Groupe de travail Réseau
Request for Comments : 4430
Catégorie : Sur la voie de la normalisation
Traduction Claude Brière de L'Isle

S. Sakane, Yokogawa Electric Corp.
K. Kamada, Yokogawa Electric Corp.
M. Thomas & J. Vilhuber, Cisco Systems
mars 2006

Négociation de clé Internet kerbérisée (KINK)

Statut du présent mémoire
Le présent document spécifie un protocole de l'Internet en cours de normalisation 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 "Protocoles officiels de l'Internet" (STD 1) pour voir l'état de 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 (2006).

Résumé
Le présent document décrit le protocole de négociation de clés Internet kerbérisées (KINK, *Kerberized Internet Negotiation of Keys*). KINK défini un protocole à faible latence, peu coûteux en calcul, facile à gérer et cryptographiquement sûr pour établir et maintenir des associations de sécurité en utilisant le système d'authentification Kerberos. KINK réutilise les charges utiles de mode rapide de l'échange de clés Internet (IKE, *Internet Key Exchange*) ce qui devrait conduire à une réutilisation substantielle des mises en œuvre existantes de IKE.

Table des matières

1. Introduction

KINK est conçu comme un mécanisme sûr et adaptable pour établir des clés entre des entités communicantes au sein d'un environnement à gestion centralisée dans lequel il est important de maintenir une politique de sécurité cohérente. Les objectifs de sécurité de KINK sont d'assurer la protection de la confidentialité, l'authentification, et la protection des messages de gestion de clés contre la répétition et d'éviter les vulnérabilités au déni de service chaque fois que possible. Les objectifs de performance du protocole sont d'avoir un faible coût de calcul, et une faible empreinte. Il sont aussi d'éviter ou de minimiser l'utilisation des opérations de clés publiques. En particulier, le protocole donne la capacité d'établir des associations de sécurité (SA) IPsec en deux messages avec un effort de calcul minimal. Ces exigences sont décrites dans la [RFC3129].

Kerberos [RFC4120 fournit un mécanisme efficace d'authentification pour les clients et serveurs qui utilisent un modèle de tiers de confiance. Kerberos fournit aussi un mécanisme d'authentification native inter domaines. Un client obtient un ticket d'un serveur d'authentification en ligne, le centre de distribution de clés (KDC, *Key Distribution Center*). Le ticket est alors utilisé pour construire un accréditif pour authentifier le client auprès du serveur. Par suite de cette opération d'authentification, le serveur va aussi partager une clé secrète avec le client. KINK utilise cette propriété comme base de la distribution des clés pour IPsec.

La gestion de clé centrale fournie par Kerberos est efficace parce que elle limite le coût de calcul et limite la complexité par rapport à la nécessité qu'a IKE d'utiliser une cryptographie à clé publique [RFC2409]. L'authentification initiale par le KDC peut être effectuée en utilisant des clés symétriques, ou asymétriques avec la cryptographie a clé publique pour l'authentification initiale dans Kerberos [RFC4556] ; cependant, des demandes ultérieures de tickets ainsi que des échanges authentifiés entre clients et serveurs utilisent toujours un chiffrement symétrique. Donc, les opérations de clé publique (s'il en est) sont limitées et sont amorties sur la durée de vie des accréditifs acquis dans l'opération initiale d'authentification auprès du KDC. Par exemple, un client peut utiliser un seul échange de clé publique avec le KDC pour établir efficacement plusieurs SA avec de nombreux autres serveurs dans le domaine du KDC. Kerberos s'adapte aussi mieux que le chiffrement direct d'homologue à homologue quand des clés symétriques sont utilisées. La raison est que comme les clés sont mémorisées dans le KDC, le nombre de clés principales est O(n+m) plutôt que O(n*m), où "n" est le nombre de clients et "m" est le nombre de serveurs.

1.2 Terminologie utilisée dans ce document

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

Le lecteur est supposé s'être familiarisé avec les termes et concepts décrits dans Kerberos version 5 [RFC4120], IPsec [RFC4301], et IKE [RFC2409].

2. Vue d'ensemble du protocole

KINK est un protocole de commande/réponse qui peut créer, supprimer, et maintenir les SA IPsec. Chaque commande ou réponse contient un en-tête commun avec un ensemble de charges utiles de Type-Longueur-Valeur. Le type d'une commande ou réponse contraint les charges utiles envoyées dans les messages de l'échange. KINK lui-même est un protocole sans état en ce que chaque commande ou réponse n'exige pas de mémorisation d'état dur pour KINK. C'est une différence avec IKE, qui utilise le mode principal pour établir d'abord une SA du protocole d'associations de sécurité et de gestion de clé Internet (ISAKMP, *Internet Security Association and Key Management Protocol*) suivie par les échanges du mode rapide.

KINK utilise les mécanismes de Kerberos pour assurer l'authentification mutuelle et la protection contre la répétition. Pour établir des SA, KINK assure la confidentialité des charges utiles qui suivent la charge utile Kerberos AP-REQ. Le concept de

KINK atténue les attaques de déni de service en exigeant des échanges authentifiés avant l'utilisation de toute opération de clé publique et l'installation de tout état. KINK donne aussi un moyen d'utiliser les mécanismes d'utilisateur à utilisateur de Kerberos quand il n'y a pas de clé partagée entre le serveur et le KDC. C'est normalement le cas, mais pas seulement, des homologues IPsec qui utilisent PKINIT pour l'authentification initiale.

KINK réutilise directement les charges utiles de mode rapide définies au paragraphe 5.5 de la [RFC2409], avec quelques changements et omissions mineurs. Dans la plupart des cas, les échanges KINK sont une seule commande et sa réponse. Un troisième message facultatif est requis à la création des SA, seulement si le répondant rejette la première proposition de l'initiateur ou veut contribuer au matériel de chiffrement. KINK assure aussi le changement de clés et la détection de l'homologue mort.

3. Flux de messages

Tous les flux de messages KINK suivent le même schéma entre les deux homologues : une commande, une réponse, et un accusé de réception facultatif dans un flux CREATE. Une commande est un message GETTGT, CREATE, DELETE, ou STATUS ; une réponse est un message REPLY ; et un accusé de réception est un message ACK.

KINK utilise Kerberos comme mécanisme d'authentification ; donc, un hôte KINK a besoin d'un ticket de service pour chaque homologue avant les négociations réelles de clés. Ceci est fondamentalement un pur échange Kerberos et le trafic de KDC réel n'est que pour l'illustrer. En pratique, quand un principal obtient divers tickets est un sujet de considération pour Kerberos et la politique locale. Avec une exception, le flux de messages GETTGT de KINK (décrit au paragraphe 3.1) est utilisé quand une authentification d'utilisateur à utilisateur est requise. Dans ce flux, on suppose que A et B ont tous des tickets distributeurs de tickets (TGT, *ticket-granting ticket*) obtenus de leurs KDC.

Après l'obtention d'un ticket de service, KINK utilise le flux de messages CREATE (paragraphe 3.2), le flux de messages DELETE (paragraphe 3.3), et le flux de messages STATUS (paragraphe 3.4) pour gérer les SA. Dans ces flux, on suppose que A a un ticket de service pour B.

3.1 Flux de messages GETTGT

Ce flux est utilisé pour récupérer un TGT auprès de l'homologue distant dans le mode d'authentification d'utilisateur à utilisateur.

Si l'initiateur détermine qu'il ne sera pas capable d'obtenir un ticket de service normal (pas d'utilisateur à utilisateur) pour le répondant, il peut essayer une authentification d'utilisateur à utilisateur. Dans ce cas, il va d'abord chercher un TGT auprès du répondant afin d'obtenir un ticket de service d'utilisateur à utilisateur :

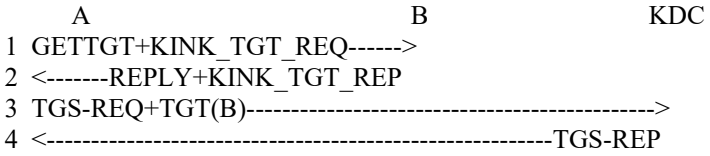


Figure 1 : Flux de messages GETTGT

L'initiateur PEUT prendre en charge les événements suivants comme déclencheurs pour aller sur le chemin d'utilisateur à utilisateur. Noter que les deux erreurs décrites ci-dessous ne vont pas être authentifiées, et la façon d'agir sur elle dépend de la politique.

- La politique locale dit que le répondant exige une authentification d'utilisateur à utilisateur.
- Une erreur KRB_AP_ERR_USER_TO_USER_REQUIRED est retournée par le répondant.
- Une erreur KDC_ERR_MUST_USE_USER2USER est retournée par le KDC.

3.2 Flux de messages CREATE

Ce flux crée les SA. La commande CREATE a une approche "optimiste" où les SA sont initialement créées dans l'attente que le répondant va choisir la charge utile initiale proposée. La proposition optimiste est placée dans la première charge utile de transformation de la première proposition. L'initiateur DOIT vérifier si la proposition optimiste a été choisie en comparant toutes les transformations et tous les attributs, qui DOIVENT être identiques à ceux de la proposition optimiste de l'initiateur à l'exception de LIFE_KILOBYTES et LIFE_SECONDS. Chacun de ces attributs PEUT être réglé à une valeur inférieure par le

répondant et s'attendre quand même au chiffrement optimiste, mais NE DOIT PAS être réglé à une valeur supérieure qui DOIT générer une erreur NO-PROPOSAL-CHOSEN. L'initiateur DOIT utiliser la durée de vie la plus courte.

Quand une commande CREATE contient un indice de paramètre de sécurité (SPI, *Security Parameter Index*) existant, le répondant DOIT la rejeter et DEVRAIT retourner une notification ISAKMP avec INVALID-SPI.

Quand une charge utile Échange de clé (KE, *key exchange*) est envoyée de l'initiateur mais que le répondant ne la prend pas en charge, le répondant DOIT la rejeter avec une notification ISAKMP de INVALID-PAYLOAD-TYPE contenant un type de charge utile KE comme données de notification. Quand l'initiateur reçoit cette erreur, il PEUT réessayer sans charge utile KE (comme toute autre transaction) si sa politique le permet.

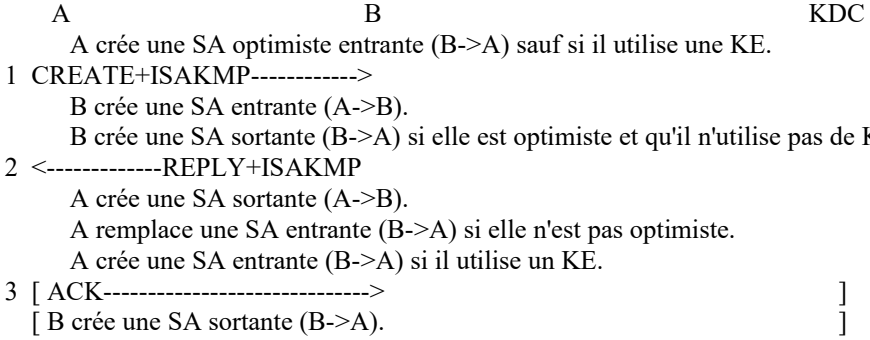


Figure 2 : Flux de message CREATE

La création des SA a deux modes : la prise de contact à deux phases et la prise de contact à trois phases. L'initiateur commence généralement une négociation en s'attendant à une prise de contact en deux phases. Quand la proposition optimiste n'est pas choisie par le répondant, la négociation passe à une prise de contact à trois phases. Quand et seulement quand l'initiateur utilise une charge utile KE, la prise de contact en trois phases est attendue depuis le début.

Les étapes suivantes sont effectuées pour une prise de contact en deux phases :
1) L'hôte A crée une SA entrante (B->A) dans sa base de données de SA en utilisant la proposition optimiste de la proposition de SA ISAKMP. Il est alors prêt à recevoir tout message de B.
2) A envoie alors le message CREATE à B.
3) Si il accepte la proposition optimiste de A, B crée une SA entrante (A->B) et une SA sortante (B->A) dans sa base de données. Si B ne choisit pas la première proposition ou veut ajouter une charge utile Nom occasionnel, on passe à l'étape 3 de la prise de contact à trois phases décrite ci-dessous.
4) B envoie alors une REPLY à A sans charge utile Nom occasionnel et sans demander de ACK.
5) À réception de la REPLY, A crée une SA sortante (A->B).

Les étapes suivantes sont effectuées pour une prise de contact en trois phases :
1) L'hôte A envoie le message CREATE à B sans créer de SA.
2) B choisit une proposition conforme à sa politique.
3) B crée une SA entrante (A->B) et envoie le choix actuel dans le REPLY. Il DEVRAIT envoyer la charge utile Nom occasionnel facultative (car cela n'augmente pas le compte de message et augmente généralement les sources d'entropie) et DOIT demander que le REPLY fasse l'objet d'un accusé de réception.
4) À réception du REPLY, A crée la SA entrante (B->A) (ou la modifie comme nécessaire, si ils sont passés de l'échange de deux phases à trois) et la SA sortante (A->B).
5) A envoie maintenant le message ACK.
6) À réception du ACK, B installe la SA sortante finale (B->A).

Si B ne choisit pas la première proposition, ajoute un nom occasionnel, ou accepte l'échange KE, il DOIT alors demander un ACK (c'est-à-dire, établir le bit ACKREQ) afin qu'il puisse installer la SA sortante finale. L'initiateur DOIT toujours générer un ACK si le bit ACKREQ est établi dans l'en-tête KINK, même si il croit que le répondant est en erreur.

3.2.1 Considérations de déduction de clé pour CREATE

L'approche optimiste de la commande CREATE permet qu'une SA soit créée en deux messages plutôt que trois. L'implication d'un échange de deux messages est que B ne va pas contribuer à la clé car A doit établir la SA entrante avant qu'il reçoive du matériel de chiffrement supplémentaire de B. Cela peut être suspect dans des circonstances normales ; cependant, KINK tire parti du fait que le KDC fournit une source fiable d'aléa qui est utilisée dans la déduction de clé. Dans de nombreux cas, cela va fournir une clé de session adéquate de sorte que B ne va pas exiger d'accusé de réception. Comme B est toujours à même de

contribuer au matériel de chiffrement, ceci est un strict compromis entre la force de la clé et le nombre de messages, que les mises en œuvre de KINK peuvent décider en fonction de leur politique.

3.3 Flux de messages DELETE

La commande DELETE supprime les SA existantes. Les charges utiles spécifiques du domaine d'interprétation (DOI) décrivent la SA à supprimer. Pour le DOI IPsec, ces charges utiles vont inclure une charge utile ISAKMP contenant la liste des SPI à supprimer.

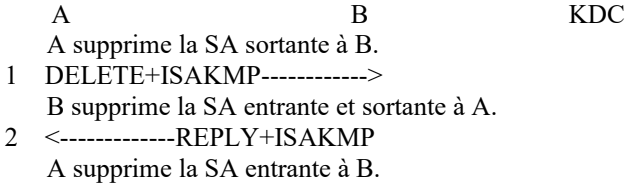


Figure 3 : Flux de messages DELETE

La commande DELETE prend une approche "pessimiste", qui ne supprime pas les SA entrantes tant qu'elle n'a pas reçu l'accusé de réception indiquant que l'autre hôte a reçu le DELETE. L'exception à l'approche pessimiste est si l'initiateur veut cesser immédiatement toute activité sur une SA entrante. Dans ce cas, il PEUT supprimer la SA entrante aussi dans l'étape 1, ci-dessus.

La charge utile ISAKMP contient une ou des charges utile Delete ISAKMP qui indiquent la ou les SA entrantes pour l'initiateur de ce flux. KINK ne permet pas de SA semi ouvertes ; donc, quand le répondant reçoit une commande DELETE, il DOIT supprimer les SA dans les deux directions, et DOIT répondre avec une ou des charges utile Delete ISAKMP qui indiquent la ou les SA entrantes pour le répondant de ce flux. Si le répondant ne peut pas trouver un SPI approprié à supprimer, il DOIT retourner une notification ISAKMP avec INVALID_SPI, qui sert aussi à informer l'initiateur qu'il peut supprimer la SA entrante.

Il existe une condition de compétition avec le flux DELETE. À cause de réarrangements dans le réseau, etc., des paquets en cours pendant que l'opération DELETE a lieu peuvent arriver après les diagrammes ci-dessus, qui recommandent de supprimer la SA entrante. Une mise en œuvre de KINK DEVRAIT appliquer un temporisateur de délai de grâce qui DEVRAIT être réglé à une durée d'au moins deux fois le délai d'aller-retour moyen, ou à une valeur configurable. Une mise en œuvre de KINK PEUT choisir de régler la période de grâce à zéro aux moments appropriés pour supprimer une SA brutalement. Le comportement décrit ici se réfère au comportement des fanions TCP [RFC0793] FIN et RST.

3.4 Flux de messages STATUS

Ce flux est utilisé pour envoyer toutes informations à un homologue ou pour obtenir toutes informations d'un homologue. Un initiateur peut envoyer une commande STATUS au répondant à tout moment, facultativement avec des charges utiles ISAKMP spécifiques du DOI. Dans le cas du DOI IPsec, elles sont généralement sous la forme de charges utiles Notification ISAKMP. Une commande STATUS est aussi utilisée comme moyen de détection d'un homologue mort décrit au paragraphe 3.7.

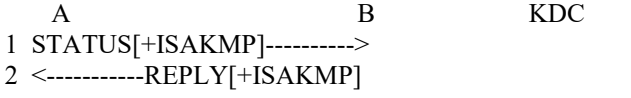


Figure 4 : Flux de messages STATUS

3.5 Rapport d'erreurs

Quand le répondant détecte une erreur dans une commande reçue, il peut envoyer une charge utile spécifique du DOI pour indiquer l'erreur dans un message REPLY. Il y a trois types de charges utiles qui peuvent indiquer des erreurs : les charges utiles KINK_KRB_ERROR pour les erreurs de Kerberos, les charges utiles KINK_ERROR pour les erreurs de KINK, et les charges utiles KINK_ISAKMP pour les erreurs de ISAKMP. Les détails sont décrits respectivement dans les paragraphes 4.2.3, 4.2.8, et 4.2.6,.

Si l'initiateur détecte une erreur dans une réponse reçue, il n'y a pas de moyen d'en faire rapport au répondant. L'initiateur DEVRAIT enregistrer l'événement et PEUT prendre une action pour y remédier en réinitialisant la commande initiale.

Si les horloges du serveur et du client sont décalées de plus que la limite de biais d'horloge déterminée par la politique (généralement 5 minutes) le serveur DOIT retourner un KRB_AP_ERR_SKEW. L'heure facultative de client dans le KRB-ERROR DEVRAIT être remplie. Si le serveur protège l'erreur en ajoutant le champ Somme de contrôle et en retournant l'heure correcte du client, le client DEVRAIT calculer la différence (en secondes) entre les deux horloges sur la base des heures de client et de serveur contenues dans le message KRB-ERROR. Le client DEVRAIT mémoriser cette différence d'horloges et l'utiliser pour ajuster son horloge dans les messages suivants. Si l'erreur n'est pas protégée, le client NE DOIT PAS utiliser la différence pour ajuster les messages suivants, parce que le faire permettrait à un attaquant de construire des authentifiants qui pourraient être utilisés pour monter des attaques en répétition.

3.6 Changement de clé des associations de sécurité

KINK attend de l'initiateur d'une SA qu'il se charge du changement de clés de la SA pour deux raisons. La première raison est pour empêcher une duplication inutile des SA par suite de collisions dues à un initiateur et un répondant essayant tous deux de renouveler une SA existante. La seconde raison est due à la nature client/serveur des échanges Kerberos, qui attend du client qu'il obtienne et maintienne les tickets. Alors que KINK s'attend à ce qu'un hôte KINK soit capable d'obtenir et maintenir les tickets, en pratique il est souvent avantageux que les serveurs attendent que les clients initient les sessions afin qu'ils n'aient pas à maintenir une grosse antémémoire de tickets.

Il n'y a pas de sémantique particulière pour les changements de clés de SA dans KINK. C'est-à-dire que afin de changer les clés d'une SA existante, l'initiateur doit créer une nouvelle SA puis soit supprimer la vieille SA avec l flux DELETE, soit la laisser arriver en fin de temporisation. Quand des sélecteurs de flux identiques sont disponibles sur des SA différentes, les mises en œuvre de KINK DEVRAIENT choisir la SA de création la plus récente. On devrait noter que KINK évite la plupart des problèmes du changement de clés de la [RFC2409] en ayant un mécanisme fiable de suppression.

Normalement, une mise en œuvre de KINK qui change les clés d'une SA existante va essayer de changer les clés de la SA avant qu'elle arrive en fin de vie, ce qui peut inclure la durée de vie complète dans time/bytecount ou le débordement du compteur de numéros de séquence. On appelle ce temps "durée de vie douce". La durée de vie douce DOIT être rendue aléatoire pour éviter la synchronisation avec des mises en œuvre similaires. Dans le cas de la durée de vie dans le temps, une approche raisonnable pour déterminer la durée de vie douce est de prendre une heure aléatoire entre T-rekey et T-retrans et de la soustraire de la durée de vie dure. Ici, T-rekey est le maximum raisonnable de la marge de changement de clés, et T-retrans est la quantité de temps que cela prendrait pour parcourir un cycle de retransmission complet. T-rekey DEVRAIT être au moins deux fois T-retrans.

3.7 Détection d'homologue mort

Afin de déterminer qu'un homologue KINK a perdu les informations de sa base de données d'information, les homologues KINK DOIVENT enregistrer l'époque courante pour laquelle ils ont des informations de SA valides pour un homologue et refléter cette époque dans chaque message AP-REQ et AP-REP. Quand un homologue KINK crée un état pour une certaine SA, il DOIT aussi enregistrer l'époque du principal. Si il découvre dans un message ultérieur que l'époque du principal a changé, il DOIT considérer toutes les SA créées par ce principal comme invalides, et les supprimer.

Alors qu'un homologue KINK DEVRAIT utiliser les retours de l'acheminement (sous la forme de messages ICMP) comme déclencheurs pour vérifier si l'homologue est ou non toujours vivant, un homologue KINK NE DOIT PAS conclure que l'homologue est mort sur la seule base d'informations d'acheminement non protégées (les dits messages ICMP).

Si on soupçonne qu'un homologue pourrait être mort (sur la base des informations disponibles à l'homologue KINK, incluant l'absence de trafic IPsec, etc.) le message KINK STATUS DEVRAIT être utilisé pour forcer l'homologue à un accusé de réception. Comme rien n'est négocié sur la détection d'homologue mort dans KINK, chaque homologue peut décider de sa propre métrique pour la "suspicion" et aussi les temporisations à utiliser avant de déclarer mort un homologue à cause du manque de réponse au message STATUS. Ceci est désirable, et ne rompt pas l'interopérabilité.

Le message STATUS a un double effet. D'abord, il provoque une réponse cryptographiquement sécurisée (et protégée contre la répétition) de la part de l'homologue, qui dit si l'homologue est accessible/vivant ou non. Ensuite, il porte le numéro d'époque de l'homologue, qui dit si l'homologue a réamorcé ou perdu tout l'état. Ceci est crucial pour le protocole KINK : dans IKE, si un homologue réamorce, il perd tout le contexte cryptographique, et aucune communication cryptographiquement sécurisée n'est possible sans une renégociation des clés. Dans KINK, à cause des tickets Kerberos, on peut communiquer de façon sûre avec un homologue, même si l'homologue a réamorcé, car la clé de chiffrement partagée utilisée est portée dans le ticket Kerberos. Donc, une communication cryptographique active n'est pas une indication que l'homologue n'a pas réamorcé et perdu tout l'état, et l'époque est nécessaire.

Supposons un homologue A qui envoie un message STATUS et un homologue B qui envoie le message REPLY (voir au paragraphe 3.4). L'homologue B PEUT supposer que l'envoyeur est vivant, et l'époque dans le message STATUS va indiquer si

l'homologue A a ou non perdu l'état. L'homologue B DOIT accuser réception du message STATUS avec un message REPLY, comme décrit au paragraphe 3.4.

Le message REPLY va indiquer à l'homologue A que l'homologue B est vivant, et l'époque dans le REPLY va indiquer si l'homologue B a ou non perdu son état. Si l'homologue A ne reçoit pas de message REPLY de l'homologue B dans un délai convenable, il PEUT envoyer un autre message STATUS. Il appartient à l'homologue A de décider avec quelle agressivité déclarer mort l'homologue B. Le niveau d'agressivité peut dépendre de nombreux facteurs comme une récupération rapide sur défaillance ou du nombre de messages envoyés par les nœud qui ont un grand nombre de SA.

Noter que l'homologue B NE DOIT PAS tirer de conclusions d'une absence de message STATUS de la part de l'homologue A. L'homologue B PEUT utiliser un message STATUS provenant de l'homologue A comme indication de la vivacité de A, mais l'homologue B NE DOIT PAS attendre un autre message STATUS à un autre moment (c'est-à-dire, la détection d'homologue mort n'est pas un mécanisme de garde en vie périodique).

Les stratégies d'envoi des messages STATUS sont les suivantes : l'homologue A peut décider d'envoyer un message STATUS seulement après une période prolongée où aucun trafic n'a été envoyé dans l'une ou l'autre direction sur les SA IPsec avec l'homologue. Une fois qu'il y a du trafic, l'homologue A peut vouloir savoir si le trafic va dans un trou noir, et envoyer un message STATUS. Autrement, l'homologue A peut utiliser un temporisateur d'inactivité pour détecter le manque de trafic avec l'homologue, et envoyer des messages STATUS dans la phase tranquille pour s'assurer que l'homologue est toujours vivant pour quand le trafic a finalement besoin d'être envoyé.

3.7.1 Traitement des homologues d'utilisateur à utilisateurs morts

Quand un initiateur utilise un ticket d'utilisateur à utilisateur et qu'un répondant a perdu son TGT précédent, le mécanisme usuel de détection d'homologue mort (DPD, *Dead Peer Detection*) ne fonctionne pas, parce que le répondant ne peut déchiffrer le ticket avec son nouveau TGT. Dans ce cas, les actions suivantes sont effectuées :

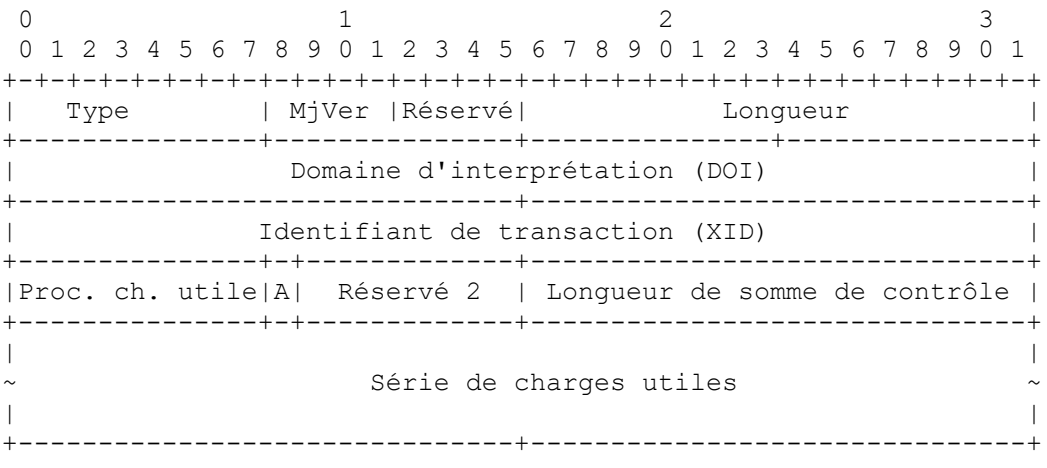
- Quand le répondant reçoit une commande KINK avec un ticket d'utilisateur à utilisateur qui ne peut pas être déchiffré avec son TGT, il retourne un REPLY avec une charge utile KINK_TGT_REP contenant le TGT.
- Quand l'initiateur reçoit un KINK_TGT_REP, il restitue un nouveau ticket de service avec le TGT et réessaye la commande.

Cela ne définit pas directement une méthode pour détecter un homologue d'utilisateur à utilisateur mort, mais de récupérer de la situation où le répondant n'a pas un TGT approprié pour déchiffrer un ticket de service envoyé par l'initiateur. Après récupération, ils peuvent échanger leurs époques, et le mécanisme de DPD usuel va détecter un homologue mort si il a été réellement mort.

L'initiateur NE DOIT PAS penser que l'homologue a été mort à réception d'un KINK_TGT_REP pour deux raisons. L'une est que le message n'est pas authentifié, et l'autre est que perdre un TGT ne signifie pas nécessairement la perte des informations de la base de données de SA. L'initiateur NE DEVRAIT PAS oublier le précédent ticket de service tant que le nouveau n'est pas bien obtenu afin de réduire le coût quand un KINK_TGT_REP falsifié est reçu.

4. Format de message KINK

Toutes les valeurs dans KINK sont formatées dans l'ordre des octets du réseau (octet de plus fort poids en premier). Les champs "Réservé" DOIVENT être réglés à zéro (0) quand un paquet est envoyé. Le receveur DOIT ignorer ces champs.



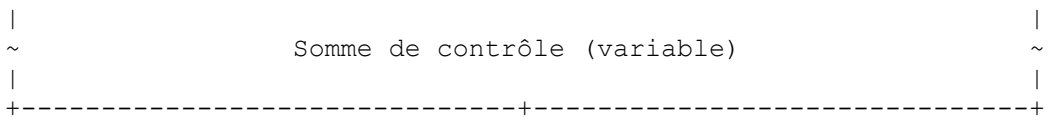


Figure 5 : Format du message KINK

Champs :

o Type (1 octet) -- type de ce message.

Valeur	Type
0	Réservé
1	CREATE
2	DELETE
3	REPLY
4	GETTGT
5	ACK
6	STATUS
7 à 127	Réservé à l'IANA
128 à 255	Utilisation privée

o MjVer (4 bits) -- Numéro de version majeure de protocole. Il DOIT être réglé à 1.

o Réservé (4 bits) -- Réservé et DOIT être zéro à l'envoi et ignoré à réception.

- Longueur (2 octets) -- Longueur du message en octets. Il n'est pas interdit dans KINK qu'il y ait des données inutiles après le message, mais le champ Longueur DOIT représenter la longueur réelle du message.

- DOI (4 octets) -- domaine d'interprétation. Tous les DOI doivent être enregistrés auprès de l'IANA dans la section Domaine d'interprétation ISAKMP du registre isakmp [ISAKMP-REG]. Le numéro alloué par l'IANA pour le DOI Sécurité IP Internet [RFC2407] est un (1). Ce champ définit le contexte de toutes les sous charges utiles dans ce message. Si les sous charges utiles ont un champ DOI (par exemple, Charge utile d'association de sécurité) le DOI dans cette sous charge utile DOIT être confronté au DOI dans cet en-tête, et les valeurs DOIVENT être les mêmes.

- XID (4 octets) -- identifiant de transaction. Une transaction KINK est liée ensemble par un identifiant de transaction, qui est créé par l'initiateur de la commande et dupliqué dans les messages suivants de la transaction. Une transaction est définie comme une commande, une réponse, et un accusé de réception facultatif. Les identifiants de transaction sont utilisés par l'initiateur pour discriminer les différentes demandes en instance avec un répondant. Ils ne sont pas utilisés pour la protection contre la répétition parce que cette fonctionnalité est fournie par Kerberos. La valeur de XID est choisie par l'initiateur et DOIT être unique sur toutes les transactions en instance. Les XID PEUVENT être construits en utilisant un compteur à accroissement monotone ou un générateur de nombres aléatoires.

- Prochaine charge utile (1 octet) -- Indique le type de la première charge utile après l'en-tête de message.

- A, ou ACKREQ (1 bit) -- demande d'accusé de réception. Réglé à un si le répondant exige un accusé de réception explicite qu'un message REPLY a été reçu. Un initiateur NE DOIT PAS établir ce fanion, et un répondant ne devrait pas l'établir sauf pour un REPLY à un CREATE quand la proposition optimiste est choisie.

- Réservé 2 (7 bits) -- Réservé et DOIT être zéro à l'envoi, DOIT être ignoré à réception.

- Longueur de somme de contrôle (2 octets) -- longueur en octets de la somme de contrôle cryptographique du message. Une longueur de somme de contrôle de zéro implique que le message n'est pas authentifié.

- Somme de contrôle (variable) -- somme de contrôle Kerberos chiffrée sur le message entier à l'exclusion du champ Somme de contrôle lui-même. Quand des octets de bourrage sont exigés entre la dernière charge utile et le champ Somme de contrôle, ils DOIVENT être inclus dans le calcul. Ce champ DOIT toujours être présent chaque fois que une clé est disponible via une charge utile AP-REQ ou AP-REP. La clé utilisée DOIT être la clé de session dans le ticket. Quand une clé n'est pas disponible, ce champ est absent, et le champ Longueur de somme de contrôle est mis à zéro. Le contenu de ce champ est le résultat de la fonction Kerberos 5 get_mic [RFC3961]. La fonction get_mic utilisée est spécifiée par un type de somme de contrôle, qui est un "mécanisme de somme de contrôle exigé" du "etype" pour la clé de session Kerberos dans le ticket Kerberos. Si le type de somme de contrôle n'est pas un algorithme de chiffrement, le message DOIT être rejeté.

Pour calculer la somme de contrôle, le champ Longueur de somme de contrôle est mis à zéro et le champ Longueur est rempli avec la longueur totale du paquet sans la somme de contrôle. Ensuite, le paquet est passé à la fonction get_mic et son résultat est ajouté au paquet. Tout bourrage KINK après le champ Somme de contrôle est interdit, sauf le bourrage interne de Kerberos, qui peut être inclus dans le résultat de la fonction get_mic. Finalement, le champ Longueur de somme de contrôle est rempli avec la longueur de la somme de contrôle et le champ Longueur est rempli avec la longueur totale du paquet y compris la somme de contrôle.

Pour vérifier la somme de contrôle, une longueur sans somme de contrôle est calculée à partir de la valeur du champ Longueur, en soustrayant la longueur de somme de contrôle. Le champ Longueur est rempli avec la valeur de la longueur sans somme de contrôle et le champ Longueur de somme de contrôle est mis à zéro. Ensuite, le paquet sans la somme de contrôle (décalage de 0 à longueur sans somme de contrôle moins 1 du paquet reçu) et la somme de contrôle (décalage de longueur sans somme de contrôle à la fin) sont passés à la fonction verify_mic. Si la vérification échoue, le message DOIT être éliminé.

L'en-tête KINK est suivi immédiatement par une série de champs Type/Longueur/Valeur, définis au paragraphe 4.2.

4.1 Règles d'alignement de KINK

KINK a les règles suivantes concernant l'alignement et le bourrage :

- Tous les champs de longueur DOIVENT refléter le nombre d'octets réels dans la structure; c'est-à-dire, ils ne comptent pas les octets de bourrage exigés par les alignements de KINK.
- Les champs d'en-têtes, de charges utiles, et de somme de contrôle KINK DOIVENT être alignés sur des limites de 4 octets.
- Les champs de longueur variable (sauf le champ Somme de contrôle) DOIVENT toujours commencer immédiatement après le dernier octet du champ précédent. C'est-à-dire, ils ne sont pas alignés sur des limites de 4 octets.

4.2 Charges utiles KINK

Immédiatement à la suite de l'en-tête se trouve une liste de charges utiles de Type/Longueur/Valeur (TLV). Il peut y en avoir un nombre quelconque. Chaque charge utile DOIT commencer par un en-tête de charge utile. Chaque en-tête de charge utile est construit sur l'en-tête générique de charge utile. Toutes les données suivent immédiatement l'en-tête générique. Les charges utiles sont toutes implicitement alignées sur une limite de 4 octets, bien que le champ Longueur de charge utile DOIVE refléter avec précision le nombre réel d'octets dans la charge utile.

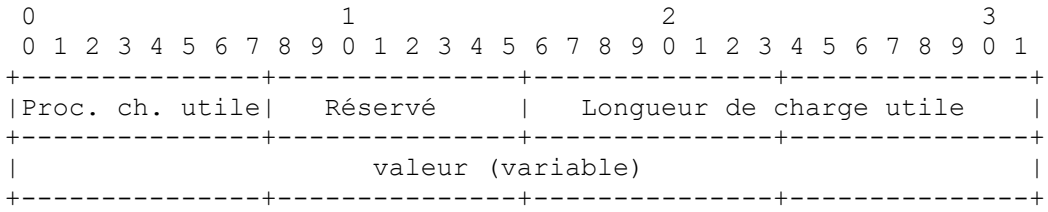


Figure 6 : Format d'une charge utile KINK

Champs :

- Prochaine charge utile (1 octet) -- type de la prochaine charge utile.

Valeur	Prochaine charge utile
0	KINK_DONE
1	KINK_AP_REQ
2	KINK_AP_REP
3	KINK_KRB_ERROR
4	KINK_TGT_REQ
5	KINK_TGT_REP
6	KINK_ISAKMP
7	KINK_ENCRYPT
8	KINK_ERROR
9 à 127	Réservé à l'IANA
128 à 255	Utilisation privée

Le type de prochaine charge utile KINK_DONE note que la charge utile en cours est la charge utile finale du message.

- Réservé (1 octet) -- DOIT être réglé à zéro à l'envoi, DOIT être ignoré à réception.

- o Longueur de charge utile (2 octets) -- longueur de cette charge utile, incluant les champs Type et Longueur.
- o Valeur (variable) -- cette valeur du champ dépend du type.

4.2.1 Charge utile KINK_AP_REQ

La charge utile KINK_AP_REQ relaye une AP-REQ Kerberos au répondant. La AP-REQ DOIT demander l'authentification mutuelle.

Le présent document ne spécifie pas comment générer le nom de principal. C'est-à-dire que les noms complets de principal peuvent être mémorisés dans la politique locale, Les noms de domaine pleinement qualifiés (FQDN, *Fully Qualified Domain Name*) peuvent être convertis en noms de principal, les adresses IP peuvent être converties en noms de principal par des services de noms sûrs, etc., mais voir le premier paragraphe de la Section "Considérations sur la sécurité".

Si le nom de principal de l'homologue pour le service KINK est généré à partir d'un FQDN, le nom de principal, par lequel commence l'initiateur, va être "kink/fqdn@REALM"; où "kink" est une chaîne littérale pour le service IPsec de KINK, "fqdn" est le nom de domaine pleinement qualifié de l'hôte du service, et "REALM" est le domaine Kerberos du service. Un nom de principal est sensible à la casse, et la partie "fqdn" DOIT être en minuscules, comme décrit dans la [RFC4120].

Le champ Valeur de cette charge utile a le format suivant :

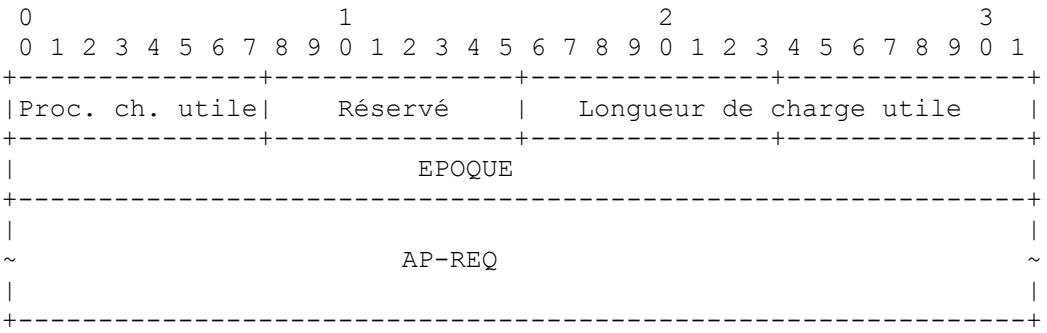


Figure 7 : Charge utile KINK_AP_REQ

Champs :

- o Prochaine charge utile, Réservé, Longueur de charge utile sont définis au début de cette section.
- o EPOQUE -- temps absolu auquel le créateur de la AP-REQ a des informations de SA valides. Normalement, c'est quand l'automate de chiffrement KINK a commencé si il ne conserve pas les informations de SA à travers les redémarrages. La valeur dans ce champ est les 4 octets de moindre poids de l'heure POSIX, qui sont les secondes écoulées (mais sans compter les secondes sautées) depuis UTC 1970-01-01T00:00:00. Par exemple, l'UTC 2038-01-19T03:14:07 est représenté par 0x7fffffff.
- o AP-REQ -- le champ Valeur de cette charge utile contient une AP-REQ Kerberos brute.

4.2.2 Charge utile KINK_AP_REP

La charge utile KINK_AP_REP relaye une AP-REP Kerberos à l'initiateur. La fraîcheur de la AP-REP DOIT être vérifiée comme décrit dans la [RFC4120].
Le champ Valeur de cette charge utile a le format suivant :

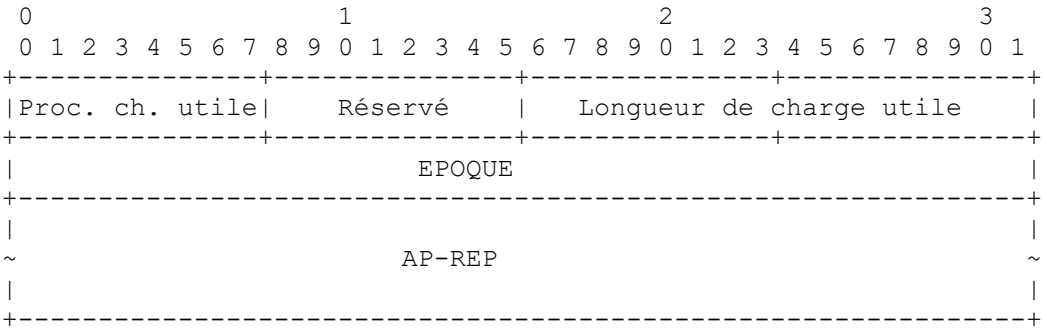


Figure 8 : Charge utile KINK_AP_REP

Champs :
- Prochaine charge utile, Réservé, Longueur de charge utile sont définis au début de cette section.
- EPOQUE -- temps absolu auquel le créateur de la AP-REP a des informations de SA valides. Normalement, c'est quand l'automate de chiffrement KINK a commencé si il ne conserve pas les informations de SA à travers les redémarrages. La valeur dans ce champ est les 4 octets de moindre poids de l'heure POSIX, qui sont les secondes écoulées (mais sans compter les secondes sautées) depuis UTC 1970-01-01T00:00:00. Par exemple, l'UTC 2038-01-19T03:14:07 est représenté par 0x7fffffff.
- AP-REP -- le champ Valeur de cette charge utile contient une AP-REP Kerberos brute.

4.2.3 Charge utile KINK_KRB_ERROR

La charge utile KINK_KRB_ERROR relaye les erreurs de type Kerberos à l'initiateur. L'initiateur DOIT être prêt à recevoir tout type d'erreur Kerberos valide [RFC4120].

Les mises en œuvre de KINK DEVRAIENT utiliser le champ Somme de contrôle KINK pour retourner KINK_KRB_ERROR quand la clé de service appropriée est disponible. En particulier dans le cas d'erreurs de biais d'horloge, protéger l'erreur au serveur crée une meilleure expérience d'utilisateur parce que cela n'exige pas que les horloges soient synchronisées. Cependant, de nombreuses mises en œuvre de Kerberos ne rendent pas facile l'obtention de la clé de session pour protéger les paquets en erreur. Pour les erreurs Kerberos non authentifiées, l'initiateur PEUT choisir d'agir sur elles, mais DEVRAIT prendre des précautions contre les attaques "make-work".

Noter que KINK n'utilise pas le champ Texte ou e_data du message d'erreur Kerberos, bien qu'une mise en œuvre conforme de KINK DOIVE être prête à les recevoir et PEUT les enregistrer.

Le champ Valeur de cette charge utile a le format suivant :

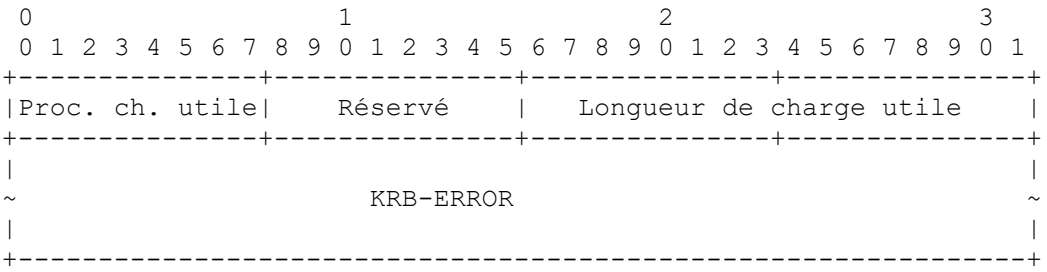


Figure 9 : Charge utile KINK_KRB_ERROR

Champs :
- Prochaine charge utile, Réservé, Longueur de charge utile sont définis au début de cette section.
- KRB-ERROR -- le champ Valeur de cette charge utile contient une KRB-ERROR Kerberos brute.

4.2.4 Charge utile KINK_TGT_REQ

La charge utile KINK_TGT_REQ donne le moyen d'obtenir un TGT de l'homologue afin d'obtenir un ticket de service d'utilisateur à utilisateur du KDC.

Le champ Valeur de cette charge utile a le format suivant :

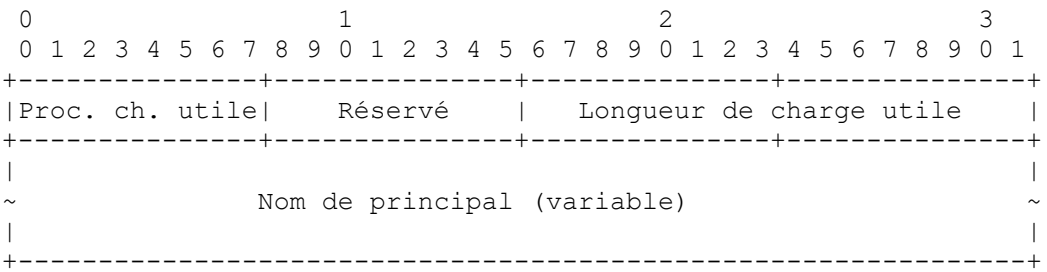


Figure 10 : Charge utile KINK_TGT_REQ

Champs :
- o Prochaine charge utile, Réservé, Longueur de charge utile sont définis au début de cette section.
- o Nom de principal -- nom de principal avec lequel l'initiateur veut communiquer. On suppose que l'initiateur connaît le nom de principal du répondant (incluant le nom de domaine) de la même façon que dans le cas non d'utilisateur à utilisateur. Le TGT retourné NE DOIT PAS être un TGT inter domaines, son "cname" et son "crealm" DOIVENT correspondre au nom de principal demandé, afin que l'initiateur puisse se retrouver avec le répondant au domaine du répondant.

Les valeurs de nom de principal sont des représentations en chaîne d'octet d'un nom de principal et de domaine formatées juste comme la chaîne d'octets utilisée dans le composant "NAME" du jeton de nom exporté d'une interface de programme d'application de service de sécurité générique (GSS-API, *Generic Security Service Application Program Interface*) [RFC2743] pour le mécanisme GSS-API de Kerberos v5 [RFC1964]. Voir au paragraphe 2.1.3 de la [RFC1964].

Si le répondant n'est pas le principal demandé et n'est pas capable d'obtenir un TGT pour le nom, il PEUT retourner un KRB_AP_ERR_NOT_US. Si la politique administrative interdit de retourner un TGT, il PEUT retourner un KINK_U2UDENIED.

4.2.5 Charge utile KINK_TGT_REP

Le champ Valeur de cette charge utile contient le TGT demandé dans une charge utile KINK_TGT_REQ précédente d'une commande GETTGT.

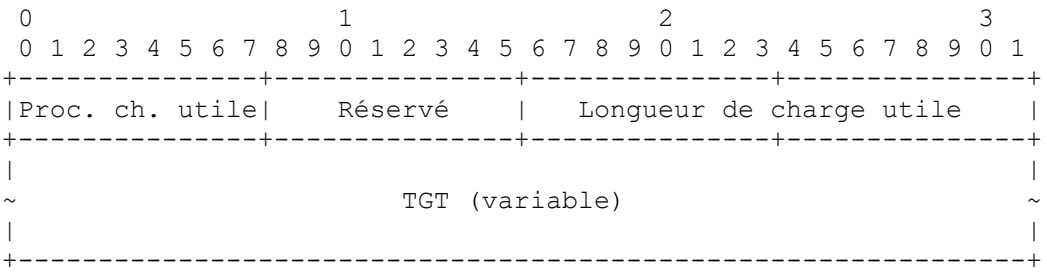


Figure 11 : Charge utile KINK_TGT_REP

Champs :
- o Prochaine charge utile, Réservé, Longueur de charge utile sont définis au début de cette section.
- o TGT -- TGT du répondant codé selon les règles distinctives de codage (DER).

4.2.6 Charge utile KINK_ISAKMP

Le champ Valeur de cette charge utile a le format suivant :

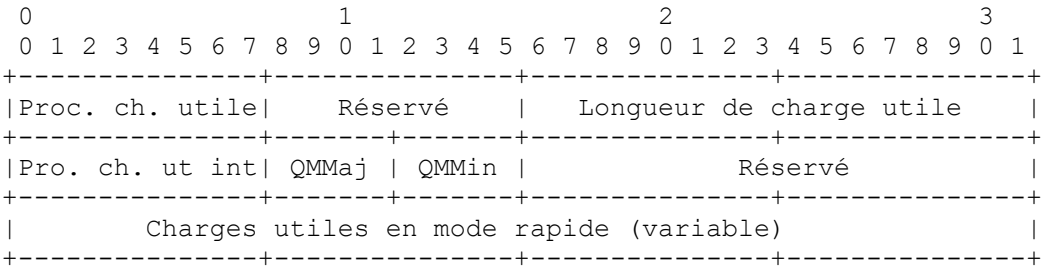


Figure 12 : Charge utile KINK_ISAKMP

Champs :
- o Prochaine charge utile, Réservé, Longueur de charge utile sont définis au début de cette section.
- o Prochaine charge utile interne -- type de la première charge utile de la série interne de charges utiles ISAKMP.
- o QMMaj -- version majeure des charges utiles internes. DOIT être réglé à 1.
- o QMMin -- version mineure des charges utiles internes. DOIT être réglé à 0.

La charge utile KINK_ISAKMP encapsule les charges utiles de mode rapide IKE (phase 2) pour prendre l'action appropriée selon la commande KINK. Il peut y avoir un nombre quelconque de charges utiles KINK_ISAKMP au sein d'un seul message KINK. Bien que la [RFC2409] soit un peu confuse quant à la question de savoir si plusieurs SA différentes peuvent être créées

avec un seul message IKE, KINK exige explicitement qu'un nouvel en-tête ISAKMP soit utilisé pour chaque opération discrète de SA. En d'autres termes, une mise en œuvre de KINK NE DOIT PAS envoyer plusieurs transactions en mode rapide au sein d'une seule charge utile KINK_ISAKMP.

L'objet de la version de mode rapide est de permettre la rétro compatibilité avec IKE et ISAKMP si il y a des révisions ultérieures. Pour l'instant, les versions majeures et mineures de mode rapide sont réglées respectivement à un et zéro (1.0). Ces versions ne correspondent pas à la version ISAKMP dans l'en-tête ISAKMP. Une mise en œuvre conforme de KINK DOIT accepter la réception de charges utiles 1.0. Elle PEUT accepter les versions suivantes (en envoi et en réception) et DEVRAIT fournir un moyen de revenir à la version de mode rapide 1.0 si l'homologue KINK est incapable de traiter les versions futures. Une mise en œuvre conforme de KINK NE DOIT PAS mêler des versions de mode rapide dans une transaction.

4.2.7 Charge utile KINK_ENCRYPT

La charge utile KINK_ENCRYPT encapsule d'autres charges utiles KINK et est chiffrée en utilisant la clé de session et l'algorithme spécifiés par son "etype". Cette charge utile DOIT être la charge utile finale dans la chaîne de charges utiles externes du message. La charge utile KINK_ENCRYPT DOIT être chiffrée avant l'application de la somme de contrôle KINK finale.

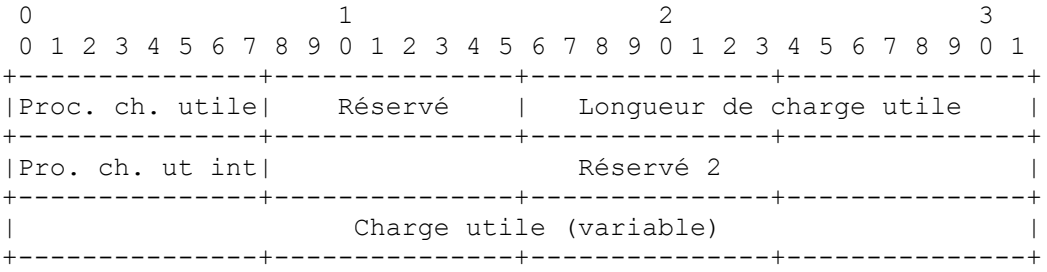


Figure 13 : Charge utile KINK_ENCRYPT

Champs :
- o Prochaine charge utile, Réservé, Longueur de charge utile sont définis au début de cette section. Cette charge utile est la dernière dans un message, et en conséquence, le champ Prochaine charge utile doit être KINK_DONE (0).
- o Prochaine charge utile interne -- type de la première charge utile de la série interne des charges utiles KINK chiffrées.
- o Réservé 2 -- Réservé. DOIT être zéro à l'envoi et ignoré à réception.

La couverture des données chiffrées commence à InnerNextPload *(prochaine charge utile interne)* de sorte que le type de la première charge utile reste confidentiel. Donc, le nombre d'octets chiffrés est Longueur de charge utile - 4.
Le format de la charge utile de chiffrement suit la sémantique normale de Kerberos. Son contenu est le résultat de la fonction de chiffrement définie dans la section Profil d'algorithme de chiffrement de la [RFC3961]. Les paramètres tels que la fonction de chiffrement elle-même, la clé spécifique, et l'état initial sont définis avec le "etype". La fonction de chiffrement peut avoir un bourrage en elle-même et il peut y avoir des données parasites à la fin du texte source déchiffré. Une mise en œuvre de KINK DOIT être prête à ignorer un tel bourrage après la dernière sous charge utile à l'intérieur de la charge utile KINK_ENCRYPT. Noter que chaque fonction de chiffrement a son propre mécanisme de protection. Il est redondant avec la somme de contrôle dans l'en-tête KINK, mais c'est inévitable parce que il n'est pas toujours possible de retirer la partie protection de l'intégrité de la fonction de chiffrement.

4.2.8 Charge utile KINK_ERROR

Le type de charge utile KINK_ERROR donne un mécanisme de niveau protocole pour retourner une condition d'erreur. Cette charge utile ne devrait pas être utilisée pour les erreurs générées par Kerberos ou les erreurs spécifiques du DOI qui ont leurs propres charges utiles définies. Le code d'erreur est dans l'ordre des octets du réseau.

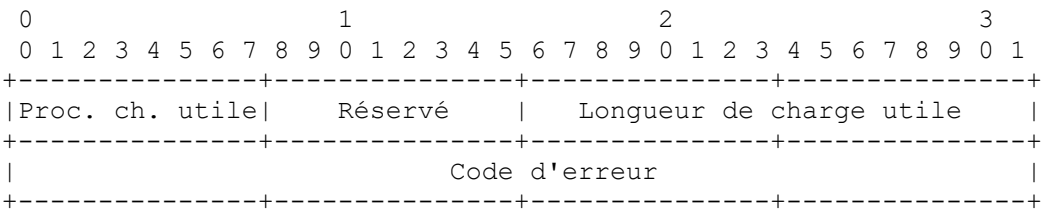


Figure 14 : Charge utile KINK_ERROR

Champs :
- o Prochaine charge utile, Réservé, Longueur de charge utile sont définis au début de cette section.
- o Code d'erreur : une des valeurs suivantes dans l'ordre des octets du réseau :

Valeur	Code d'erreur	Objet
0	KINK_OK	Pas d'erreur détectée
1	KINK_PROTOERR	Message mal formé
2	KINK_INVDOI	DOI invalide
3	KINK_INVMAJ	Version majeure invalide
4	Réservé	
5	KINK_INTERR	Erreur interne irrécupérable
6	KINK_BADQMVERS	Version de mode rapide non acceptée
7	KINK_U2UDENIED	Retourner un TGT est interdit
8 à 8191	Réservé à l'IANA	
8192 à 16383	Utilisation privée	
16384 -	Réservé	

Le répondant NE DOIT PAS retourner KINK_OK. Quand il est reçu, l'initiateur PEUT agir comme si la charge utile KINK_ERROR spécifique n'était pas présente. Si l'initiateur prend en charge plusieurs versions de mode rapide ou de DOI, KINK_BADQMVERS ou KINK_INVDOI est reçu, et la somme de contrôle est vérifiée, ensuite il PEUT réessayer avec une autre version ou DOI. Un répondant DEVRAIT retourner une erreur KINK avec KINK_INVMAJ, quand il reçoit un numéro de version KINK non acceptée dans l'en-tête. Quand KINK_U2UDENIED est reçu, l'initiateur PEUT réessayer avec le mode non d'utilisateur à utilisateur (si il n'a pas encore été essayé).

En général, le répondant PEUT choisir de retourner ces erreurs dans des réponses à des commandes non authentifiées, mais DEVRAIT veiller à éviter d'être impliqué dans des attaques de déni de service. De même, l'initiateur PEUT choisir d'agir sur des erreurs non authentifiées, mais DEVRAIT veiller à éviter des attaques de déni de service.

5. Différences avec le mode rapide IKE

KINK utilise directement les charges utiles ISAKMP pour négocier les SA. En particulier, KINK utilise les types de charge utile IKE phase 2 (autrement dit, de mode rapide). En général, très peu de changements devraient être nécessaires à une mise en œuvre IKE pour établir les SA, et sauf si il y a une mention contraire dans ce document, toutes les capacités et exigences de la [RFC2409] DOIVENT être prises en charge. Les charges utiles IKE phase 1 NE DOIVENT PAS être envoyées.

À la différence de IKE, KINK définit des commandes spécifiques pour la création, la suppression, et l'état des SA, principalement pour faciliter la création/suppression prévisible de SA (voir les paragraphes 3.2 et 3.3). À ce titre, KINK fait certaines restrictions sur les charges utiles qui peuvent être envoyées avec certaines commandes, et certaines restrictions supplémentaires sur la sémantique des charges utiles. Les mises en œuvre devraient se référer à la [RFC2409] et la [RFC2408] pour les formats et la sémantique actuels. Si une charge utile IKE phase 2 particulière n'est pas mentionnée ici, cela signifie qu'il n'y a pas de différence dans son utilisation.

- o L'en-tête Charge utile d'association de sécurité pour IP est défini au paragraphe 4.6.1 de la [RFC2407]. Pour le présent mémoire, le domaine d'interprétation DOIT être réglé à 1 (IPsec) et le codage de tramage de situation DOIT être réglé à 1 (SIT_IDENTITY_ONLY). Tous les autres champs sont omis (parce que SIT_IDENTITY_ONLY est établi).

- o KINK étend aussi la sémantique de IKE en ce qu'il définit une proposition optimiste pour les commandes CREATE pour permettre que la création de SA se fasse en deux messages.

- o Le mode rapide IKE (phase 2) utilise l'algorithme de hachage utilisé en mode principal (phase 1) pour générer le matériel de chiffrement. À cette fin, KINK DOIT utiliser une fonction pseudo aléatoire déterminée par le "etype" de la clé de session.

- o KINK n'utilise pas du tout la charge utile HASH.

- o KINK permet que le numéro de charge utile Nom occasionnel soit facultatif pour faciliter le chiffrement optimiste.

5.1 Charges utiles d'association de sécurité

KINK prend en charge les attributs de SA suivants de la [RFC2407] :

Valeur	Classe	Type
1	Type de vie de SA	B
2	Durée de vie de SA	V
4	Mode d'encapsulation	B
5	Algorithme d'authentification	B
6	Longueur de clé	B
7	Tours de clé	B

Voir dans la [RFC2407] les définitions réelles de ces attributs.

5.2 Charges utiles Proposition et Transformation

KINK utilise directement les charges utiles Proposition et Transformation sans différence. Cependant, KINK, met un accent particulier sur la première proposition et la première transformation de chaque conjugué pour un chiffrement optimiste.

5.3 Charges utiles d'identification

La charge utile Identification porte des informations qui sont utilisées pour identifier le trafic qui est à protéger par la SA qui va être établie. KINK restreint les types d'identifiants, qui sont définis au paragraphe 4.6.2.1 de la [RFC2407], aux valeurs suivantes :

Valeur	Type d'identifiant
1	ID_IPV4_ADDR
4	ID_IPV4_ADDR_SUBNET
5	ID_IPV6_ADDR
6	ID_IPV6_ADDR_SUBNET
7	ID_IPV4_ADDR_RANGE
8	ID_IPV6_ADDR_RANGE

5.4 Charges utiles de nom occasionnel

La charge utile Nom occasionnel contient des données aléatoires qui DOIVENT être utilisées dans la génération de clé. Elle DOIT être envoyée par l'homologue KINK initiateur, et PEUT être envoyée par l'homologue KINK répondant. Voir à la Section 7 la discussion de son utilisation dans la génération de clé.

5.5 Charges utiles Notify

Les charges utiles Notify sont utilisées pour transmettre plusieurs données d'information, comme des conditions d'erreur et des transitions d'état à un homologue. Par exemple, les informations de notification transmises peuvent être des messages d'erreur qui spécifient pourquoi une SA n'a pas pu être établie. Elles peuvent aussi être des données d'état qu'un processus qui gère une base de données de SA souhaite communiquer à un processus homologue.

Les types dans la gamme 0 à 16383 sont destinés à rapporter des erreurs [RFC2408]. Une mise en œuvre qui reçoit un type dans cette gamme qu'il ne reconnaît pas dans une réponse DOIT supposer que la demande correspondante a entièrement échoué. Les types d'erreur non reconnus dans une demande et les types d'état dans une demande ou une réponse DOIVENT être ignorés, et ils DEVRAIENT être enregistrés. Les charges utiles Notify avec des types d'état PEUVENT être ajoutées à tout message et DOIVENT être ignorées si elles ne sont pas reconnues. Elles sont destinées à indiquer des capacités, et au titre de la négociation de SA sont utilisées pour négocier des paramètres non cryptographiques.

Le tableau ci-dessous fait la liste des messages de notification et de leur valeur correspondante. PAYLOAD-MALFORMED note des types d'erreur définis dans la [RFC2408]. Donc INVALID-PROTOCOL-ID, par exemple, n'est pas utilisé dans le présent document. INVALID-MAJOR-VERSION et INVALID-MINOR-VERSION ne sont pas utilisés parce que KINK_BADQMVERS est utilisé pour dire à l'initiateur que la version de IKE n'est pas prise en charge.

Valeur	Message Notify - Types d'erreur	Commentaire
1	INVALID-PAYLOAD-TYPE	Envoyé si le type de charge utile ISAKMP n'est pas reconnu. Il est aussi envoyé quand la charge utile KE n'est pas prise en charge par le répondant. Les données de notification DOIVENT contenir le type de charge utile d'un octet.
11	INVALID-SPI	Envoyé si le répondant a un SPI indiqué par l'initiateur en cas de flux CREATE, ou si le répondant n'a pas un SPI indiqué par l'initiateur en cas de flux DELETE.

14	NO-PROPOSAL-CHOSEN	Envoyé si aucune des propositions dans la charge utile SA n'est acceptable.
16	PAYLOAD-MALFORMED	Envoyé si la charge utile KINK_ISAKMP reçue est invalide à cause d'un type, longueur, ou valeur hors gamme. Il est aussi envoyé quand la demande est rejetée pour une raison qui ne correspond pas à d'autres types d'erreur.

5.6 Charges utiles Delete

KINK utilise directement les charges utiles ISAKMP Delete sans changement.

5.7 Charges utiles KE

IKE exige que le secret parfait vers l'avant (PFS) soit pris en charge par l'utilisation de la charge utile KER. KINK conserve la capacité d'utiliser le PFS, mais relâche l'exigence de DOIT mettre en œuvre à DEVRAIT mettre en œuvre. Les raisons sont décrites dans la Section "Considérations sur la sécurité".

6. Construction et contraintes de message pour le DOI IPsec

Toutes les commandes, réponses, et accusés de réception sont liées ensemble par le champ XID de l'en-tête de message. Le XID est normalement un champ à accroissement monotone, et est utilisé par l'initiateur pour différencier les demandes en instances chez un répondant. Le champ XID ne fournit pas de protection contre la répétition car cette fonctionnalité est fournie par les mécanismes de Kerberos. De plus, les commandes et réponses DOIVENT utiliser une somme de contrôle cryptographique sur le message entier si les deux homologues partagent une clé via un échange de ticket.

Dans tous les cas de cette Section, si un message contient une charge utile KINK_AP_REQ ou KINK_AP_REP, d'autres charges utiles KINK PEUVENT être encapsulées dans une charge utile KINK_ENCRYPT.

6.1 Message REPLY

Le message REPLY est une réponse générique qui DOIT contenir une charge utile KINK_AP_REP, KINK_KRB_ERROR, ou KINK_ERROR. Les messages REPLY PEUVENT contenir des charges utiles spécifiques du DOI telles que des charges utiles ISAKMP qui sont définies dans les paragraphes suivants.

6.2 Message ACK

Les ACK ne sont envoyés que quand le bit ACKREQ est établi dans un message REPLY. Un message ACK DOIT contenir une charge utile AP-REQ et pas d'autre charge utile.

6.3 Message CREATE

Ce message initie un établissement d'une ou de nouvelles associations de sécurité. Le message CREATE doit contenir une charge utile AP-REQ et toutes charges utiles spécifiques du DOI.

```
En-tête CREATE KINK
   KINK_AP_REQ
   [KINK_ENCRYPT]
     charges utiles KINK_ISAKMP
        charge utile SA
           charges utiles de proposition
                 charges utiles de transformation
         charge utile Nom occasionnel (Ni)
        [KE]
        [IDci, IDcr]
        [charges utiles de notification]
```

Les réponses sont de la forme suivante :

```
En-tête REPLY KINK
   KINK_AP_REP
   [KINK_ENCRYPT]
      charges utiles KINK_ISAKMP
```

charge utile SA
charges utiles de proposition
charges utiles de transformation
[charge utile Nom occasionnel (Nr)]
[KE]
[IDci, IDcr]
[charges utiles de notification]

Noter qu'il DOIT y avoir au moins une seule charge utile Proposition et une seule charge utile Transformation dans les messages REPLY. Il ne va y avoir plusieurs charges utiles Proposition que lorsque un bouquet de SA est négocié. Aussi, à la différence de IKE, la charge utile Nom occasionnel Nr n'est pas exigée, et si elle existe, un accusé de réception doit être demandé pour indiquer que les SA sortantes de l'initiateur doivent être modifiées. Si aucune des premières propositions n'est choisie par le receveur, il DEVRAIT inclure la charge utile Nom occasionnel.

KINK, comme IKE, permet la création de nombreuses SA dans une commande Create. Si aucune des propositions optimistes n'est choisie par le répondant, il DOIT demander un ACK.

Si une erreur spécifique du DOI IPsec est rencontrée, le répondant doit répondre avec une charge utile Notify décrivant l'erreur :

En-tête REPLY KINK
KINK_AP_REP
[KINK_ENCRYPT]
[KINK_ERROR]
charges utiles KINK_ISAKMP
[charges utiles de notification]

Si le répondant trouve une erreur Kerberos pour laquelle il peut produire un authentifiant valide, le REPLY prend la forme suivante :

En-tête REPLY KINK
KINK_AP_REP
[KINK_ENCRYPT]
KINK_KRB_ERROR

Finalement, si le répondant trouve un type d'erreur Kerberos ou KINK pour lequel il ne peut pas créer une AP-REP, il DOIT répondre avec une seule charge utile KINK_KRB_ERROR ou KINK_ERROR :

En-tête REPLY KINK
[KINK_KRB_ERROR]
[KINK_ERROR]

6.4 Message DELETE

Ce message indique que l'homologue envoyeur a supprimé ou va bientôt supprimer la ou les associations de sécurité avec l'autre homologue.

En-tête DELETE KINK
KINK_AP_REQ
[KINK_ENCRYPT]
charges utiles KINK_ISAKMP
charges utiles Delete
[charges utiles de notification]

Il y a trois formes de réponses pour un DELETE. La forme normale est :

En-tête REPLY KINK
KINK_AP_REP
[KINK_ENCRYPT]
[KINK_ERROR]
charges utiles KINK_ISAKMP
charges utiles Delete

[charges utiles de notification]

Si une erreur spécifique du DOI IPsec est rencontrée, le répondant doit répondre avec une charge utile Notify décrivant l'erreur :

En-tête REPLY KINK
KINK_AP_REP
[KINK_ENCRYPT]
[KINK_ERROR]
charges utiles KINK_ISAKMP
[charges utiles de notification]

Si le répondant trouve une erreur Kerberos pour laquelle il peut produire un authentifiant valide, le REPLY prend la forme suivante :

En-tête REPLY KINK
KINK_AP_REP
[KINK_ENCRYPT]
KINK_KRB_ERROR

Si le répondant trouve un type d'erreur KINK ou Kerberos, il DOIT répondre avec une seule charge utile KINK_KRB_ERROR ou KINK_ERROR :

En-tête REPLY KINK
[KINK_KRB_ERROR]
[KINK_ERROR]

6.5 Message STATUS

La commande STATUS est utilisée de deux façons :
1) comme moyen de relayer un message ISAKMP Notification,
2) comme moyen de vérifier si un homologue a changé son époque pour la détection d'homologue mort.

STATUS contient les charges utiles suivantes :
En-tête KINK
KINK_AP_REQ
[[KINK_ENCRYPT]
charge utile KINK_ISAKMP
[charges utiles de notification]

Il y a trois formes de réponses pour un STATUS. La forme normale est :

En-tête REPLY KINK
KINK_AP_REP
[[KINK_ENCRYPT]
[KINK_ERROR]
charge utile KINK_ISAKMP
[charges utiles de notification]

Si le répondant trouve une erreur Kerberos pour laquelle il peut produire un authentifiant valide, le REPLY prend la forme suivante :

En-tête REPLY KINK
KINK_AP_REP
[KINK_ENCRYPT]
KINK_KRB_ERROR

Si le répondant trouve un type d'erreur KINK ou Kerberos, il DOIT répondre avec une seule charge utile KINK_KRB_ERROR ou KINK_ERROR :

En-tête REPLY KINK
[KINK_KRB_ERROR]

[KINK_ERROR]

6.6 Message GETTGT

Une commande GETTGT n'est utilisée que pour porter un TGT Kerberos et est sans rapport avec la gestion de SA ; donc, elle contient seulement une charge utile KINK_TGT_REQ et ne contient aucune charge utile spécifique de DOI.

Il y a deux formes de réponses pour une GETTGT. Dans la forme normale, où le répondant est autorisé à retourner son TGT, le REPLY contint la charge utile KINK_TGT_REP. Si le répondant n'est pas autorisé à retourner son TGT, il DOIT répondre avec une charge utile KINK_ERROR.

7. Déduction de clé ISAKMP

KINK utilise les mêmes mécanismes de déduction de clé que défini au paragraphe 5.5 de la [RFC2409], qui est :

KEYMAT = prf(SKEYID_d, [g(qm)^xy |] protocole | SPI | Ni_b [| Nr_b])

Les différences suivantes s'appliquent :

- o prf est la fonction pseudo aléatoire correspondant au "etype" de clé de session. Elle est définie dans la [RFC3961].
- o SKEYID_d est la clé de session dans le ticket de service Kerberos provenant de AP-REQ. Noter que des sous clés ne sont pas utilisées dans KINK et DOIVENT être ignorées si on en reçoit.
- o Ni_b et Nr_b font tous deux partie des charges utiles Nom occasionnel (Ni et Nr, respectivement) comme décrit au paragraphe 3.2 de la [RFC2409]. Nr_b est facultatif, ce qui signifie que Nr_b est traité comme si une valeur de longueur zéro était fournie quand le nom occasionnel du répondant (Nr) n'existe pas. Quand Nr existe, Nr_b DOIT être inclus dans le calcul.

Noter que g(qm)^xy se réfère au matériel de chiffrement généré quand les charges utiles KE sont fournies en utilisant l'accord de clés Diffie-Hellman. Ceci est expliqué au paragraphe 5.5 de la [RFC2409].

Le reste de la déduction de clé (par exemple, comment étendre KEYMAT) suit IKE. Il appartient à chaque service de décrire comment utiliser les matériaux de chiffrement déduits (par exemple, le paragraphe 4.5.2 de la [RFC4301]).

8. Numéros d'usage de clé pour la déduction de clé Kerberos

Les fonctions de Kerberos encrypt/decrypt et get_mic/verify_mic exigent des "numéros d'usage de clé". Ils sont utilisés pour générer des clés spécifiques pour les opérations de chiffrement de sorte que des clés différentes sont utilisées pour les objets différents. KINK utilise deux numéros d'usage, indiqués ci-dessous.

Numéro d'usage	Objet
39	charge utile KINK_ENCRYPT (pour le chiffrement)
40	Champ de somme de contrôle (pour la somme de contrôle)

9. Considérations de transport

KINK utilise UDP sur l'accès 910 pour transporter ses messages. C'est un temporisateur T qui DEVRAIT prendre en considération les questions de temps d'aller-retour et DOIT mettre en œuvre un mécanisme de retard exponentiel tronqué. L'automate à états est simple : tout message qui attend une réponse DOIT retransmettre la demande en utilisant le temporisateur T. Comme Kerberos exige que les messages soient retransmis avec de nouvelles heures pour la protection contre la répétition, le message DOIT être recréé chaque fois en incluant la somme de contrôle du message. Les commandes et les réponses avec le bit ACKREQ établi sont gardées sur les temporisateurs de retransmission. Quand un initiateur KINK reçoit un REPLY avec le bit ACKREQ établi, il DOIT conserver la capacité de régénérer le message ACK pour la transaction pendant un minimum de son cycle complet de temporisation de retransmission ou jusqu'à ce qu'il remarque que les paquets sont arrivés sur la nouvelle SA construite, selon ce qui se produit en premier.

Quand un homologue KINK retransmet un message, il DOIT créer un nouvel authentifiant Kerberos pour la AP-REQ afin que l'homologue puisse différencier les répétitions et les paquets abandonnés. Il en résulte un potentiel de condition de compétition quand une retransmission se produit avant qu'une répétition en vol soit reçue/traitée. Pour contrer cette condition de répétition, celui qui retransmet DEVRAIT tenir une liste des authentifiants valides qui sont en instance pour toute transaction.

Quand un homologue KINK retransmet une commande, il DOIT utiliser le même ticket au sein des retransmissions. C'est pour éviter les conditions de compétition en utilisant des clés différentes, qui résultent en des KEYMAT différents entre un initiateur et un répondant. Pour cette raison, (1) un initiateur DOIT obtenir un ticket dont la durée de vie est supérieure au temps de transaction maximum de l'initiateur, incluant les fins de temporisation, ou (2) il DOIT continuer d'utiliser le même ticket dans un ensemble de retransmissions, et si il reçoit une erreur (très vraisemblablement KRB_AP_ERR_TKT_EXPIRED) du répondant, il commence une nouvelle transaction avec un nouveau ticket.

10. Considérations pour la sécurité

Les noms de principaux sont les identités des services KINK, mais le trafic protégé par les SA est identifié par des sélecteurs spécifiques du DOI (adresses IP, numéros d'accès, etc.). Cela peut conduire à ce que des données protégées par les SA soient écartées de l'authentification. Par exemple, si deux hôtes différents revendiquent la même adresse IP, il peut être impossible de prédire la clé de quel principal protège les données. Donc, une mise en œuvre doit veiller au lien entre les noms de principal et les sélecteurs de SA.

L'envoi d'erreurs sans protection cryptographique doit être traitée avec un grand soin. Il y a un compromis entre vouloir aider au diagnostic d'un problème et vouloir éviter d'être la dupe d'une attaque de déni de service.

KINK met bout à bout et réutilise de nombreuses parties de Kerberos et IKE, ce dernier à son tour est pavé de morceaux de nombreux autres documents. À ce titre, KINK hérite des nombreuses faiblesses et problèmes de chacun de ses composants. Cependant, KINK utilise seulement les charges utiles IKE phase 2 pour créer et supprimer les SA ; les considérations de sécurité qui relèvent de IKE phase 1 peut être ignorées en toute sécurité. Cependant, être capable d'ignorer la phase d'authentification de IKE signifie nécessairement que KINK hérite de tous les problèmes de sécurité de l'authentification Kerberos comme mentionné dans la [RFC4120]. Pour un, un KDC, comme un serveur d'authentification, autorisation, et comptabilité (AAA) est un point d'attaque et tout ce que cela implique. Beaucoup a été écrit sur les diverses insuffisances et faiblesses de Kerberos, et elles devraient être évaluées pour tout déploiement.

L'utilisation de Kerberos par KINK présente deux problèmes. D'abord, KINK attend explicitement que le KDC fournisse une entropie adéquate quand il génère les clés de session. Ensuite, Kerberos est utilisé comme un protocole d'authentification d'utilisateur avec la possibilité d'attaques de dictionnaire sur les mots de passe de l'utilisateur. Le présent mémoire ne décrit pas de méthode particulière pour éviter ces pièges, mais recommande que des clés générées avec un aléa convenable devraient être utilisées pour les principaux de service comme d'utiliser l'option "-randomkey" avec la commande "kadmin addprinc" de MIT ainsi que pour les clients quand c'est praticable.

Kerberos ne fournit pas actuellement de secret parfait vers l'avant en général. KINK avec la charge utile KE peut fournir le PFS pour une clé de service à partir d'une clé Kerberos, mais le KE n'est pas obligatoire à cause de son coût de calcul. C'est un compromis et les opérateurs peuvent choisir le PFS plutôt que le coût, et vice versa. KINK lui-même devrait être sécurisé contre l'analyse hors ligne à partir de mots de passe de principaux compromis si le PFS est utilisé, mais du point de vue du système global, l'existence d'autres services Kerbérisés qui ne fournissent pas le PFS rend cette situation sous optimale.

11. Considérations relatives à l'IANA

L'IANA a alloué un numéro d'accès bien connu pour KINK.
L'IANA a créé un nouveau registre pour les paramètres KINK, et a enregistré les identifiants suivants :
Types de message KINK (section 4)
Types de prochaine charge utile KINK (paragraphe 4.2)
Codes d'erreur KINK (paragraphe 4.2.8)

Les changements et ajouts à ce registre suivent les politiques décrites ci-dessous. Leur signification est décrite dans la [RFC2434].

- L'utilisation des numéros dans la gamme "Utilisation privée" est Utilisation privée.
- L'allocation dans la gamme "Réservé à l'IANA" doit être par action de normalisation, ou des RFC qui ne sont pas sur la voie de la norlaisation avec Revue d'expert. (Mais la spécification complète peut être un document public et permanent d'un organe de normalisation autre que l'IETF, une RFC y faisant référance est nécessaire.)
- Les autres changements requièrent une action de normalisation.

12. Considérations de compatibilité

KINK peut s'accommoder de futures versions de mode rapide par l'utilisation du champ Version dans la charge utile ISAKMP ainsi que de nouveaux domaines d'interprétation. Dans le présent mémoire, la seule version de mode rapide prise en charge est 1.0, qui correspond à la [RFC2409]. De même, le seul DOI pris en charge est le domaine d'interprétation IPsec [RFC2407]. Les nouvelles versions de mode rapide et de DOI DOIVENT être décrites dans des documents à venir.

Les mises en œuvre de KINK DOIVENT rejeter les versions de ISAKMP supérieures à la plus haute version acutellement prise en charge avec un type d'erreur KINK_BADQMVERS. Une mise en œuvre de KINK qui reçoit un message KINK_BADQMVERS DEVRAIT être capable de revenir à la version 1.0.

12.1 Nouvelles versions de mode rapide

Le groupe de travail IPsec définit le protocole de prochaine génération de IKE [RFC4306], qui n'utilise pas le mode rapide, mais est similaire à celui de IKEv1. La différence entre les deux est résumée dans l'Appendice A de la [RFC4306]. Chacun d'eux doit être pris en compte afin d'utiliser IKEv2 avec KINK.

12.2 Nouveau DOI

L'en-tête de message KINK contient un champ appelé "Domaine d'interprétation (DOI)" pour permettre à d'autres domaines d'interprétation d'utiliser KINK comme mécanisme de transport sûr pour le chiffrement.

Comme exemple de nouveau DOI, le groupe de travail MSEC a défini le domaine d'interprétation de groupe (GDOI, *Group Domain of Interpretation*) [RFC3547], qui définit quelques nouveaux messages, qui ressemblent aux messages ISAKMP, mais ne sont pas définis dans ISAKMP.

Afin de porter les messages GDOI dans KINK, le champ DOI dans l'en-tête KINK va indiquer que GDOI est utilisé, au lieu de IPSEC-DOI, et la charge utile KINK_ISAKMP va contenir les charges utiles définies dans le document GDOI plutôt que les charges utiles utilisées par le mode rapide de la [RFC2409]. Le numéro de version dans l'en-tête KINK_ISAKMP se rapporte au DOI dans l'en-tête KINK, de sorte qu'une version majeure/mineure de 1.0 sous le DOI GDOI est différente d'une version majeure/mineure de 1.0 sous le DOI IPSEC-DOI.

13. Travaux connexes

Le groupe de travail IPsec a défini un certain nombre de protocoles qui donnent la capacité de créer et maintenir des SA cryptographiquement sûres à la couche trois (c'est-à-dire, la couche IP). Cet effort a produit deux protocoles distincts :

- un mécanisme pour chiffrer et authentifier les charges utiles de datagrammes IP qui supposent un secret partagé entre envoyeur et receveur,
- un mécanisme pour que les homologues IPsec effectuent l'authentification mutuelle et l'échange du matériel de chiffrement.

Le groupe de travail IPsec a défini un mécanisme d'homologue à homologue d'authentification et de chiffrement, IKE (RFC2409). Un des inconvénients d'un protocole d'homologue à homologue est que chaque homologue doit connaître et mettre en œuvre une politique de sécurité de site, qui en pratique peut être assez complexe. De plus, la nature d'homologue à homologue de IKE exige l'utilisation de Diffie-Hellman (DH) pour établir un secret partagé. Malheureusement, DH est assez coûteux en calcul et enclin aux attaques de déni de service. IKE s'appuie aussi sur les certificats X.509 pour réaliser une authentification adaptable des homologues. Les signatures numériques sont aussi coûteuses en calcul, et les modèles de confiance fondés sur le certificat sont difficiles à déployer en pratique. Alors que IKE permet bien une clé pré partagée, la distribution de clé est requise entre tous les homologues -- un problème O(n^2) -- qui est problématique pour les grands déploiements.

14. Remerciements

Nombreux sont ceux qui ont contribué à l'effort KINK, incluant nos présidents de groupe de travail Derek Atkins et Jonathan Trostle. L'inspiration originale est venu de l'effort PacketCable de CableLab's, qui a défini une version simplifiée de IPsec kerbérisé, incluant Sasha Medvinsky, Mike Froh, Matt Hur et David McGrew. L'inspiration pour réutiliser pleinement IKE phase 2 est le résultat du document de Tero Kivinen qui suggère de greffer l'authentification Kerberos sur le mode rapide.

15. Références

15.1 Références normatives

[ISAKMP-REG] IANA, "Internet Security Association and Key Management Protocol (ISAKMP) Identifiers", <http://www.iana.org/assignments/isakmp-registry>.

[RFC1964] J. Linn, "Mécanisme GSS-API de Kerberos version 5", juin 1996. (*MàJ par* RFC4121 et RFC6649)

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

[RFC2407] D. Piper, "Le domaine d'interprétation de sécurité IP de l'Internet pour ISAKMP", novembre 1998. *(Obs., voir* 4306)

[RFC2408] D. Maughan, M. Schertler, M. Schneider et J. Turner, "Protocole Internet d'association de sécurité et gestion de clés (ISAKMP)", novembre 1998. *(Obsolète, voir la* RFC4306)

[RFC2409] D. Harkins et D. Carrel, "L'échange de clés Internet (IKE)", novembre 1998. *(Obsolète, voir la* RFC4306)

[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)

[RFC3961] K. Raeburn, "Spécifications de chiffrement et de somme de contrôle pour Kerberos 5", février 2005. *(MàJ par* 8429

[RFC4120] C. Neuman et autres, "Service Kerberos d'authentification de réseau (V5)", juillet 2005. (*MàJ par* RFC4537, 5021, 6649, 7751, 8062, 8129, 8429)

[RFC4301] S. Kent et K. Seo, "Architecture de sécurité pour le protocole Internet", décembre 2005. *(P.S.) (Remplace la* RFC2401)

15.2 Références pour information

[RFC0793] J. Postel (éd.), "Protocole de commande de transmission – Spécification du protocole du programme Internet DARPA", STD 7, septembre 1981.

[RFC2743] J. Linn, "Interface générique de programme d'application de service de sécurité, version 2, mise à jour 1", janvier 2000. (*MàJ par* RFC5554)

[RFC3129] M. Thomas, "Exigences pour la négociation de clés Internet Kerbérisées", juin 2001. *(Information)*

[RFC3547] M. Baugher, B. Weis, T. Hardjono et H. Harney, "Le domaine d'interprétation de groupe", juillet 2003. *(Obsolète, voir la* RFC6407)

[RFC4306] C. Kaufman, "Protocole d'échange de clés sur Internet (IKEv2)", décembre 2005. *(Obsolète, voir la* RFC5996)

[RFC4556] L. Zhu, B. Tung, "Cryptographie à clé publique pour authentification initiale dans Kerberos (PKINIT)", juin 2006. *(P.S. ; MàJ par* RFC8062)

Adresse des auteurs

Shoichi Sakane
Yokogawa Electric Corporation
2-9-32 Nakacho, Musashino-shi,
Tokyo 180-8750 Japan
mél : Shouichi.Sakane@jp.yokogawa.com

Ken'ichi Kamada
Yokogawa Electric Corporation
2-9-32 Nakacho, Musashino-shi,
Tokyo 180-8750 Japan
mél : Ken-ichi.Kamada@jp.yokogawa.com

Michael Thomas
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
mél : mat@cisco.com

Jan Vilhuber
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
mél : vilhuber@cisco.com

Déclaration de droits de reproduction

Copyright (C) The Internet Society (2006).

Le présent document est soumis aux droits, licences et restrictions contenus dans le BCP 78, et à www.rfc-editor.org, et sauf pour ce qui est mentionné ci-après, les auteurs conservent tous leurs droits.

Le présent document et les informations qui y sont 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 ci-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 pourraient 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 faits 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 Internet Society.