Groupe de travail Réseau
Request for Comments : 4432
Catégorie : Sur la voie de la normalisation

B. Harris
mars 2006
Traduction Claude Brière de L'Isle

Échange de clé RSA pour le protocole de couche transport Secure Shell (SSH)

Résumé
Le présent mémoire décrit une méthode d'échange de clé pour le protocole Secure Shell (SSH) fondée sur le chiffrement à clé publique de Rivest-Shamir-Adleman (RSA). Elle utilise beaucoup moins de temps de CPU de client que l'algorithme Diffie-Hellman spécifié au titre du protocole cœur, et est donc particulièrement bien adaptée pour les systèmes de clients lents.

Table des matières

1. Introduction

Secure Shell (SSH) [RFC4251] est un protocole sûr de connexion à distance. Le protocole cœur utilise l'échange de clés Diffie-Hellman. Sur les CPU lentes, cet échange de clé peut prendre des dizaines de secondes pour s'effectuer, ce qui peut être irritant pour l'utilisateur. Une version précédente du protocole SSH, décrite dans [SSH1], utilise une méthode d'échange de clés fondée sur le chiffrement à clé publique de Rivest-Shamir-Adleman (RSA) qui consomme un ordre de grandeur de moins de temps de CPU chez le client, et donc est particulièrement adaptée pour les systèmes de client lents comme les appareils mobiles. Le présent mémoire décrit un mécanisme d'échange de clé pour la version de SSH décrite dans la [RFC4251] qui est similaire à celui utilisé par l'ancienne version, et aussi rapide, tout en conservant les avantages de sécurité du protocole plus récent.

2. Conventions utilisées dans le document

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

Les types de données "octet", "chaîne", et "mpint" sont définis à la Section 5 de la [RFC4251].

Le reste de la terminologie et des symboles a la même signification que dans la [RFC4253].

3. Généralités

La méthode d'échange de clés RSA consiste en trois messages. Le serveur envoie au client une clé publique RSA, K_T, dont le serveur détient la clé privée. Cela peut être une clé transitoire générée seulement pour cette connexion SSH, ou elle peut être réutilisée pour plusieurs connexions. Le client génère une chaîne d'octets aléatoires, K, la chiffre en utilisant K_T, et renvoie le résultat au serveur, qui la déchiffre. Le client et le serveur hachent K, K_T, et les divers paramètres d'échange de clé pour générer le hachage de l'échange, H, qui est utilisé pour générer les clés de chiffrement pour la session, et le serveur signe H avec sa clé d'hôte et envoie la signature au client. Le client vérifie alors la clé d'hôte comme décrit à la Section 8 de la [RFC4253].

Cette méthode fournit une identification explicite du serveur comme défini à la Section 7 de la [RFC4253]. Elle exige une clé d'hôte capable de signature.

4. Détails

La méthode d'échange de clé RSA a les paramètres suivants :

HASH : algorithme de hachage pour calculer le hachage d'échange, etc.
HLEN : longueur du résultat de HASH en bits.
MINKLEN : longueur minimum du module RSA transitoire en bits.

Leurs valeurs sont définies à la Section 5 et à la Section 6 pour les deux méthodes définies par ce document.

La méthode utilise les messages suivants.

D'abord, le serveur envoie :

octet SSH_MSG_KEXRSA_PUBKEY
chaîne clé et certificat de clé publique d'hôte serveur (K_S)
chaîne K_T, clé publique transitoire RSA

La clé K_T est codée conformément au schéma "ssh-rsa" décrit au paragraphe 6.6 de la [RFC4253]. Noter qu'à la différence d'une clé d'hôte "ssh-rsa", K_T n'est utilisé que pour le chiffrement, et non pour la signature. Le module de K_T DOIT être long d'au moins MINKLEN bits.

Le client génère un entier aléatoire, K, dans la gamme $0 \leq K < 2^{(KLEN-2*HLEN-49)}$, où KLEN est la longueur du module de K_T, en bits. Le client utilise alors K_T pour chiffrer :

mpint : K, le secret partagé

Le chiffrement est effectué conformément au schéma RSAES-OAEP de la [RFC3447], avec une fonction de génération de gabarit de MGF1-with-HASH, un hachage de HASH, et une étiquette vide. Voir à l'Appendice A une preuve que le codage de K est toujours assez court pour être ainsi chiffré. Ayant effectué le chiffrement, le client envoie :

octet SSH_MSG_KEXRSA_SECRET
chaîne RSAES-OAEP-ENCRYPT(K_T, K)

Noter que la dernière étape de RSAES-OAEP-ENCRYPT est de coder un entier comme une chaîne d'octets en utilisant la

primitive I2OSP de la [RFC3447]. Cela, combiné au codage du résultat comme une "chaîne" SSH, donne un résultat qui est similaire, mais pas identique, au codage SSH "mpint" appliqué à cet entier. C'est le même codage que celui utilisé par les signatures "ssh-rsa" dans la [RFC4253].

Le serveur déchiffre K. Si une erreur de déchiffrement se produit, le serveur DEVRAIT envoyer un SSH_MESSAGE_DISCONNECT avec un code de cause de SSH_DISCONNECT_KEY_EXCHANGE_FAILED et DOIT déconnecter. Autrement, le serveur répond par :

octet SSH_MSG_KEXRSA_DONE
chaîne signature de H avec clé d'hôte

Le hachage H est calculé comme hachage HASH de l'enchaînement de ce qui suit :
chaîne V_C, chaîne d'identification du client (CR et LF exclus)
chaîne V_S, chaîne d'identification du serveur (CR et LF exclus)
chaîne I_C, charge utile du SSH_MSG_KEXINIT du client
chaîne I_S, charge utile du SSH_MSG_KEXINIT du serveur
chaîne K_S, clé d'hôte
chaîne K_T, clé RSA transitoire
chaîne RSAES_OAEP_ENCRYPT(K_T, K), le secret chiffré
mpint K, le secret partagé

Cette valeur est appelés le hachage d'échange, et elle est utilisée pour authentifier l'échange de clés. Le hachage d'échange DEVRAIT être gardé secret.

L'algorithme de signature DOIT être appliqué sur H, et non sur les données originales. La plupart des algorithmes de signature incluent un hachage et un bourrage supplémentaire. Par exemple, "ssh-dss" spécifie le hachage SHA-1. Dans ce cas, les données sont d'abord hachées avec HASH pour calculer H, et H est ensuite haché à nouveau au titre de l'opération de signature.

5. rsa1024-sha1

La méthode "rsa1024-sha1" spécifie l'échange de clés RSA comme décrit ci-dessus avec les paramètres suivants :

HASH : SHA-1, comme défini dans la [RFC3174]
HLEN : 160
MINKLEN : 1024

6. rsa2048-sha256

La méthode "rsa2048-sha256" spécifie l'échange de clés RSA comme décrit ci-dessus avec les paramètres suivants :

HASH : SHA-256, comme défini dans [FIPS-180-2]
HLEN : 256
MINKLEN : 2048

7. Numéros de message

Les numéros de message suivants sont définis :

SSH_MSG_KEXRSA_PUBKEY : 30
SSH_MSG_KEXRSA_SECRET : 31
SSH_MSG_KEXRSA_DONE : 32

8. Considérations sur la sécurité

Les considérations sur la sécurité de la [RFC4251] s'appliquent.

Si la clé privée RSA générée par le serveur est révélée, alors la clé de session est révélée. Le serveur devrait donc s'arranger pour l'écraser de sa mémoire aussitôt qu'il n'en a plus besoin. Si la même clé RSA est utilisée pour plusieurs connexions SSH, un attaquant qui peut trouver la clé privée (soit en factorisant la clé publique, soit par d'autres moyens) va obtenir l'accès à toutes les sessions qui ont utilisé cette clé. Par suite, les serveurs DEVRAIENT utiliser chaque clé RSA pour aussi peu d'échanges de clés que possible.

La [RFC3447] recommande que les clés RSA utilisées avec RSAES-OAEP ne soient pas utilisées avec d'autres schémas, ou avec RSAES-OAEP en utilisant une fonction de hachage différente. En particulier, cela signifie que K_T ne devrait pas être utilisé comme clé d'hôte, ou comme clé de serveur dans les versions antérieures du protocole SSH.

Comme tous les mécanismes d'échange de clés, celui-ci dépend pour sa sécurité de l'aléa des secrets générés par le client (le nombre aléatoire K) et le serveur (la clé privée RSA transitoire). En particulier, il est essentiel que le client utilise un générateur de nombres pseudo aléatoires de haute qualité cryptographique pour générer K. Utiliser un mauvais générateur de nombres aléatoires va permettre à un attaquant de casser tout le chiffrement et la protection d'intégrité de la couche de transport Secure Shell. Voir dans la [RFC4086] les recommandations sur la génération de nombres aléatoires.

La taille de la clé transitoire utilisée devrait être suffisante pour protéger les clés de chiffrement et d'intégrité générées par la méthode d'échange de clés. Pour des recommandations sur ce point, voir la [RFC3766]. La force de RSAES-OAEP dépend en partie de la fonction de hachage qu'il utilise. La [RFC3447] suggère d'utiliser un hachage avec une longueur de sortie de deux fois le niveau de sécurité requis, de sorte que SHA-1 est approprié pour les applications qui exigent jusqu'à 80 bits de sécurité, et SHA-256 pour celles qui exigent jusqu'à 128 bits.

À la différence de la méthode d'échange de clés Diffie-Hellman définie par la [RFC4253], cette méthode permet au client de pleinement déterminer le secret partagé, K. Ceci n'est pas estimé être significatif, car K n'est utilisé que quand il est haché avec les données fournies en partie par le serveur (généralement sous la forme du hachage d'échange, H). Si une extension à SSH devait utiliser K directement et supposer qu'il a été généré par l'échange de clés Diffie-Hellman, cela pourrait produire une faiblesse de la sécurité. Les extensions de protocole qui utilisent K directement devrait être tenues en grande suspicion.

Cette méthode d'échange de clés est conçue pour être résistante aux attaques de collision sur le hachage d'échange, en s'assurant qu'aucun des côtés n'est capable de choisir librement ses entrées au hachage après avoir vu toutes les entrées de l'autre côté. La dernière entrée du serveur est dans SSH_MSG_KEXRSA_PUBKEY, avant qu'il ait vu le choix de K du client. La dernière entrée du client est K et son chiffrement RSA, et la nature unidirectionnelle du chiffrement RSA devrait assurer que le client ne peut pas choisir K de façon à causer une collision.

9. Considérations relatives à l'IANA

L'IANA a alloué les noms "rsa1024-sha1" et "rsa2048-sha256" comme noms de méthode d'échange de clés conformément à la [RFC4250].

10. Remerciements

L'auteur remercie de son assistance Simon Tatham pour la conception de cette méthode d'échange de clés.

Le texte de ce document est dérivé en partie de la [RFC4253].

11. Références

11.1 Références normatives

[FIPS-180-2] National Institute of Standards and Technology (NIST), "Secure Hash Standard (SHS)", FIPS PUB 180-2,

août 2002.

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

[RFC3174] D. Eastlake 3 et P. Jones, "Algorithme US de hachage sécurisé n° 1 (SHA1)", sept. 2001. *(Info, MàJ par* 4634 *et* 6234*)*

[RFC3447] J. Jonsson et B. Kaliski, "Normes de cryptographie à clés publiques (PKCS) n° 1 : Spécifications de la cryptographie RSA version 2.1", février 2003. *(Obsolète, remplacée par* RFC8017*) (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)

11.2 Références pour information

[RFC3766] H. Orman, P. Hoffman, "Détermination de la force des clés publiques utilisées pour l'échange de clés symétriques", avril 2004. (BCP0086)

[RFC4086] D. Eastlake 3rd, J. Schiller, S. Crocker, "Exigences d'aléa pour la sécurité", juin 2005. (*Remplace* RFC1750) (BCP0106)

[SSH1] Ylonen, T., "SSH -- Secure Login Connections over the Internet", 6th USENIX Security Symposium, pp. 37-42, juillet 1996.

Appendice A. Sur la taille de K

Les exigences sur la taille de K sont destinées à assurer qu'il est toujours possible de le chiffrer avec K_T. Le codage mpint de K exige un bit de tête de zéro, le bourrant à un nombre entier d'octets, et un champ long de quatre octets, donnant une longueur maximum en octets, B = (KLEN-2*HLEN-49+1+7)/8 + 4 = (KLEN-2*HLEN-9)/8 (où "/" note la division d'entier arrondi à la valeur inférieure).

La longueur maximum des messages qui peuvent être chiffrés en utilisant RSAEP-OAEP est définie par la [RFC3447] en termes de longueur de la clé en octets, qui est (KLEN+7)/8. La longueur maximum est donc L = (KLEN+7-2*HLEN-16)/8 = (KLEN-2*HLEN-9)/8. Donc la version codée de K est toujours assez petite pour être chiffrée sous K_T.

Adresse de l'auteur

Ben Harris
2a Eachard Road
CAMBRIDGE
CB4 1XA
UNITED KINGDOM

mél : bjh21@bjh21.me.uk

Déclaration complète de droits de reproduction

Propriété intellectuelle

L'IETF ne prend pas position sur la validité et la portée de tout droit de propriété intellectuelle ou autres droits qui pourrait être revendiqués au titre de la mise en œuvre ou l'utilisation de la technologie décrite dans le présent document ou sur la mesure dans laquelle toute licence sur de tels droits pourrait être ou n'être pas disponible ; pas plus qu'elle ne prétend avoir accompli aucun effort pour identifier de tels droits. Les informations sur les procédures de l'ISOC au sujet des droits dans les documents de l'ISOC figurent dans les BCP 78 et BCP 79.

Des copies des dépôts d'IPR faites au secrétariat de l'IETF et toutes assurances de disponibilité de licences, ou le résultat de tentatives faites pour obtenir une licence ou permission générale d'utilisation de tels droits de propriété par ceux qui mettent en œuvre ou utilisent la présente spécification peuvent être obtenues sur répertoire en ligne des IPR de l'IETF à http://www.ietf.org/ipr.

L'IETF invite toute partie intéressée à porter son attention sur tous copyrights, licences ou applications de licence, ou autres droits de propriété qui pourraient couvrir les technologies qui peuvent être nécessaires pour mettre en œuvre la présente 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 fourni par l'activité de soutien administatif de l'IETF (IASA, *IETF Administrative Support Activity*).