le temps, peut-être de 40% à 60% chaque année.

Noter que si l'attaquant a une grande quantité de mémoire disponible, les résultats intermédiaires peuvent être mis en antémémoire, épargnant pas mal de travail pour la prochaine attaque avec le même sel et un plus grand nombre d'itérations que n'avaient été effectuées au moment où les résultats intermédiaires ont été mémorisés. Donc, il serait sage de générer une nouvelle chaîne aléatoire de sel lorsque les mots de passe sont changés. La chaîne de sel par défaut, déduite du nom du principal, ne protège que contre l'utilisation d'un dictionnaire de clés contre plusieurs utilisateurs.

Si le compte d'itérations PBKDF2 peut être usurpé par un intrus sur le réseau, et si la limite sur le compté d'itérations accepté est très élevé, l'intrus peut être capable d'introduire une forme d'attaque de déni de service contre le client par l'envoi d'un compte d'itérations très élevé, causant une dépense énorme de temps de CPU au client pour calculer une clé incorrecte.

Un intrus qui usurpe la réponse du KDC, fournissant un faible compte d'itération et lisant la réponse du client sur le réseau, peut être capable de réduire le travail nécessaire à une attaque en force brute comme mentionné ci-dessus.

Donc, les mises en œuvre peuvent chercher à appliquer de plus faibles limites au nombre d'itérations qui seront utilisées.

Comme les modèles de menaces et l'équipement typique de l'utilisateur final vont varier largement d'un site à l'autre, il est recommandé de permettre une configuration spécifique de site pour de telles limites.

Tout avantage contre d'autres attaques spécifiques des algorithmes HMAC ou SHA-1 est probablement réalisé avec un très

petit nombre d'itérations.

Dans le cas de la "pré authentification optimiste" mentionnée à la Section 3, le client peut tenter de produire une clé sans d'abord communiquer avec le KDC. Si le client n'a pas d'informations supplémentaires, il peut seulement deviner le compte d'itérations à utiliser. Même une heuristique telle que "le compte d'itération X qui a été utilisé pour acquérir des tickets pour le même principal il y a seulement N heures" peut être fausse. Étant donnée la recommandation ci-dessus pour augmenter le compte d'itération utilisé avec le temps, il est impossible de recommander une valeur par défaut spécifique pour ce cas ; permettre une configuration du site local est recommandé. (Si les vérifications de limite inférieure et supérieure décrites ci-dessus sont mises en œuvre, le compte par défaut pour la pré authentification optimiste devrait être dans ces limites.)

Le mode de soustraction du texte chiffré, qui n'exige pas de bourrage supplémentaire dans la plupart des cas, va révéler la longueur exacte de chaque message qui est chiffré plutôt que de simplement le renfermer dans une petite gamme de longueurs possibles comme en mode CBC. On ne devrait en aucun cas compter sur un tel obscurcissement à des niveaux supérieurs ; si la longueur doit être cachée à un observateur extérieur, ceci devrait être fait en variant intentionnellement la longueur du message à chiffrer.

9. Considérations relatives à l'IANA

Les valeurs de chiffrement et de type de somme de contrôle Kerberos utilisées dans la Section 7 ont été précédemment réservées dans la [RFC3961] pour les mécanismes définis dans le présent document. Les registres ont été mis à jour pour citer le présent document comme référence.

10. Remerciements

Merci à John Brezak, Gerardo Diaz Cuellar, Ken Hornstein, Paul Leach, Marcus Watts, Larry Zhu, et les autres pour leurs retours sur les versions antérieures de ce document.

Annexe A. Errata pour la section 8 de la RFC 2040

(Copié du site Errata de l'éditeur des RFC le 8 juillet 2004.)

Rapporté par : Bob Baldwin; baldwin@plusfive.com
Date : vendredi 26 mars 2004 06:49:02-0800

À la Section 8, Description de RC5-CTS, de la méthode de chiffrement, il est dit :1. OUixer Pn-1 avec le bloc de texte chiffré précédent, Cn-2, pour créer Xn-1.

On devrait dire :
1. OUixer Pn-1 avec le bloc de texte chiffré précédent, Cn-2, pour créer Xn-1. Pour les messages courts où Cn-2 n'existe pas, utiliser l'IV.

Rapporté par : Bob Baldwin; baldwin@plusfive.com
Date : lundi 22 mars 2004 20:26:40-0800

À la Section 8, premier paragraphe, la seconde phrase dit : Ce mode traite toute longueur de texte source et produit du texte chiffré dont la longueur correspond à la longueur du texte source.

On devrait lire : Ce mode traite toute longueur de texte source supérieure à un bloc et produit un texte chiffré dont la longueur correspond à la longueur du texte source.

À la Section 8, l'étape 6 de la méthode de déchiffrement dit :
6. Déchiffrer En pour créer Pn-1.

Elle devrait dire :
6. Déchiffrer En et OUixer avec Cn-2 pour créer Pn-1. Pour les messages courts où Cn-2 n'existe pas, utiliser l'IV.

Annexe B. Échantillon de valeurs d'essai

Échantillon de valeurs pour la fonction chaîne à clé PBKDF2 HMAC-SHA1.

Compte d'itérations = 1
Phrase de passe = "password"
Sel = "ATHENA.MIT.EDUraeburn"
Résultat PBKDF2 de 128 bits : cd ed b5 28 1b b2 f8 01 56 5a 11 22 b2 56 35 15
Clé AES de 128 bits : 42 26 3c 6e 89 f4 fc 28 b8 df 68 ee 09 79 9f 15
Résultat PBKDF2 de 256 bits :
cd ed b5 28 1b b2 f8 01 56 5a 11 22 b2 56 35 15
0a d1 f7 a0 4b b9 f3 a3 33 ec c0 e2 e1 f7 08 37
Clé AES de 256 bits : fe 69 7b 52 bc 0d 3c e1 44 32 ba 03 6a 92 e6 5b
bb 52 28 09 90 a2 fa 27 88 39 98 d7 2a f3 01 61

Compte d'itérations = 2
Phrase de passe = "password"
Sel ="ATHENA.MIT.EDUraeburn"
Résultat PBKDF2 de 128 bits : 01 db ee 7f 4a 9e 24 3e 98 8b 62 c7 3c da 93 5d
Clé AES de 128 bits : c6 51 bf 29 e2 30 0a c2 7f a4 69 d6 93 bd da 13
Résultat PBKDF2 de 256 bits :
01 db ee 7f 4a 9e 24 3e 98 8b 62 c7 3c da 93 5d
a0 53 78 b9 32 44 ec 8f 48 a9 9e 61 ad 79 9d 86
Clé AES de 256 bits :
a2 e1 6d 16 b3 60 69 c1 35 d5 e9 d2 e2 5f 89 61
02 68 56 18 b9 59 14 b4 67 c6 76 22 22 58 24 ff

Compte d'itérations = 1200
Phrase de passe = "password"
Sel = "ATHENA.MIT.EDUraeburn"
Résultat PBKDF2 de 128 bits : 5c 08 eb 61 fd f7 1e 4e 4e c3 cf 6b a1 f5 51 2b
Clé AES de 128 bits : 4c 01 cd 46 d6 32 d0 1e 6d be 23 0a 01 ed 64 2a
Résultat PBKDF2 de 128 bits :
5c 08 eb 61 fd f7 1e 4e 4e c3 cf 6b a1 f5 51 2b
a7 e5 2d db c5 e5 14 2f 70 8a 31 e2 e6 2b 1e 13Clé AES de 256 bits :
55 a6 ac 74 0a d1 7b 48 46 94 10 51 e1 e8 b0 a7
54 8d 93 b0 ab 30 a8 bc 3f f1 62 80 38 2b 8c 2a

Compte d'itérations = 5
Phrase de passe = "password"
Sel =0x1234567878563412
Résultat PBKDF2 de 128 bits : d1 da a7 86 15 f2 87 e6 a1 c8 b1 20 d7 06 2a 49
Clé AES de 128 bits : e9 b2 3d 52 27 37 47 dd 5c 35 cb 55 be 61 9d 8e
Résultat PBKDF2 de 128 bits :
d1 da a7 86 15 f2 87 e6 a1 c8 b1 20 d7 06 2a 49
3f 98 d2 03 e6 be 49 a6 ad f4 fa 57 4b 6e 64 ee
Clé AES de 256 bits :
97 a4 e7 86 be 20 d8 1a 38 2d 5e bc 96 d5 90 9c
ab cd ad c8 7c a4 8f 57 45 04 15 9f 16 c3 6e 31
(Cet essai se fonde sur les valeurs données dans la[RFC3211].)

Compte d'itérations = 1200
Phrase de passe = (64 characters)
"XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
Sel ="la phrase de passe est égale à la taille de bloc"
Résultat PBKDF2 de 128 bits : 13 9c 30 c0 96 6b c3 2b a5 5f db f2 12 53 0a c9
Clé AES de 128 bits : 59 d1 bb 78 9a 82 8b 1a a5 4e f9 c2 88 3f 69 ed
Résultat PBKDF2 de 256 bits :
13 9c 30 c0 96 6b c3 2b a5 5f db f2 12 53 0a c9
c5 ec 59 f1 a4 52 f5 cc 9a d9 40 fe a0 59 8e d1
Clé AES de 256 bits :
89 ad ee 36 08 db 8b c7 1f 1b fb fe 45 94 86 b0

56 18 b7 0c ba e2 20 92 53 4e 56 c5 53 ba 4b 34

Compte d'itérations = 1200
Phrase de passe = (65 characters)
"XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
Set "pass phrase exceeds block size"
Résultat PBKDF2 de 128 bits : 9c ca d6 d4 68 77 0c d5 1b 10 e6 a6 87 21 be 61
Clé AES de 128 bits : cb 80 05 dc 5f 90 17 9a 7f 02 10 4c 00 18 75 1d
Résultat PBKDF2 de 256 bits :
9c ca d6 d4 68 77 0c d5 1b 10 e6 a6 87 21 be 61
1a 8b 4d 28 26 01 db 3b 36 be 92 46 91 5e c8 2a
Clé AES de 256 bits :
d7 8c 5c 9c b8 72 a8 c9 da d4 69 7f 0b b5 b2 d2
14 96 c8 2b eb 2c ae da 21 12 fc ee a0 57 40 1b

Compte d'itérations = 50
Phrase de passe = g-clef (0xf09d849e)
Set "EXAMPLE.COMpianist"
Résultat PBKDF2 de 128 bits : 6b 9c f2 6d 45 45 5a 43 a5 b8 bb 27 6a 40 3b 39
Clé AES de 128 bits : f1 49 c1 f2 e1 54 a7 34 52 d4 3e 7f e6 2a 56 e5
Résultat PBKDF2 de 256 bits :
6b 9c f2 6d 45 45 5a 43 a5 b8 bb 27 6a 40 3b 39
e7 fe 37 a0 c4 1e 02 c2 81 ff 30 69 e1 e9 4f 52
Clé AES de 256 bits :
4b 6d 98 39 f8 44 06 df 1f 09 cc 16 6d b4 b8 3c
57 18 48 b7 84 a3 d6 bd c3 46 58 9a 3e 39 3f 9e

Valeurs d'essai pour CBC avec suppression de texte chiffré, en utilisant une valeur initiale toute à zéro.

Clé AES de 128 bits : 0000: 63 68 69 63 6b 65 6e 20 74 65 72 69 79 61 6b 69
IV : 0000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Entrée :
0000: 49 20 77 6f 75 6c 64 20 6c 69 6b 65 20 74 68 65
0010: 20
Résultat :
0000: c6 35 35 68 f2 bf 8c b4 d8 a5 80 36 2d a7 ff 7f
0010: 97
IV suivante :
0000: c6 35 35 68 f2 bf 8c b4 d8 a5 80 36 2d a7 ff 7f

IV : 0000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Entrée :
0000: 49 20 77 6f 75 6c 64 20 6c 69 6b 65 20 74 68 65
0010: 20 47 65 6e 65 72 61 6c 20 47 61 75 27 73 20
Résultat :
0000: fc 00 78 3e 0e fd b2 c1 d4 45 d4 c8 ef f7 ed 22
0010: 97 68 72 68 d6 ec cc c0 c0 7b 25 e2 5e cf e5
IV suivante :
0000: fc 00 78 3e 0e fd b2 c1 d4 45 d4 c8 ef f7 ed 22

IV : 0000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Entrée :
0000: 49 20 77 6f 75 6c 64 20 6c 69 6b 65 20 74 68 65
0010: 20 47 65 6e 65 72 61 6c 20 47 61 75 27 73 20 43
Résultat :
0000: 39 31 25 23 a7 86 62 d5 be 7f cb cc 98 eb f5 a8
0010: 97 68 72 68 d6 ec cc c0 c0 7b 25 e2 5e cf e5 84
IV suivante : 0000: 39 31 25 23 a7 86 62 d5 be 7f cb cc 98 eb f5 a8

IV : 0000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Entrée :
0000: 49 20 77 6f 75 6c 64 20 6c 69 6b 65 20 74 68 65
0010: 20 47 65 6e 65 72 61 6c 20 47 61 75 27 73 20 43

0020: 68 69 63 6b 65 6e 2c 20 70 6c 65 61 73 65 2c
Résultat :
0000: 97 68 72 68 d6 ec cc c0 c0 7b 25 e2 5e cf e5 84
0010: b3 ff fd 94 0c 16 a1 8c 1b 55 49 d2 f8 38 02 9e
0020: 39 31 25 23 a7 86 62 d5 be 7f cb cc 98 eb f5
IV suivante : 0000: b3 ff fd 94 0c 16 a1 8c 1b 55 49 d2 f8 38 02 9e

IV : 0000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Entrée :
0000: 49 20 77 6f 75 6c 64 20 6c 69 6b 65 20 74 68 65
0010: 20 47 65 6e 65 72 61 6c 20 47 61 75 27 73 20 43
0020: 68 69 63 6b 65 6e 2c 20 70 6c 65 61 73 65 2c 20
Résultat :
0000: 97 68 72 68 d6 ec cc c0 c0 7b 25 e2 5e cf e5 84
0010: 9d ad 8b bb 96 c4 cd c0 3b c1 03 e1 a1 94 bb d8
0020: 39 31 25 23 a7 86 62 d5 be 7f cb cc 98 eb f5 a8
IV suivante : 0000: 9d ad 8b bb 96 c4 cd c0 3b c1 03 e1 a1 94 bb d8

IV : 0000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Entrée :
0000: 49 20 77 6f 75 6c 64 20 6c 69 6b 65 20 74 68 65
0010: 20 47 65 6e 65 72 61 6c 20 47 61 75 27 73 20 43
0020: 68 69 63 6b 65 6e 2c 20 70 6c 65 61 73 65 2c 20
0030: 61 6e 64 20 77 6f 6e 74 6f 6e 20 73 6f 75 70 2e
Résultat :
0000: 97 68 72 68 d6 ec cc c0 c0 7b 25 e2 5e cf e5 84
0010: 39 31 25 23 a7 86 62 d5 be 7f cb cc 98 eb f5 a8
0020: 48 07 ef e8 36 ee 89 a5 26 73 0d bc 2f 7b c8 40
0030: 9d ad 8b bb 96 c4 cd c0 3b c1 03 e1 a1 94 bb d8
IV suivante : 0000: 48 07 ef e8 36 ee 89 a5 26 73 0d bc 2f 7b c8 40

Références normatives

[AC] Schneier, B., "Applied Cryptography", second edition, John Wiley and Sons, New York, 1996.

[AES] National Institute of Standards and Technology, U.S. Department of Commerce, "Advanced Encryption Standard", Federal Information Processing Standards Publication 197, Washington, DC, novembre 2001.

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

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

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

[RFC3961] K. Raeburn, "Spécifications de chiffrement et de somme de contrôle pour Kerberos 5", février 2005.

[SHA1] National Institute of Standards and Technology, U.S. Department of Commerce, "Secure Hash Standard", Federal Information Processing Standards Publication 180-1, Washington, DC, avril 1995.

Références pour information

[LEACH] Leach, P., message au groupe de travail Kerberos de l'IETF, 5 mai 2003, ftp://ftp.ietf.org/ietf-mail-archive/krb-wg/2003-05.mail .

[RFC3211] P. Gutmann, "Chiffrement fondé sur le mot de passe pour CMS", décembre 2001. (*Obsolète, voir* RFC3370) *(P.S.)*

Adresse de l'auteur

Kenneth Raeburn
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139
USA
mél : raeburn@mit.edu

Déclaration de droits de reproduction

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