Groupe de travail Réseau
Request for Comments : 4344
Catégorie : Sur la voie de la normalisation
Traduction Claude Brière de L'Isle

M. Bellare, UC San Diego
T. Kohno, UC San Diego
C. Namprempre, Thammasat University
janvier 2006

Modes de chiffrement de couche transport de Secure Shell (SSH)

Résumé
Les chercheurs ont découvert que la portion chiffrement authentifiée du protocole de transport SSH actuel est vulnérable à plusieurs attaques.
Le présent document décrit de nouvelles méthodes de chiffrement symétrique pour le protocole de transport Secure Shell (SSH) et donne des recommandations spécifiques sur la fréquence à laquelle les mises en œuvre de SSH devraient changer de clés.

Table des matières

1. Introduction

La portion symétrique du protocole de transport SSH a été conçue de façon à assurer à la fois la confidentialité et l'intégrité des données encapsulées. Les chercheurs ([DAI], [BKN1], [BKN2]) ont cependant identifié plusieurs problèmes de sécurité avec la portion symétrique du protocole de transport SSH, comme décrit dans la [RFC4253]. Par exemple, le mode de chiffrement spécifié dans la [RFC4253] est vulnérable à une attaque de texte en clair choisi contre la confidentialité. De plus, si les clés ne sont pas changées assez fréquemment, le protocole de transport SSH peut laisser fuir des informations sur les données de charge utile. Cette dernière propriété est vraie sans considération du mode de chiffrement utilisé.

Dans [BKN1] et [BKN2], Bellare, Kohno, et Namprempre montrent comment modifier la portion symétrique du protocole de transport SSH de façon à ce qu'il préserve de façon démontrable la confidentialité et l'intégrité contre les attaques de texte en clair choisi, de texte chiffré choisi et les attaques de réaction. Le présent document met en œuvre les recommandations décrites dans [BKN1] et [BKN2].

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].

Les types de données et la terminologie utilisés sont spécifiés dans le document d'architecture [RFC4251].

Le protocole de transport SSH est spécifié dans le document de transport [RFC4253].

3. Changement de clés

La Section 9 de la [RFC4253] suggère que les mises en œuvre de SSH changent de clés après chaque giga octets de données transmises. La [RFC4253] ne discute cependant pas de tous les problèmes qui pourraient survenir si une mise en œuvre de SSH ne change pas de clés assez fréquemment. Cette Section sert à renforcer la suggestion de la [RFC4253] en donnant des limites supérieures fermes au nombre tolérable de chiffrements entre les opérations de changement de clé. À la Section 6, on discute plus en détails des motifs de ces recommandations de changement de clés.

Cette Section fait deux recommandations. De façon informelle, la première recommandation est destinée à se protéger contre de possibles fuites d'informations à travers l'étiquette de MAC, et la seconde recommandation est destinée à se protéger contre de possibles fuites d'informations à travers le chiffrement de bloc. Noter que, selon la longueur de bloc du chiffrement de bloc sous-jacent et la longueur des paquets chiffrés, la première recommandation peut outrepasser la seconde recommandation, ou vice versa.

3.1 Première recommandation de changement de clés

À cause de possibles fuites d'informations à travers l'étiquette de MAC, les mises en œuvre de SSH DEVRAIENT changer de clé au moins tous les 2**32 paquets sortants. Plus explicitement, après un échange de clés, une mise en œuvre de SSH NE DEVRAIT PAS envoyer plus de 2**32 paquets avant de changer de clés à nouveau.

Les mises en œuvre de SSH DEVRAIENT aussi tenter de changer de clé avant de recevoir plus de 2**32 paquets depuis la dernière opération de changement de clés. La façon préférée de faire cela est de changer de clés après avoir reçu plus de 2**31 paquets depuis la dernière opération de changement de clés.

3.2 Seconde recommandation de changement de clés

À cause de la propriété des chiffrements de bloc et de certains modes de fonctionnement d'être vulnérables à l'attaque de l'anniversaire, les mises en œuvre doivent être attentives à ne pas chiffrer trop de blocs avec la même clé de chiffrement.

Soit L la longueur de bloc (en bits) d'un chiffrement de bloc d'une méthode de chiffrement SSH (par exemple, 128 pour AES). Si L est d'au moins 128, alors, après le changement de clés, une mise en œuvre de SSH NE DEVRAIT PAS chiffrer plus de 2**(L/4) blocs avant de changer de clés à nouveau. Si L est d'au moins 128, alors les mises en œuvre de SSH devraient aussi tenter de forcer un changement de clé avant de recevoir plus de 2**(L/4) blocs. Si L est de moins de 128 (ce qui est le cas des plus anciens chiffrements comme 3DES, Blowfish, CAST-128, et IDEA) alors, bien qu'il puisse être trop coûteux de changer de clés tous les 2**(L/4) blocs, il est quand même conseillé aux mises en œuvre de SSH de suivre la recommandation originale de la [RFC4253] : changer de clé au moins une fois tous les giga octets de données transmises.

Noter que si L est inférieur ou égal à 128, alors la recommandation de ce paragraphe se substitue à la recommandation du paragraphe 3.1. Si une mise en œuvre de SSH utilise un chiffrement de bloc avec une plus grande taille de bloc (par exemple, Rijndael avec des blocs de 256 bits) alors les recommandations du paragraphe 3.1 peuvent se substituer aux recommandations de ce paragraphe (selon la longueur des paquets).

4. Modes de chiffrement

Le présent document décrit de nouvelles méthodes de chiffrement à utiliser avec le protocole de transport SSH. Ces

méthodes de chiffrement sont en plus des méthodes de chiffrement décrites au paragraphe 6.3 de la [RFC4253].

On rappelle de la [RFC4253] que les méthodes de chiffrement dans chaque direction d'une connexion SSH DOIVENT fonctionner indépendamment l'une de l'autre et que, quand le chiffrement est effectué, les champs Longueur de paquet, Longueur du bourrage, Charge utile, et Bourrage de chaque paquet DOIVENT être chiffrés avec la méthode choisie. On rappelle de plus que la longueur totale de l'enchaînement de la longueur de paquet, longueur de bourrage, charge utile, et bourrage DOIT être un multiple de la taille du bloc de chiffrement quand la taille du bloc de chiffrement est supérieure ou égale à 8 octets (ce qui est le cas pour toutes les méthodes suivantes).

Le présent document décrit les nouvelles méthodes suivantes :

aes128-ctr : RECOMMANDÉ	AES (Rijndael) en mode SDCTR, avec clé de 128 bits
aes192-ctr : RECOMMANDÉ	AES avec clé de 192 bits
aes256-ctr : RECOMMANDÉ	AES avec clé de 256 bits
3des-ctr : RECOMMANDÉ	Trois clés 3DES en mode SDCTR
blowfish-ctr : FACULTATIF	Blowfish en mode SDCTR
twofish128-ctr : FACULTATIF	Twofish en mode SDCTR, avec clé de 128 bits
twofish192-ctr : FACULTATIF	Twofish avec clé de 192 bits
twofish256-ctr : FACULTATIF	Twofish avec clé de 256 bits
serpent128-ctr : FACULTATIF	Serpent en mode SDCTR, avec clé de 128 bits
serpent192-ctr : FACULTATIF	Serpent avec clé de 192 bits
serpent256-ctr : FACULTATIF	Serpent avec clé de 256 bits
idea-ctr : FACULTATIF	IDEA en mode SDCTR
cast128-ctr : FACULTATIF	CAST-128 en mode SDCTR, avec clé de 128-bits

L'étiquette <cipher>-ctr indique que le chiffrement de bloc <cipher> est à utiliser en mode "compteur de déchiffrement à état pleins" (SDCTR, *Stateful-Decryption Counter*). Soit L la longueur de bloc <cipher> en bits. En mode compteur de déchiffrement à état pleins, l'envoyeur et le receveur tiennent tous deux un compteur interne X de L bits. La valeur initiale de X devrait être la valeur initiale (comme calculée au paragraphe 7.2 de la [RFC4253]) interprétée comme un entier non signé de L bits dans l'ordre des octets du réseau. Si X = (2**L)-1, alors "incrémenter X" a la sémantique traditionnelle de "régler X à 0". On utilise la notation <X> pour signifier "convertir X en une chaîne de L bits dans l'ordre des octets du réseau". Naturellement, les mises en œuvre peuvent différer dans la façon dont la valeur X est mémorisée en interne. Par exemple, les mises en œuvre peuvent mémoriser X comme plusieurs compteurs de 32 bits non signés.

Pour chiffrer un paquet P = P1||P2||...||Pn (où P1, P2, ..., Pn sont chacun des blocs de longueur L) le chiffreur commence par chiffrer <X> avec <cipher> pour obtenir un bloc B1. Le bloc B1 est alors OUXé avec P1 pour générer le bloc de texte chiffré C1. Le compteur X est alors incrémenté, et le processus est répété pour chaque bloc suivant afin de générer le texte chiffré entier C = C1||C2||...||Cn correspondant au paquet P. Noter que le compteur X n'est pas inclus dans le texte chiffré. Noter aussi que le flux de clés peut être pré-calculé et que le chiffrement est parallélisable.

Pour déchiffrer un texe chiffré C = C1||C2||...||Cn, le déchiffreur (qui tient aussi sa propre copie de X) commence par chiffrer sa copie de <X> avec <cipher> pour générer un bloc B1 et ensuite OUX B1 avec C1 pour obtenir P1. Le déchiffreur incrémente alors sa copie du compteur X et répète le processus ci-dessus pour chaque bloc pour obtenir le paquet en texte clair P = P1||P2||...||Pn. Comme précédemment, le flux de clés peut être pré-calculé, et le déchiffrement est parallélisable.

La méthode "aes128-ctr" utilise AES (la norme de chiffrement évolué, anciennement appelée Rijndael) avec des clés de 128 bits [AES]. La taille de bloc est de 16 octets.

Pour l'instant, il paraît probable qu'une future spécification proposera que AES128-ctr soit EXIGÉ ; la mise en œuvre de cet algorithme est très vivement encouragée.

La méthode "aes192-ctr" utilise AES avec des clés de 192 bits.
La méthode "aes256-ctr" utilise AES avec des clés de 256 bits.

La méthode "3des-ctr" utilise triple-DES à trois clés (encrypt-decrypt-encrypt) où les 8 premiers octets de la clé sont utilisés pour le premier chiffrement, les 8 octets suivants pour le déchiffrement, et les 8 octets suivants pour le chiffrement final. Cela exige 24 octets de données de clé (dont 168 bits sont en fait utilisés). La taille de bloc est de 8 octets. Cet algorithme est défini dans [DES].

La méthode "blowfish-ctr" utilise Blowfish avec des clés de 256 bits [SCHNEIER]. La taille de bloc est de 8 octets. (Noter que "blowfish-cbc" de la [RFC4253] utilise des clés de 128 bits.)

La méthode "twofish128-ctr" utilise Twofish avec des clés de 128 bits [TWOFISH]. La taille de bloc est de 16 octets.

La méthode "twofish192-ctr" utilise Twofish avec des clés de 192 bits.

La méthode "twofish256-ctr" utilise Twofish avec des clés de 256 bits.

La méthode "serpent128-ctr" utilise le chiffrement de bloc Serpent [SERPENT] avec des clés de 128 bits. La taille de bloc est de 16 octets.

La méthode "serpent192-ctr" utilise Serpent avec des clés de 192 bits.

La méthode "serpent256-ctr" utilise Serpent avec des clés de 256 bits.

La méthode "idea-ctr" utilise le chiffrement IDEA [SCHNEIER]. La taille de bloc est de 8 octets.

La méthode "cast128-ctr" utilise le chiffrement CAST-128 avec des clés de 128 bits [RFC2144]. La taille de bloc est de 8 octets.

5. Considérations relatives à l'IANA

Les treize noms d'algorithmes de chiffrement définis à la Section 4 ont été ajoutés au registre de noms d'algorithme de chiffrement Secure Shell établi par le paragraphe 4.11.1 de la [RFC4250].

6. Considérations sur la sécurité

Le présent document décrit des méthodes de chiffrement et recommandations supplémentaires pour le protocole de transport SSH [RFC4253]. [BKN1], [BKN2] prouvent que si une application SSH incorpore les méthodes et recommandations décrites dans le présent document, la portion de chiffrement symétrique de cette application va alors résister à une large classe d'attaques contre la confidentialité et l'intégrité.

Cette Section est destinée à aider les développeurs à comprendre les motivations en relation avec la sécurité, ainsi que les possibles conséquences des écarts, des méthodes et recommandations décrites dans ce document. Des motivations et une discussion supplémentaires, ainsi que des preuves de sécurité, figurent dans les articles de recherche [BKN1], [BKN2].

On notera que la notion de "preuve" dans le contexte de [BKN1], [BKN2] est celle d'une sécurité réductionniste orientée vers la pratique : si un attaquant est capable de casser la portion symétrique du protocole de transport SSH en utilisant un certain type d'attaque (par exemple, une attaque de texte chiffré choisi) alors l'attaquant va aussi être capable de casser un des composants sous-jacents du protocole de transport (par exemple, le chiffrement de bloc ou le MAC sous-jacent). Si on fait l'hypothèse raisonnable que les composants sous-jacents (comme AES et HMAC-SHA1) sont sûrs, alors l'attaque contre la portion symétrique du protocole SSH ne peut pas être un succès (car autrement il y aurait une contradiction). Voir les détails dans [BKN1], [BKN2]. En particulier, les attaques ne sont pas impossibles, juste extrêmement improbables (sauf si les blocs de construction, comme AES, ne sont pas sûrs).

Noter aussi que la cryptographie ne joue souvent qu'un petit rôle (mais critique) dans la sécurité globale d'une application. Dans le cas du protocole de transport SSH, même si une application peut mettre en œuvre la portion symétrique du protocole SSH exactement comme décrit dans ce document, l'application peut encore être vulnérable à des attaques non fondées sur le protocole (exemple stupide, une application pourrait sauvegarder les clés de chiffrement en clair dans un fichier non protégé). Par conséquent, même si les méthodes décrites ici sont données avec des preuves de sécurité, les développeurs doivent quand même être prudents quand ils développent des applications qui les mettent en œuvre.

6.1 Considérations de changement de clés

La Section 3 de ce document fait deux recommandations de changement de clés : (1) changer de clés au moins une fois tous les 2**32 paquets, et (2) changer de clés après un certain nombre de blocs chiffrés (par exemple, 2**(L/4) blocs si la longueur de bloc L du chiffrement de bloc est d'au moins 128 bits). Les motifs des recommandations (1) et (2) sont différents, et on considère chaque recommandation tour à tour. En bref, (1) est conçu pour protéger contre la fuite

d'informations à travers le MAC sous-jacent du protocole SSH, et (2) est conçu pour protéger contre la fuite d'informations à travers le schéma de chiffrement sous-jacent du protocole SSH. Noter que selon longueur de bloc L de la méthode de chiffrement et le nombre de blocs chiffrés par paquet, la recommandation (1) peut se substituer à la recommandation (2) ou vice versa.

La recommandation (1) déclare que les mises en œuvre de SSH devraient changer de clés au moins une fois tous les 2**32 paquets. Si plus de 2**32 paquets sont chiffrés et qu'un MAC est appliqué par le protocole de transport SSH entre les changements de clés, alors le protocole de transport SSH peut devenir vulnérable aux attaques en répétition et de ré-arrangement. Cela signifie qu'un adversaire peut être capable de convaincre le receveur d'accepter le même message plus d'une fois ou d'accepter des messages dans le désordre. De plus, le MAC sous-jacent peut commencer à laisser fuir des informations sur les données de charge utile du protocole. Plus précisément, un adversaire cherche une collision entre les MAC associés à deux paquets dont les MAC ont le même numéro de séquence de 32 bits (voir au paragraphe 4.4 de la [RFC4253]). Si une collision est trouvée, alors les données de charge utile associées à ces deux textes chiffrés sont probablement identiques. Noter que ce problème se pose sans considération de la sûreté de la méthode de chiffrement sous-jacente. Noter aussi que bien que la compression des données de charge utile avant le chiffrement, l'ajout d'un MAC, et l'utilisation d'un bourrage aléatoire, puissent réduire le risque de fuite d'informations à travers le MAC sous-jacent, la compression et l'utilisation d'un bourrage aléatoire ne vont pas empêcher les fuites d'informations. Les mises en œuvre qui décident de ne pas changer de clés au moins une fois tous les 2**32 paquets devraient comprendre ces problèmes. Ceux-ci sont discutés plus avant dans [BKN1] et [BKN2].

Une solution de remplacement de la recommandation (1) serait de faire un numéro de séquence de protocole de transport SSH de plus de 32 bits. Le présent document ne suggère pas d'augmenter la longueur du numéro de séquence parce que le faire entraverait l'interopérabilité avec les plus anciennes versions du protocole SSH. Une autre solution de remplacement à la recommandation (1) serait de passer du HMAC de base à un autre MAC, qui par exemple aurait son propre compteur interne. À cause du compteur de 32 bits déjà présent dans le protocole, un tel compteur n'aurait besoin d'être incrémenté qu'une fois tous les 2**32 paquets.

La recommandation 2) déclare que les mises en œuvre de SSH devraient changer de clés avant d'avoir chiffré plus de 2**(L/4) blocs avec la même clé (en supposant que L est d'au moins 128). Cette recommandation est destinée à minimiser le risque des attaques de l'anniversaire contre le chiffrement de bloc sous-jacent de la méthode de chiffrement. Par exemple, il y a une attaque théorique sur la confidentialité contre le mode compteur de déchiffrement à état pleins si un adversaire a la possibilité de chiffrer approximativement 2**(L/2) messages avec la même clé. C'est à cause de ces attaques de l'anniversaire que les mises en œuvre sont fortement invitées à utiliser des chiffrements de bloc sûrs avec de grandes longueurs de blocs. De plus, la recommandation 2) est destinée à protéger un chiffreur contre le chiffrement de plus de 2**L blocs avec la même clé. Le motif en est que, si un chiffreur devait utiliser le mode SDCTR pour chiffrer plus de 2**L blocs avec la même clé, il réutiliserait le flux de clés, ce qui peut conduire à de sérieuses attaques contre la confidentialité [SCHNEIER].

6.2. Considérations de méthode de chiffrement

Les chercheurs ont montré que les méthodes de chiffrement originales fondées sur CBC dans la [RFC4253] sont vulnérables aux attaques de texte source choisi contre la confidentialité [DAI], [BKN1], [BKN2]. Les nouvelles méthodes de chiffrement en mode compteur avec déchiffrement à états pleins décrites à la Section 4 du présent document ont été conçues pour être des remplacements sûrs des méthodes de chiffrement originales décrites dans la [RFC4253].

De nombreuses personnes s'écartent des schémas de chiffrement fondés sur le mode compteur parce que, quand il est utilisé de façon incorrecte (comme quand il est permis au flux de clés de se répéter) le mode compteur peut être très peu sûr. Heureusement, les soucis courants du mode compteur ne s'appliquent pas à SSH à cause des recommandations de changement de clés et à cause de la protection additionnelle fournie par le MAC du protocole de transport. Cette discussion est formalisée avec des preuves de sécurité dans [BKN1], [BKN2].

On notera de plus que quand une des méthodes de chiffrement en mode compteur avec déchiffrement à états pleins (Section 4) est utilisée, le bourrage inclus dans un paquet SSH (Section 4 de la [RFC4253]) n'a alors pas besoin d'être aléatoire (mais peut l'être quand même). Cela élimine le besoin de générer des octets pseudo aléatoires cryptographiquement sûrs pour chaque paquet.

Une propriété du chiffrement en mode compteur est qu'il n'exige pas que les messages soient bourrés à un multiple de la longueur de bloc du chiffrement de bloc. Bien que ne pas bourrer les messages puisse réduire la consommation réseau du protocole, le présent document exige que le bourrage soit un multiple de la longueur de bloc du chiffrement de bloc afin (1) de ne pas altérer la description de paquet de la [RFC4253] et (2) de ne pas laisser fuir des informations précises sur la

longueur des données de charge utile du paquet. (Bien qu'il puisse y avoir quelques économies pour le réseau de ne bourrer qu'à 8 octets même si le chiffrement de bloc utilise des blocs de 16 octets, à cause de 1) on ne fait pas cette recommandation ici.)

En plus du mode compteur à déchiffrement à états pleins, [BKN1], [BKN2] décrivent d'autres méthodes de chiffrement à la sécurité prouvée à utiliser avec le protocole de transport SSH. Les méthodes de mode compteur à déchiffrement à états pleins de la Section 4 sont cependant les solutions de remplacement préférées aux méthodes non sûres de la [RFC4253] parce que le mode compteur à déchiffrement à états pleins est le plus efficace (en termes aussi bien de consommation du réseau que de nombre d'opérations cryptographiques requises par paquet).

Références normatives

[AES] National Institute of Standards et Technology, "Advanced Encryption Standard (AES)", Federal Information Processing Standards Publication 197, novembre 2001.

[DES] National Institute of Standards et Technology, "Data Encryption Standard (DES)", Federal Information Processing Standards Publication 46-3, octobre 1999.

[RFC2119] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997. *(MàJ par* RFC8174)

[RFC2144] C. Adams, "L'algorithme de chiffrement CAST-128", mai 1997. *(Information)*

[RFC4250] S. Lehtinen et C. Lonvick, éd., "Numéros alloués du protocole Secure Shell (SSH)", janvier 2006. *(P.S. ; MàJ par* RFC8268)

[RFC4251] T. Ylonen et C. Lonvick, "Architecture du protocole Secure Shell (SSH)", janvier 2006. *(P.S. ; MàJ par* RFC8308)

[RFC4253] C. Lonvick, "Protocole de couche Transport Secure Shell (SSH)", janvier 2006. *(P.S., MàJ par* RFC6668, 8268, 8308, 8332, 8709)

[SCHNEIER] Schneier, B., "Applied Cryptography Second Edition: Protocols algorithms et source in code in C", Wiley, 1996.

[SERPENT] Anderson, R., Biham, E., et Knudsen, L., "Serpent: A proposal for the Advanced Encryption Standard", NIST AES Proposal, 1998.

[TWOFISH] Schneier, B., et al., "The Twofish Encryptions Algorithm: A 128-bit block cipher, 1st Edition", Wiley, 1999.

Références pour information

[BKN1] Bellare, M., Kohno, T., et Namprempre, C., "Authenticated Encryption in SSH: Provably Fixing the SSH Binary Packet Protocol", Ninth ACM Conference on Computer et Communications Security, 2002.

[BKN2] Bellare, M., Kohno, T., et Namprempre, C., "Breaking et Provably Repairing the SSH Authenticated Encryption Scheme: A Case Study of the Encode-then-Encrypt-et-MAC Paradigm", ACM Transactions on Information et System Security, 7(2), mai 2004.

[DAI] Dai, W., "An Attack Against SSH2 Protocol", à ietf-ssh@netbsd.org , 2002.

Adresse des auteurs

Mihir Bellare
Computer Science et Engineering
University of California at San Diego
9500 Gilman Drive, MC 0404
La Jolla, CA 92093-0404
téléphone : +1 858-534-8833
mél : mihir@cs.ucsd.edu

Tadayoshi Kohno
Computer Science et Engineering
University of California at San Diego
9500 Gilman Drive, MC 0404
La Jolla, CA 92093-0404
téléphone : +1 858-534-8833
mél : tkohno@cs.ucsd.edu

Chanathip Namprempre
Thammasat University
Faculty of Engineering
Electrical Engineering Department
Rangsit Campus, Klong Luang
Pathumthani, Thailand 12121
mél : meaw@alum.mit.edu

Déclaration complète de droits de reproduction

Remerciement
Le financement de la fonction d'édition des RFC est fourni par l'activité de soutien administratif (IASA) de l'IETF.