

Groupe de travail Réseau
Request for Comments : 3972
 Catégorie : En cours de normalisation

T. Aura, Microsoft Research
 mars 2005
 Traduction Claude Brière de L'Isle

Adresses générées par cryptographie (CGA)

Statut du présent mémoire

Le présent document spécifie un protocole en cours de normalisation de l'Internet pour la communauté de l'Internet, et appelle à des discussions et suggestions pour son amélioration. Prière de se référer à l'édition en cours des "Normes officielles des protocoles de l'Internet" (STD 1) pour connaître l'état de la normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.

Notice de Copyright

Copyright (C) The Internet Society (2005). Tous droits réservés.

Résumé

Le présent document décrit une méthode pour lier une clé de signature publique à une adresse IPv6 dans le protocole sûr de découverte de voisin (SEND, *Secure Neighbor Discovery*). Les adresses générées cryptographiquement (CGA, *Cryptographically Generated Address*) sont des adresses IPv6 pour lesquelles l'identifiant d'interface est généré par le calcul d'une fonction de hachage cryptographique unidirectionnelle à partir d'une clé publique et de paramètres auxiliaires. Le lien entre la clé publique et l'adresse peut être vérifié en recalculant la valeur du hachage et en comparant le hachage à l'identifiant de l'interface. Les messages envoyés depuis une adresse IPv6 peuvent être protégés en attachant la clé publique et les paramètres auxiliaires et en signant le message avec la clé privée correspondante. La protection fonctionne sans autorité de certification ou sans aucune infrastructure de sécurité.

Table des Matières

1. Introduction.....	1
2. Format de CGA.....	2
3. Paramètres de CGA et valeurs de hachage.....	3
4. Génération de CGA.....	4
5. Vérification de CGA.....	5
6. Signatures de CGA.....	6
7. Considérations sur la sécurité.....	7
7.1 Objectifs et limitations de la sécurité.....	7
7.2 Extension de hachage.....	7
7.3 Considérations de confidentialité.....	8
7.4 Protocoles concernés.....	9
8. Considérations relatives à l'IANA.....	9
9. Références.....	9
9.1 Références normatives.....	9
9.2 Références pour information.....	10
Appendice A Exemple de génération de CGA.....	10
Appendice B Remerciements.....	11
Adresse de l'auteur.....	11
Déclaration complète de droits de reproduction.....	11

1. Introduction

Le présent document spécifie une méthode pour associer en toute sécurité une clé publique de chiffrement à une adresse IPv6 dans le protocole sûr de découverte de voisin (SEND, *Secure Neighbor Discovery*) [RFC3971]. L'idée de base est de générer l'identifiant d'interface (c'est-à-dire, les 64 bits de droite) de l'adresse IPv6 en calculant un hachage cryptographique de la clé publique. L'adresse IPv6 résultante est appelée une adresse générée cryptographiquement (CGA, *Cryptographically Generated Address*). La clé privée correspondante peut alors être utilisée pour signer les messages envoyés à partir de l'adresse. Une introduction aux CGA et à leur application à SEND se trouve dans [Aura03] et [AAKMNR].

Le présent document spécifie :

- o comment générer une CGA à partir du hachage cryptographique d'une clé publique et des paramètres auxiliaires,
- o comment vérifier l'association entre la clé publique et la CGA, et
- o comment signer un message envoyé de la CGA, et comment vérifier la signature.

Pour vérifier l'association entre l'adresse et la clé publique, le vérificateur a besoin de savoir l'adresse elle-même, la clé publique, et les valeurs des paramètres auxiliaires. Le vérificateur peut alors vérifier les messages signés par le possesseur de la clé publique (c'est-à-dire, le propriétaire de l'adresse). Aucune infrastructure de sécurité supplémentaire, comme une infrastructure de clé publique (PKI), des autorités de certification, ou autres serveurs de confiance, n'est nécessaire.

Noter que parce que les CGA elles-mêmes ne sont pas certifiées, un attaquant peut créer une nouvelle CGA à partir de tout préfixe de sous réseau et sa propre (ou celle de n'importe qui) clé publique. Cependant, l'attaquant ne peut pas prendre une CGA créée par quelqu'un d'autre et envoyer des messages signés qui paraîtraient venir du propriétaire de cette adresse.

Le format d'adresse et le format des paramètres de CGA sont définis dans les Sections 2 et 3. Des algorithmes détaillés pour générer des adresses et pour les vérifier sont donnés respectivement dans les Sections 4 et 5. La Section 6 définit les procédures pour générer et vérifier les signatures de CGA. Les considérations sur la sécurité de la Section 7 incluent les limitations de la sécurité fondée sur les CGA, le raisonnement derrière la technique d'extension du hachage qui permet des longueurs de hachage efficace au dessus de la limite de 64 bits de l'identifiant d'interface, les implications des CGA sur la confidentialité, et la protection contre les attaques en rapport avec le protocole.

Dans le présent document, les mots clés DOIT, NE DOIT PAS, EXIGÉ, DEVRA, NE DEVRA PAS, DEVRAIT, NE DEVRAIT PAS, RECOMMANDÉ, PEUT, et FACULTATIF sont à interpréter comme décrit dans la [RFC2119].

2. Format de CGA

Lorsque il parle d'adresses, le présent document se réfère aux adresses IPv6 dans lesquelles les 64 bits de gauche d'une adresse de 128 bits forment le préfixe de sous réseau et les 64 bits de droite de l'adresse forment l'identifiant d'interface [RFC3513]. On numérote les bits de l'identifiant d'interface en commençant au bit zéro sur la gauche.

Une adresse générée cryptographiquement (CGA) a un paramètre de sécurité (Sec) qui détermine sa force contre les attaques en force brute. Le paramètre de sécurité est un entier non signé de trois bits, et il est codé dans les trois bits les plus à gauche (c'est-à-dire, les bits 0 - 2) de l'identifiant d'interface. Cela peut s'écrire comme suit :

$$\text{Sec} = (\text{identifiant d'interface} \& 0xe000000000000000) \gg 61$$

La CGA est associée à un ensemble de paramètres qui consistent en une clé publique et des paramètres auxiliaires. Deux valeurs de hachage Hash1 (64 bits) et Hash2 (112 bits) sont calculées à partir des paramètres. Les formats de clé publique et des paramètres auxiliaires, et la façon de calculer les valeurs de hachage, sont définis à la Section 3.

Une adresse générée cryptographiquement est définie comme une adresse IPv6 qui satisfait aux deux conditions suivantes :

- o La première valeur de hachage, Hash1, est égale à l'identifiant d'interface de l'adresse. Les bits 0, 1, 2, 6, et 7 (c'est-à-dire, les bits qui codent le paramètre de sécurité Sec et les bits "u" et "g" provenant du format standard d'architecture d'adresse IPv6 de l'identifiant d'interface [RFC3513]) sont ignorés dans la comparaison.
- o Les 16*Sec bits les plus à gauche de la seconde valeur de hachage, Hash2, sont à zéro.

La définition ci-dessus peut être donnée dans les termes des deux gabarits binaires suivants :

Gabarit1 (64 bits) = 0x1cffffffffffff

Gabarit2 (112 bits) = 0x00000000000000000000000000000000 si Sec=0,
 0xffff0000000000000000000000000000 si Sec=1,
 0xffffffff000000000000000000000000 si Sec=2,
 0xffffffffffff00000000000000000000 si Sec=3,
 0xffffffffffffff000000000000000000 si Sec=4,
 0xffffffffffffffff0000000000000000 si Sec=5,
 0xffffffffffffffffffff00000000 si Sec=6, et
 0xffffffffffffffffffffff si Sec=7

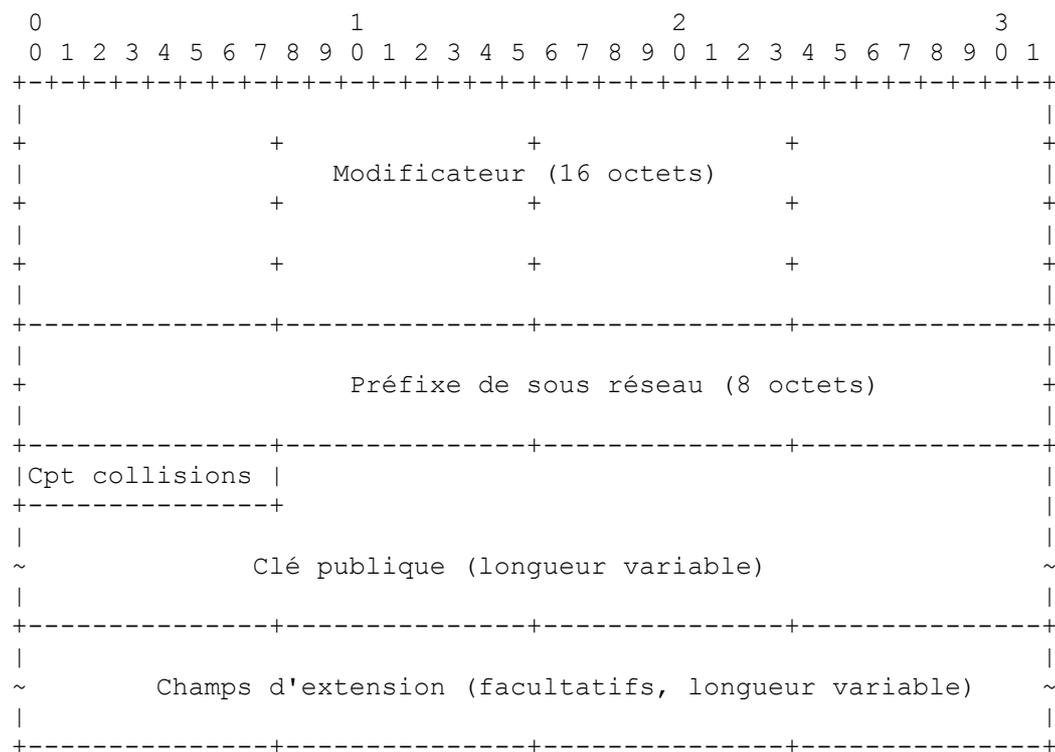
Une adresse générée cryptographiquement est une adresse IPv6 pour laquelle tiennent les deux équations suivantes :

Hachage1 & Gabarit1 == identifiant d'interface & Gabarit1

Hachage2 & Gabarit2 == 0x00000000000000000000000000000000

3. Paramètres de CGA et valeurs de hachage

Chaque CGA est associée à la structure de données de paramètres de CGA, qui a le format suivant :



Modificateur : ce champ contient un entier non signé de 128 bits, qui peut être de n'importe quelle valeur. Le modificateur est utilisé durant la génération de la CGA pour mettre en œuvre l'extension de hachage et améliorer la confidentialité en ajoutant de l'aléa à l'adresse.

Préfixe de sous réseau : ce champ contient les 64 bits du préfixe de sous réseau de la CGA.

Compte de collisions : c'est un entier non signé de huit bits qui DOIT être 0, 1, ou 2. Le compte de collision est incrémenté durant la génération de la CGA pour récupérer d'une collision d'adresses détectée par la détection d'une adresse dupliquée.

Clé publique : c'est un champ de longueur variable qui contient la clé publique du propriétaire de l'adresse. La clé publique DOIT être formatée comme une structure ASN.1 codée en DER [UIT.X690] du type SubjectPublicKeyInfo, défini dans le profil de certificat Internet X.509 [RFC3280]. SEND DEVRAIT utiliser une paire de clés RSA publique/privée. Lorsque RSA est utilisé, l'identifiant d'algorithme DOIT être rsaEncryption, qui est 1.2.840.113549.1.1.1, et la clé publique RSA DOIT être formatée en utilisant le type RSAPublicKey comme spécifié au paragraphe 2.3.1 de la [RFC3279]. La longueur de la clé RSA DEVRAIT être d'au moins 384 bits. D'autres types de clé publique sont indésirables dans SEND, car ils peuvent résulter en des incompatibilités entre mises en œuvre. La longueur de ce champ est déterminée par le codage ASN.1.

Champs d'extension : c'est un champ facultatif de longueur variable qui n'est pas utilisé dans la spécification actuelle. De futures versions de la présente spécification pourront utiliser ce champ pour des éléments de données supplémentaires qui auraient besoin d'être inclus dans la structure des données des paramètres de CGA. Une action de normalisation de l'IETF est exigée pour spécifier l'utilisation des champs d'extension. Les mises en œuvre DOIVENT ignorer la valeur de tous champs d'extension non reconnus.

Les deux valeurs de hachages DOIVENT être calculées comme suit. L'algorithme de hachage SHA-1 [FIPS.180-1] est appliqué aux paramètres de CGA. Lorsque Hachage1 est calculé, l'entrée de l'algorithme SHA-1 est la structure de données des paramètres de CGA. Les 64 bits de Hachage1 sont obtenus en prenant les 64 bits les plus à gauche de la valeur du hachage SHA-1 de 160 bits. Lorsque Hachage2 est calculé, l'entrée est la même structure de données des paramètres de CGA sauf que le préfixe de sous réseau et le compte de collision sont réglés à zéro. Les 112 bits de Hachage2 sont obtenus en prenant les 112 bits de gauche de la valeur du hachage SHA-1 de 160 bits. Noter que les valeurs de hachage sont calculées sur la structure de données des paramètres de CGA entière, y compris tous les champs d'extension non reconnus.

4. Génération de CGA

Le processus de génération d'une nouvelle CGA prend trois valeurs d'entrée : un préfixe de sous réseau de 64 bits, la clé publique du propriétaire de l'adresse comme structure ASN.1 codée en DER du type SubjectPublicKeyInfo, et le paramètre de sécurité Sec, qui est un entier non signé de trois bits. Le coût de génération d'une nouvelle CGA est une fonction exponentielle du paramètre de sécurité Sec, qui peut avoir des valeurs de 0 à 7.

Une CGA et ses paramètres associés DEVRAIENT être générés comme suit :

1. Régler le modificateur à une valeur de 128 bits aléatoire ou pseudo aléatoire.
2. Enchaîner de gauche à droite le modificateur, neuf octets de zéros, la clé publique codée, et tous les champs d'extension facultatifs. Exécuter l'algorithme SHA-1 sur l'enchaînement. Prendre les 112 bits les plus à gauche de la valeur du hachage SHA-1. Le résultat est Hachage2.
3. Comparer les 16*Sec bits de gauche de Hachage2 à zéro. Si ils sont tous à zéro (ou si Sec=0), continuer à l'étape 4. Sinon, incrémenter le modificateur de un et retourner à l'étape 2.
4. Régler à zéro le compte de collision de 8 bits.
5. Enchaîner de gauche à droite la valeur finale de modificateur, le préfixe de sous réseau, le compte de collisions, la clé publique codée, et tous les champs d'extension facultatifs. Exécuter l'algorithme SHA-1 sur l'enchaînement. Prendre les 64 bits de gauche de la valeur du hachage SHA-1. Le résultat est Hachage1.
6. Former un identifiant d'interface à partir de Hachage1 en écrivant la valeur de Sec dans les trois bits de gauche et en réglant les bits 6 et 7 (c'est-à-dire, les bits "u" et "g") à zéro.
7. Enchaîner les 64 bits du préfixe de sous réseau et les 64 bits de l'identifiant d'interface pour former une adresse IPv6 de 128 bits avec le préfixe de sous réseau à gauche et l'identifiant d'interface à droite, comme dans une adresse IPv6 standard [RFC3513].
8. Effectuer si nécessaire la détection de duplication d'adresse, selon la [RFC3971]. Si une collision d'adresse est détectée, incrémenter le compte de collision de un et retourner à l'étape 5. Cependant, après trois collisions, arrêter et rapporter l'erreur.
9. Former la structure de données de paramètres de CGA en enchaînant de gauche à droite la valeur finale du modificateur, le préfixe de sous réseau, la valeur du compte final de collisions, la clé publique codée, et tous les champs d'extension facultatifs.

Le résultat de l'algorithme de génération d'adresse est une nouvelle CGA et une structure de données de paramètres de CGA.

La valeur initiale du modificateur dans l'étape 1 DEVRAIT être choisie au hasard pour rendre sans lien les adresses générées à partir de la même clé publique, ce qui améliore la confidentialité (voir le paragraphe 7.3). La qualité du générateur de nombres aléatoires n'affecte pas la force du lien entre l'adresse et la clé publique. Les mises en œuvre qui n'ont pas de forts nombres aléatoires disponibles PEUVENT utiliser un générateur de nombres pseudo aléatoire non cryptographique initialisé avec l'heure courante.

Pour Sec=0, l'algorithme ci-dessus est déterministe et relativement rapide. Les nœuds qui mettent en œuvre la génération de CGA PEUVENT toujours utiliser la valeur de paramètre de sécurité Sec=0. Si Sec=0, les étapes 2 à 3 de l'algorithme de génération peuvent être sautées.

Pour les valeurs de Sec supérieures à zéro, il n'est pas garanti que l'algorithme ci-dessus se termine après un certain nombre d'itérations. La recherche en force brute dans les étapes 2 à 3 prend $O(2^{(16*Sec)})$ itérations pour s'achever. L'algorithme a été intentionnellement conçu pour que la génération des CGA avec de fortes valeurs de Sec soit infaisable avec la technologie actuelle.

Les mises en œuvre PEUVENT utiliser des versions optimisées ou par ailleurs modifiées de l'algorithme ci-dessus pour la génération de CGA. Cependant, le résultat de toute version modifiée DOIT satisfaire aux deux exigences suivantes. D'abord, la CGA et la structure de données de paramètres de CGA résultantes DOIVENT être formatées comme spécifié aux Sections 2 et 3. Ensuite, la procédure de vérification de CGA définie à la Section 5 DOIT réussir lorsque invoquée sur le résultat de l'algorithme de génération de CGA. Noter que certaines optimisations impliquent des compromis entre confidentialité et coût de la génération d'adresse.

Une optimisation est particulièrement importante. Si le préfixe de sous réseau de l'adresse change mais si la clé publique du propriétaire de l'adresse ne change pas, la vieille valeur de modificateur PEUT être réutilisée. Si elle est réutilisée, l'algorithme DEVRAIT être redémarré depuis l'étape 4. Cette optimisation évite de répéter la recherche coûteuse d'une valeur acceptable de modificateur mais peut, dans certaines situations, rendre plus facile à un observateur de lier deux adresses l'une à l'autre.

Noter que le présent document ne spécifie pas si la détection d'adresses dupliquées devrait être effectuée et comment est faite la détection. L'étape 8 définit seulement que faire si certaine forme de détection de duplication d'adresse est effectuée et qu'une collision d'adresse est détectée.

De futures versions de la présente spécification pourront spécifier des entrées supplémentaires à l'algorithme de génération de CGA qui sont enchaînées comme champs d'extension à la fin de la structure de données de paramètres de CGA. Aucun de ces champs d'extension n'est défini dans le présent document.

5. Vérification de CGA

La vérification de CGA prend en entrées une adresse IPv6 et une structure de données de paramètres de CGA. Les paramètres de CGA consistent en l'enchaînement du modificateur, du préfixe de sous réseau, du compte de collisions, de la clé publique, et des champs d'extension facultatifs. La vérification réussit ou échoue.

La CGA DOIT être vérifiée selon les étapes suivantes :

1. Vérifier que le compte de collisions dans la structure de données de paramètres de CGA est 0, 1, ou 2. La vérification de CGA échoue si le compte de collisions est hors de la gamme valide.
2. Vérifier que le préfixe de sous réseau dans la structure de données de paramètres de CGA est égal au préfixe de sous réseau (c'est-à-dire, aux 64 bits de gauche) de l'adresse. La vérification de CGA échoue si les valeurs de préfixe diffèrent.
3. Exécuter l'algorithme SHA-1 sur la structure de données de paramètres de CGA. Prendre les 64 bits de gauche de la valeur du hachage SHA-1. Le résultat est Hachage1.
4. Comparer Hachage1 à l'identifiant d'interface (c'est-à-dire, aux 64 bits de droite) de l'adresse. Les différences dans les trois bits de gauche et dans les bits 6 et 7 (c'est-à-dire, les bits "u" et "g") sont ignorées. Si les valeurs de 64 bits diffèrent (autrement que par les cinq bits ignorés) la vérification de CGA échoue.
5. Lire le paramètre de sécurité Sec à partir des trois bits de gauche des 64 bits d'identifiant d'interface de l'adresse. (Sec est un entier non signé de 3 bits.)
6. Enchaîner de gauche à droite le modificateur, les 9 octets de zéros, la clé publique, et tous les champs d'extension qui suivent la clé publique dans la structure de données de paramètres de CGA. Exécuter l'algorithme SHA-1 sur l'enchaînement. Prendre les 112 bits de gauche de la valeur du hachage SHA-1. Le résultat est Hachage2.
7. Comparer les 16*Sec bits de gauche de Hachage2 à zéro. Si l'un d'eux n'est pas zéro, la vérification de CGA échoue. Sinon, la vérification a réussi. (Si Sec=0, la vérification de CGA n'échoue jamais à cette étape.)

Si la vérification échoue à une étape, l'exécution de l'algorithme DOIT être immédiatement arrêtée. D'un autre côté, si la vérification réussit, le vérificateur sait que la clé publique dans les paramètres de CGA est la clé publique authentique du propriétaire de l'adresse. Le vérificateur peut extraire la clé publique en retirant 25 octets au commencement des paramètres de CGA et en décodant la structure de données SubjectPublicKeyInfo suivante.

Noter que les valeurs des bits 6 et 7 (bits "u" et "g") de l'identifiant d'interface sont ignorés durant la vérification de CGA. Dans le protocole SEND, après la réussite de la vérification, le vérificateur DEVRAIT traiter toutes les CGA de la même façon sans considération des valeurs de Sec, du modificateur, et du compte de collisions. En particulier, le vérificateur dans le protocole SEND NE DEVRAIT PAS avoir une politique de sécurité qui différencie entre les adresses sur la base de la valeur de Sec. De cette façon, le générateur d'adresse est libre de choisir n'importe quelle valeur de Sec.

Tous les nœuds qui mettent en œuvre la vérification de CGA DOIVENT être capables de traiter toutes les valeurs de paramètre de sécurité Sec = 0, 1, 2, 3, 4, 5, 6, 7. La procédure de vérification est relativement rapide et exige toujours au plus deux calculs de la fonction de hachage SHA-1. Si Sec=0, la vérification n'échoue jamais dans les étapes 6 - 7 et ces étapes peuvent être sautées.

Les nœuds qui mettent en œuvre la vérification de CGA pour SEND DEVRAIENT être capables de traiter les clés publiques RSA qui ont l'identifiant d'algorithme rsaEncryption et une longueur de clé entre 384 et 2 048 bits. Les mises en œuvre PEUVENT prendre en charge des clés plus longues. De futures versions de la présente spécification pourront recommander la prise en charge de clés plus longues.

Les mises en œuvre de vérification de CGA DOIVENT ignorer la valeur de tout champ d'extension non reconnu qui suit la clé publique dans la structure de données de paramètres de CGA. Cependant, les mises en œuvre DOIVENT inclure ces données non reconnues dans l'entrée de hachage lors du calcul de Hash1 à l'étape 3 et de Hash2 dans l'étape 6 de l'algorithme de vérification de CGA. Ceci est important pour s'assurer de la compatibilité avec les futures extensions.

6. Signatures de CGA

Cette section définit les procédures pour générer et vérifier les signatures de CGA. Pour signer un message, un nœud a besoin de la CGA, de la structure de données de paramètres de CGA associée, du message, et de la clé de chiffrement privée qui correspond à la clé publique dans les paramètres de CGA. Le nœud doit aussi avoir une étiquette de type de 128 bits pour le message provenant de l'espace de noms de type de message CGA.

Pour signer un message, un nœud DEVRAIT faire ce qui suit :

- o Enchaîner l'étiquette de type de 128 bits (dans l'ordre des octets du réseau) et le message avec l'étiquette de type à gauche et le message à droite. L'enchaînement est le message à signer à la prochaine étape.
- o Générer la signature RSA en utilisant l'algorithme de signature RSASSA-PKCS1-v1_5 [RFC3447] avec l'algorithme de hachage SHA-1. La clé privée et l'enchaînement créés ci-dessus sont les entrées de l'opération de génération.

La spécification du protocole SEND [RFC3971] définit plusieurs messages qui contiennent une signature dans l'option Signature. La spécification du protocole SEND définit aussi une étiquette de type à partir de l'espace de nom de type de message CGA. La même étiquette de type est utilisée pour tous les messages SEND qui ont l'option Signature. Cette étiquette de type est un entier de 128 bits alloué par l'IANA qui a été choisi au hasard pour empêcher une collision de type accidentelle avec les messages d'autres protocoles qui utilisent la même clé publique mais qui peuvent ou non utiliser les étiquettes de type allouées par l'IANA.

La CGA, la structure des données des paramètres de CGA, le message, et la signature sont envoyés au vérificateur. La spécification du protocole SEND définit comment ces éléments de données sont envoyés dans les messages du protocole SEND. Noter que l'étiquette de type de 128 bits n'est pas incluse dans les messages du protocole SEND parce que le vérificateur connaît implicitement sa valeur à partir du champ Type de message ICMP dans le message SEND. Voir dans la spécification SEND [RFC3971] les informations précises sur la façon dont SEND traite l'étiquette de type.

Pour vérifier une signature, le vérificateur a besoin de la CGA, de la structure de données de paramètres de CGA associée, du message, et de la signature. Le vérificateur a aussi besoin de l'étiquette de type de 128 bits pour le message.

Pour vérifier la signature, un nœud DEVRAIT faire ce qui suit :

- o Vérifier la CGA comme défini à la Section 5. Les entrées à la vérification de CGA sont la CGA et la structure de données de paramètres de CGA.
- o Enchaîner l'étiquette de type de 128 bits et le message avec l'étiquette de type à gauche et le message à droite. L'enchaînement est le message dont la signature est à vérifier à l'étape suivante.
- o Vérifier la signature RSA en utilisant l'algorithme RSASSA-PKCS1-v1_5 [RFC3447] avec l'algorithme de hachage SHA-1. Les entrées à l'opération de vérification sont la clé publique (c'est-à-dire, la structure RSAPublicKey provenant de la structure SubjectPublicKeyInfo qui fait partie de la structure de données de paramètres de CGA) l'enchaînement créé ci-dessus, et la signature.

Le vérificateur ne DOIT accepter la signature comme authentique que si la vérification de CGA et la vérification de signature ont toutes deux réussi.

7. Considérations sur la sécurité

7.1 Objectifs et limitations de la sécurité

L'objet des CGA est d'empêcher le vol et l'usurpation des adresses IPv6 existantes. Le possesseur de la clé publique de l'adresse est lié cryptographiquement à l'adresse. Le possesseur de l'adresse peut utiliser la clé privée correspondante pour affirmer sa possession et pour signer les messages SEND envoyés depuis l'adresse.

Il est important de comprendre qu'un attaquant peut créer une nouvelle adresse à partir d'un préfixe de sous réseau arbitraire et sa propre clé publique (ou celle de quelqu'un d'autre parce que les CGA ne sont pas certifiées. Cependant, l'attaquant ne peut pas se faire passer pour l'adresse de quelqu'un d'autre. Cela parce que l'attaquant devrait trouver une collision de la valeur de hachage cryptographique Hash1. (La propriété de la fonction de hachage nécessaire ici est appelée résistance seconde de pré image [MOV97].)

Pour chaque structure de données de paramètres de CGA valide, il y a $4 * (\text{Sec} + 1)$ CGA différentes qui correspondent à la valeur. Cela parce que décrémenter la valeur de Sec dans les trois bits de gauche de l'identifiant d'interface n'invalide pas l'adresse, et que le vérificateur ignore les valeurs des bits "u" et "g". Dans SEND, cela n'a aucune conséquence pour la sécurité ou la mise en œuvre.

Une autre limitation des CGA est qu'il n'y a pas de mécanisme pour prouver qu'une adresse n'est pas une CGA. Donc, un attaquant pourrait prendre la CGA de quelqu'un d'autre et la présenter comme une adresse qui n'est pas générée par un procédé cryptographique (par exemple, comme une adresse de la [RFC3041]). Un attaquant ne va pas en tirer profit parce que bien que les nœuds SEND acceptent les messages signés aussi bien que non signés provenant de toute adresse, ils donnent la priorité aux informations des messages signés.

La longueur de clé RSA minimum requise pour SEND est seulement de 384 bits. Des clés aussi courtes sont vulnérables aux attaques de mise en facteurs d'entiers et ne peuvent pas être utilisées pour une authentification ou secret fort. D'un autre côté, le coût d'une mise en facteurs de clés de 384 bits est actuellement assez élevé pour prévenir la plupart des attaques de déni de service. Les mises en œuvre qui utilisaient initialement des clés RSA courtes DEVRAIENT être prêtes à passer à des clés plus longues lorsque des attaques de déni de service résultant de la mise en facteurs d'entier deviendront un problème.

L'impact de la compromission d'une clé sur les CGA dépend de l'application pour laquelle elles sont utilisées. Dans SEND, ce n'est pas un souci majeur. Si la clé de signature privée est compromise parce que le nœud SEND a lui-même été compromis, l'attaquant n'a pas besoin de faire passer ses messages pour des messages SEND provenant du nœud. Lorsque on découvre qu'un nœud a été compromis, une nouvelle clé de signature et une nouvelle CGA DEVRAIENT être générées.

Par ailleurs, si la clé RSA est compromise parce que l'attaque de mise en facteurs d'entiers pour la longueur de clé choisie est devenue praticable, la clé doit être remplacée par une plus longue, comme expliqué ci-dessus. Dans tous les cas, le changement d'adresse révoque effectivement la vieille clé publique. Il n'est pas nécessaire d'avoir des mécanismes de révocation de clé supplémentaires ou de limiter la durée de vie des clés de signature.

7.2 Extension de hachage

Avec la montée en puissance des ordinateurs, les 64 bits de l'identifiant d'interface ne seront pas suffisants pour empêcher des attaquants de chercher des collisions de hachage. On est un peu aidé par le fait d'avoir inclus le préfixe de sous réseau de l'adresse dans l'entrée du hachage. Cela empêche l'attaquant d'utiliser une seule base de données pré calculées pour attaquer les adresses qui ont des préfixes de sous réseau différents. L'attaquant a besoin de créer une base de données séparée pour chaque préfixe de sous réseau. Les adresses de liaison locale restent cependant vulnérables parce que le même préfixe est utilisé par tous les nœuds IPv6.

Pour empêcher que la technologie de CGA ne soit rendue obsolète par l'augmentation de puissance des ordinateurs, la technique de hachage utilisée pour générer les CGA doit être un peu étendue. La technique d'extension choisie est d'augmenter le coût de la génération d'adresse et des attaques en force brute du même facteur paramétré tout en gardant constant le coût de l'utilisation de l'adresse et de la vérification. Cela fournit aussi une protection pour les adresses de liaison locale. L'introduction de l'extension de hachage est la principale différence entre le présent document et les propositions antérieures de CGA [OR01], [RFC4423], [MC02].

Pour réaliser une extension effective de la longueur du hachage, l'entrée de la seconde fonction de hachage, Hash2, est modifiée (en changeant la valeur du modificateur) jusqu'à ce que les $16 * \text{Sec}$ bits de gauche de la valeur du hachage soient à zéro. Cela augmente le coût de la génération d'adresse d'approximativement un facteur de $2^{(16 * \text{Sec})}$. Cela augmente aussi le coût des attaques en force brute du même facteur. C'est-à-dire que le coût de la création d'une structure de données de paramètres de CGA qui lie la clé publique de l'attaquant à l'adresse de quelqu'un d'autre est augmenté de $O(2^{59})$ à

$O(2^{(59+16*Sec)})$). Le générateur d'adresses peut choisir le paramètre de sécurité Sec en fonction de sa propre capacité de calcul, du risque perçu d'attaques, et de la durée de vie attendue de l'adresse. Actuellement, les valeurs de Sec entre 0 et 2 sont suffisantes pour la plupart des nœuds IPv6. Avec la montée en puissance des ordinateurs, des valeurs de Sec plus élevées deviendront utiles.

Théoriquement, si aucune extension de hachage n'est utilisée (c'est-à-dire, si $Sec=0$) et si un attaquant normal est capable d'espionner sur N réseaux locaux en même temps, une attaque contre des adresses de liaison locale est N fois aussi efficace qu'une attaque contre les adresses d'un réseau spécifique. L'effet pourrait être contré par l'utilisation d'une valeur de Sec légèrement plus élevée pour les adresses de liaison locale. Lorsque des valeurs supérieures de Sec (comme $2^{(16*Sec)} > N$) sont utilisées pour toutes les adresses, l'avantage relatif d'attaquer les adresses de liaison locale devient insignifiant.

L'efficacité de l'extension du hachage dépend de l'hypothèse que les capacités de calcul de l'attaquant et du générateur d'adresses vont croître au même taux (potentiellement exponentiel). Ceci n'est pas nécessairement vrai si les adresses sont générées sur des appareils mobiles d'extrémité, pour lesquels le principal objectif de conception est de diminuer les coûts et la taille, plutôt que d'augmenter la puissance de calcul. Mais ceci n'a rien d'obligatoire. La partie coûteuse de la génération d'adresse (étapes 1 à 3 de l'algorithme de génération) peut être déléguée à un ordinateur plus puissant. De plus, ce travail peut être fait à l'avance ou hors ligne, plutôt qu'en temps réel, lorsque on a besoin d'une nouvelle adresse.

Pour permettre aux nœuds mobiles dont le préfixe de sous réseau change fréquemment d'utiliser des valeurs de Sec supérieures à zéro, on a décidé de ne pas inclure le préfixe de sous réseau dans les entrées à Hash2. Le résultat est plus faible que si le préfixe de sous réseau était inclus dans l'entrée des deux hachages. Par ailleurs, notre schéma est au moins aussi fort que si on utilisait la technique d'extension de hachage sans inclure le préfixe de sous réseau dans l'un et l'autre des hachages. Il est aussi d'une force égale à celle de ne pas utiliser l'extension de hachage mais d'inclure le préfixe de sous réseau. Ce compromis a été fait parce que les nœuds mobiles vont fréquemment sur des réseaux non sûrs, où ils courent le risque d'attaques de déni de service (par exemple, durant la procédure de détection d'adresse dupliquée).

Dans la plupart des réseaux, le but de la découverte sûre de voisin et des signatures de CGA est d'empêcher les attaques de déni de service. Donc, il est généralement raisonnable de commencer par utiliser une valeur faible de Sec et de ne remplacer les adresses par des plus fortes que lorsque des attaques de déni de service fondées sur une recherche en force brute deviennent un problème significatif. Si les CGA étaient utilisées au titre d'un mécanisme fort d'authentification ou de secret, il pourrait être nécessaire de commencer par des valeurs plus élevées de Sec.

La valeur de compte de collisions est utilisée pour modifier l'entrée à Hash1 si il y a une collision d'adresse. Il est important de ne pas permettre des valeurs de compte de collision supérieures à 2. D'abord, il est extrêmement peu probable que trois collisions se produisent, et la raison est certainement une erreur de configuration ou de mise en œuvre ou une attaque de déni de service. (Lorsque le protocole SEND est utilisé, les collisions délibérées causées par un attaquant de DoS sont détectées et ignorées.) Ensuite, un attaquant qui fait une recherche en force brute pour correspondre à une certaine CGA peut essayer toutes les différentes valeurs d'un compte de collisions sans répéter la recherche en force brute pour la valeur du modificateur. Donc, si des valeurs supérieures sont permises pour le compte de collisions, la technique d'extension de hachage devient moins efficace pour empêcher les attaques en force brute.

7.3 Considérations de confidentialité

Les CGA peuvent donner le même niveau de pseudonymie que les extensions de confidentialité d'adresse IPv6 définies dans la [RFC3041]. Un hôte IP peut générer plusieurs CGA pseudo aléatoires en exécutant plusieurs fois l'algorithme de génération de CGA de la Section 4 et en utilisant à chaque fois une valeur initiale aléatoire ou pseudo aléatoire différente pour le modificateur. L'hôte devrait changer périodiquement son adresse comme dans la [RFC3041]. Lorsque la protection de la confidentialité est nécessaire, le générateur de nombres (pseudo) aléatoires utilisé dans la génération d'adresse DEVRAIT être assez fort pour produire des valeurs imprévisibles et sans relation entre elles. On trouvera des conseils pour la génération de nombres aléatoires dans la [RFC1750].

Il y a deux limitations apparentes à cette protection de la confidentialité. Cependant, comme on va l'expliquer ci-dessous, aucune n'est vraiment sérieuse.

D'abord, le coût élevé de la génération d'adresse peut empêcher les hôtes qui utilisent une valeur élevée de Sec de changer fréquemment leur adresse. Ce problème est atténué parce que la partie coûteuse de la génération d'adresse peut être faite à l'avance ou hors ligne, comme expliqué au paragraphe précédent. On notera aussi que les nœuds qui bénéficient le plus de valeurs de Sec élevées (par exemple, les serveurs DNS, les routeurs, et les serveurs de données) n'exigent généralement pas de pseudonyme, et les nœud qui ont de fortes exigences de confidentialité (par exemple, les PC clients et les hôtes mobiles) sont des cibles improbables pour de coûteuses attaques de déni de service en force brute et peuvent s'accommoder de valeurs de Sec plus faibles.

Ensuite, la clé publique du propriétaire de l'adresse est révélée dans les messages SEND signés. Cela signifie que si le propriétaire de l'adresse veut utiliser un pseudonyme à l'égard des nœuds dans les liaisons locales auxquelles il accède, il devrait générer non seulement une nouvelle adresse mais aussi une nouvelle clé publique. Cependant, avec les technologies normales de liaison locale, une adresse de couche liaison d'un nœud est un identifiant unique pour le nœud. Tant que le nœud continue d'utiliser la même adresse de couche liaison, il n'y a pas de sens à changer la clé publique pour des raisons de confidentialité.

7.4 Protocoles concernés

Bien que le présent document ne définisse les CGA que pour les besoins de la découverte de voisin sécurisée, d'autres protocoles pourraient être définis ailleurs qui utilisent les mêmes adresses et clés publiques. Cela soulève la possibilité d'attaques de relation de protocole dans lesquelles un message signé provenant d'un protocole est répété dans un autre protocole. Cela signifie que d'autres protocoles (peut-être même ceux conçus sans connaître l'existence de SEND) pourraient mettre en danger la sécurité de SEND. Ce qui rend cette menace encore plus significative est que l'attaquant pourrait créer une CGA à partir de la clé publique de quelqu'un d'autre puis répéter des messages signés à partir d'un protocole qui n'a rien à voir avec les CGA ou les adresses IP.

Pour empêcher l'attaque de relation de protocole, une étiquette de type est ajoutée devant chaque message avant sa signature. Les étiquettes de type font 128 bits et sont des valeurs choisies au hasard, ce qui empêche des collisions de type accidentelles même avec des protocoles de conception très faible qui n'utilisent aucune étiquette de type. De plus, le protocole SEND comporte l'adresse CGA de l'expéditeur dans tous les messages signés. Cela rend encore plus difficile à un attaquant de prendre des messages dans un autre contexte et de les répéter comme messages SEND.

Finalement, une mise en garde forte doit être faite à l'égard de l'utilisation des signatures de CGA pour d'autres objets que SEND. D'abord, les autres protocoles DOIVENT inclure une étiquette de type et l'adresse de l'expéditeur dans tous les messages signés de la même façon que dans SEND. Chaque protocole DOIT définir ses propres valeurs d'étiquette de type comme expliqué à la Section 8. De plus, à cause de la possibilité d'attaques de relation de protocole, la clé publique DOIT être utilisée seulement pour signer, et elle NE DOIT PAS être utilisée pour le chiffrement. Ensuite, la longueur minimum de clé RSA de 384 bits peut être trop courte pour de nombreuses applications et l'impact d'une compromission de clé sur ce protocole particulier doit être évalué. Enfin, l'autorisation fondée sur la CGA convient particulièrement pour sécuriser la découverte de voisin [RFC2461] et la détection d'adresse dupliquée [RFC2462] parce que ce sont des protocoles de signalisation de couche réseau pour lesquels les adresses IPv6 sont des identifiants naturels de point d'extrémité. Dans tout protocole qui utilise d'autres identifiants, comme les noms du DNS, les signatures CGA seules ne sont pas un mécanisme de sécurité suffisant. Il doit aussi y avoir une façon sûre de transposer les autres identifiants en adresses IPv6. Si le but n'est pas de vérifier les revendications d'adresses IPv6, les signatures CGA ne sont probablement pas la bonne solution.

8. Considérations relatives à l'IANA

Le présent document définit un nouvel espace de noms de type de message CGA à utiliser comme étiquette de type dans les messages qui peuvent être signés en utilisant des signatures CGA. Les valeurs de cet espace de noms sont des entiers non signés de 128 bits. Les valeurs dans cet espace de noms sont allouées sur la base du premier qui les demande [RFC2434]. L'IANA alloue les nouvelles valeurs de 128 bits directement sans révision.

Le demandeur DEVRAIT générer les nouvelles valeurs avec un générateur de nombres aléatoires fort. Des gammes continues d'au plus 256 valeurs peuvent être demandées pourvu que les 120 bits de poids fort des valeurs aient été générées avec un générateur de nombres aléatoires fort.

L'IANA ne génère pas de valeurs aléatoires pour le demandeur. L'IANA alloue les valeurs demandées sans vérifier la façon dont elles ont été générées. L'espace de noms est par nature illimité, et tout nombre de valeurs individuelles et de gammes d'au plus 256 valeurs peut être alloué.

Les valeurs de type de message de CGA pour utilisation privée PEUVENT être générées avec un générateur de nombres aléatoire fort sans allocation par l'IANA.

Le présent document ne définit aucune nouvelle valeur dans aucun espace de noms.

9. Références

9.1 Références normatives

[FIPS.180-1] National Institute of Standards and Technology, "Secure Hash Standard", Federal Information Processing Standards Publication FIPS PUB 180-1, avril 1995, < <http://www.itl.nist.gov/fipspubs/fip180-1.htm> >.

- [RFC2119] S. Bradner, "[Mots clés à utiliser](#) dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997.
- [RFC2434] T. Narten et H. Alvestrand, "Lignes directrices pour la rédaction d'une section Considérations relatives à l'IANA dans les RFC", BCP 26, octobre, 1998. (*Rendue obsolète par la [RFC5226](#)*)
- [RFC3279] L. Bassham, W. Polk et R. Housley, "[Algorithmes et identifiants](#) pour le profil de certificat d'infrastructure de clé publique X.509 et de liste de révocation de certificat (CRL) pour l'Internet", avril 2002.
- [RFC3280] R. Housley, W. Polk, W. Ford et D. Solo, "Profil de certificat d'infrastructure de clé publique X.509 et de liste de révocation de certificat (CRL) pour l'Internet", avril 2002. (*Obsolète, voir [RFC5280](#)*)
- [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.
- [RFC3513] R. Hinden et S. Deering, "[Architecture d'adressage du protocole Internet](#) version 6 (IPv6)", avril 2003. (*Obs. voir [RFC4291](#)*)
- [RFC3971] J. Arkko et autres, "[Découverte de voisin sûr](#) (SEND)", mars 2005. (*MàJ par [RFC6494](#)*) (*P.S.*)
- [UIT.X690] International Telecommunications Union, "Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, juillet 2002.

9.2 Références pour information

- [AAKMNR] Arkko, J., Aura, T., Kempf, J., Mantyla, V., Nikander, P., and M. Roe, "Securing IPv6 neighbor discovery and router discovery", ACM Workshop on Wireless Security (WiSe 2002), Atlanta, GA USA , septembre 2002.
- [Aura03] Aura, T., "Cryptographically Generated Addresses (CGA)", 6th Information Security Conference (ISC'03), Bristol, UK, octobre 2003.
- [MOV97] Menezes, A., van Oorschot, P., and S. Vanstone, "Handbook of Applied Cryptography", CRC Press, 1997.
- [MC02] Montenegro, G. and C. Castelluccia, "Statistically unique and cryptographically verifiable identifiers and addresses", ISOC Symposium on Network and Distributed System Security (NDSS 2002), San Diego, CA USA , février 2002.
- [OR01] O'Shea, G. and M. Roe, "Child-proof authentication for MIPv6 (CAM)", ACM Computer Communications Review 31(2), avril 2001.
- [RFC1750] D. Eastlake 3rd et autres, "Recommandations d'[aléa pour la sécurité](#)", décembre 1994. (*Info., remplacée par la [RFC4086](#)*)
- [RFC2461] T. Narten, E. Nordmark, W. Simpson, "[Découverte de voisins pour IP version 6](#) (IPv6)", décembre 1998. (*Obsolète, voir [RFC4861](#)*) (*D.S.*)
- [RFC2462] S. Thomson, T. Narten, "Autoconfiguration d'adresse IPv6 sans état", décembre 1998. (*Obsolète, voir [RFC4862](#)*) (*D.S.*)
- [RFC3041] T. Narten, R. Draves, "Extensions de confidentialité pour l'auto-configuration d'adresse sans état dans IPv6", janvier 2001. (*Obsolète, voir [RFC4941](#)*) (*P.S.*)
- [RFC4423] R. Moskowitz, P. Nikander, "Architecture du protocole d'identité d'hôte (HIP)", mai 2006. (*Information*)

Appendice A Exemple de génération de CGA

On génère une CGA avec Sec=1 à partir du préfixe de sous réseau fe80:: et la clé publique suivante :

```
305c 300d 0609 2a86 4886 f70d 0101 0105 0003 4b00 3048 0241 00c2 c2f1 3730 5454 f10b d9ce a368 44b5 30e9 211a 4b26
2b16 467c b7df ba1f 595c 0194 f275 be5a 4d38 6f2c 3c23 8250 8773 c786 7f9b 3b9e 63a0 9c7b c48f 7a54 ebef af02 0301
```

0001

Le modificateur est initialisé à une valeur aléatoire 89a8 a8b2 e858 d8b8 f263 3f44 d2d4 ce9a. L'entrée à Hash2 est :
89a8 a8b2 e858 d8b8 f263 3f44 d2d4 ce9a 0000 0000 0000 0000 00 305c 300d 0609 2a86 4886 f70d 0101 0105 0003 4b00
3048 0241 00c2 c2f1 3730 5454 f10b d9ce a368 44b5 30e9 211a 4b26 2b16 467c b7df ba1f 595c 0194 f275 be5a 4d38 6f2c
3c23 8250 8773 c786 7f9b 3b9e 63a0 9c7b c48f 7a54 ebef af02 0301 0001

Les 112 premiers bits de la valeur de hachage SHA-1 calculé à partir de l'entrée ci-dessus sont Hash2=436b 9a70 dbfd dbf1 926e 6e66 29c0. cela ne commence pas par les bits zéro 16*Sec=16. Donc, on doit incrémenter le modificateur de un et recalculer le hachage. La nouvelle entrée à Hash2 est :

89a8 a8b2 e858 d8b8 f263 3f44 d2d4 ce9b 0000 0000 0000 0000 00 305c 300d 0609 2a86 4886 f70d 0101 0105 0003 4b00
3048 0241 00c2 c2f1 3730 5454 f10b d9ce a368 44b5 30e9 211a 4b26 2b16 467c b7df ba1f 595c 0194 f275 be5a 4d38 6f2c
3c23 8250 8773 c786 7f9b 3b9e 63a0 9c7b c48f 7a54 ebef af02 0301 0001

La nouvelle valeur de hachage est Hash2=0000 01ca 680b 8388 8d09 12df fccc. Les 16 bits de gauche de Hash2 sont tous à zéro. Donc, on trouve un modificateur convenable. (On a eu beaucoup de chance de le trouver si tôt.)

L'entrée à Hash1 est :

89a8 a8b2 e858 d8b8 f263 3f44 d2d4 ce9b fe80 0000 0000 0000 00 305c 300d 0609 2a86 4886 f70d 0101 0105 0003 4b00
3048 0241 00c2 c2f1 3730 5454 f10b d9ce a368 44b5 30e9 211a 4b26 2b16 467c b7df ba1f 595c 0194 f275 be5a 4d38 6f2c
3c23 8250 8773 c786 7f9b 3b9e 63a0 9c7b c48f 7a54 ebef af02 0301 0001

Les 64 premiers bits de la valeur de hachage SHA-1 de l'entrée ci-dessus sont Hash1=fd4a 5bf6 ffb4 ca6c. On forme un identifiant d'interface à partir de cela en écrivant Sec=1 dans les trois bits de gauche et en réglant les bits 6 et 7 (les bits "u" et "g") à zéro. Le nouvel identifiant d'interface est 3c4a:5bf6:ffb4:ca6c.

Finalement, on forme l'adresse IPv6 fe80::3c4a:5bf6:ffb4:ca6c. C'est la nouvelle CGA. Aucune collision d'adresse n'a été détectée cette fois. (Les collisions sont très rares.) La structure de données de paramètres de CGA associée à l'adresse est la même que l'entrée à Hash1 ci-dessus.

Appendice B Remerciements

L'auteur remercie chaleureusement de leurs contributions Jari Arkko, Francis Dupont, Pasi Eronen, Christian Huitema, James Kempf, Pekka Nikander, Michael Roe, Dave Thaler, et tous les autres participants au groupe de travail SEND.

Adresse de l'auteur

Tuomas Aura
Microsoft Research
Roger Needham Building
7 JJ Thomson Avenue
Cambridge CB3 0FB
United Kingdom

téléphone : +44 1223 479708
mél : tuomaura@microsoft.com

Déclaration complète de droits de reproduction

Copyright (C) The Internet Society (2005). Tous droits réservés.

Le présent document et ses traductions peuvent être copiés et fournis aux tiers, et les travaux dérivés qui les commentent ou les expliquent ou aident à leur mise en œuvre peuvent être préparés, copiés, publiés et distribués, en tout ou partie, sans restriction d'aucune sorte, pourvu que la déclaration de droits de reproduction ci-dessus et le présent paragraphe soient inclus dans toutes telles copies et travaux dérivés. Cependant, le présent document lui-même ne peut être modifié d'aucune façon, en particulier en retirant la notice de droits de reproduction ou les références à la Internet Society ou aux autres organisations Internet, excepté autant qu'il est nécessaire pour le besoin du développement des normes Internet, auquel cas les procédures de droits de reproduction définies dans les procédures des normes Internet doivent être suivies, ou pour les besoins de la traduction dans

d'autres langues que l'anglais.

Les permissions limitées accordées ci-dessus sont perpétuelles et ne seront pas révoquées par la Internet Society ou ses successeurs ou ayant droits.

Le présent document et les informations contenues sont fournis sur une base "EN L'ÉTAT" et le contributeur, l'organisation qu'il ou elle représente ou qui le/la finance (s'il en est), la INTERNET SOCIETY et la INTERNET ENGINEERING TASK FORCE déclinent toutes garanties, exprimées ou implicites, y compris mais non limitées à toute garantie que l'utilisation des informations encloses ne violent aucun droit ou aucune garantie implicite de commercialisation ou d'aptitude à un objet particulier.

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 actuellement fourni par l'Internet Society.