Groupe de travail Réseau
Request for Comments : 4306
RFC rendues obsolètes : 2407, 2408, 2409
Rendue obsolète par la RFC5996
Catégorie : En cours de normalisation

C. Kaufman, éd., Microsoft
décembre 2005

Traduction Claude Brière de L'Isle

Protocole d'échange de clés sur Internet (IKEv2)

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

Résumé

Le présent document décrit la version 2 du protocole d'échange de clés sur Internet (IKE, *Internet Key Exchange*). IKE est une composante d'IPsec utilisée pour effectuer l'authentification mutuelle et établir et entretenir les associations de sécurité (SA).

Cette version de la spécification IKE combine les contenus de documents précédemment distincts, incluant l'association de sécurité Internet et le protocole de gestion de clés (ISAKMP, RFC 2408), IKE (RFC 2409), le domaine Internet d'interprétation (DOI, RFC 2407), la traversée de traducteur d'adresse réseau (NAT), l'authentification traditionnelle, et l'acquisition d'adresse distante.

La version 2 de IKE n'interfonctionne pas avec la version 1, mais le format d'en-tête commun aux deux versions amène de façon non ambiguë sur le même port UDP.

Table des matières

1 Introduction....2
1.1 Scénarios d'utilisation....3
1.2 Les échanges initiaux....4
1.3 L'échange CREATE_CHILD_SA....6
1.4 L'échange INFORMATIONAL....7
1.5 Messages d'information en-dehors d'une IKE_SA....7
2 Détails du protocole IKE et variantes....8
2.1 Utilisations des temporisateurs de retransmission....8
2.2 Utilisation des numéros de séquence pour l'identifiant de message....8
2.3 Taille de fenêtre pour les demandes en chevauchement....9
2.4 Synchronisation d'état et fin de temporisation de connexion....9
2.5 Numéros de version et rétro compatibilité....10
2.6 Témoins....11
2.7 Négociation d'algorithme cryptographique....12
2.8 Renouvellement des clés....13
2.9 Négociation de sélecteur de trafic....14
2.10 Noms occasionnels....15
2.11 Adresse et agilité d'accès....15
2.12 Réutilisation des exponentielles de Diffie-Hellman....16
2.13 Génération du matériel de clés....16
2.14 Génération du matériel de clés pour la IKE_SA....17
2.15 Authentification de la IKE_SA....17
2.16 Méthodes de protocole d'authentification extensible....18
2.17 Génération du matériel de clés pour les CHILD_SA....19
2.18 Renouvellement des clés des IKE_SA en utilisant l'échange CREATE_CHILD_SA....19

2.19 Demande d'une adresse interne sur un réseau distant....20
2.20 Demande de la version de l'homologue....21
2.21 Traitement des erreurs....21
2.22 IPComp....21
2.23 Traversée de NAT....22
2.24 Notification explicite d'encombrement (ECN)....23
3 Formats d'en-tête et de charge utile....24
3.1 L'en-tête IKE....24
3.2 En-tête générique de charge utile....25
3.3 Charge utile d'association de sécurité....26
3.4 Charge utile d'échange de clés....32
3.5 Charges utiles d'identification....33
3.6 Charge utile de certificat....34
3.7 Charge utile de demande de certificat....35
3.8 Charge utile d'authentification....36
3.9 Charge utile de nom occasionnel....37
3.10 Charge utile de Notification....37
3.13 Charge utile de sélecteur de trafic....43
3.14 Charge utile chiffrée....44
3.15 Charge utile de configuration....46
3.16 Charge utile de protocole d'authentification extensible (EAP)....48
4 Exigences de conformité....49
5 Considérations sur la sécurité....50
6 Considérations relatives à l'IANA....52
7 Remerciements....52
8 Références....53
8.1 Références normatives....53
8.2 Références informatives....53
Appendice A : Résumé des changements depuis IKEv1....54
Appendice B : Groupes Diffie-Hellman....55
B.1 Groupe 1 – MODP à 768 bits....55
B.2 Groupe 2 – MODP à 1024 bits....55

1 Introduction

La sécurité sur IP (IPsec) fournit la confidentialité, l'intégrité des données, le contrôle d'accès, et l'authentification de la source des données aux datagrammes IP. Ces services sont fournis en maintenant un état partagé entre la source et le collecteur d'un datagramme IP. Cet état définit, entre autres choses, les services spécifiques fournis au datagramme, quels algorithmes cryptographiques seront utilisés pour fournir les services, et les clés utilisées comme entrées des algorithmes cryptographiques.

Établir cet état partagé de façon manuelle ne convient pas très bien. Il est donc nécessaire d'avoir un protocole pour établir cet état de façon dynamique. Le présent mémoire décrit un tel protocole – l'échange de clés sur Internet (IKE, *Internet Key Exchange*). Ceci est la version 2 de IKE. La version 1 de IKE était définie dans les RFC 2407, 2408, et 2409 [Pip98, MSST98, HC98]. Cet unique document est destiné à remplacer ces trois RFC.

Les définitions des principaux termes de ce document (comme Association de sécurité ou SA) se trouvent dans la [RFC4301].

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRAIT", "NE DEVRAIT PAS" et "PEUT" qui apparaissent dans le présent document sont à interpréter comme décrit dans la [RFC2119].

Le terme"Révision d'experts" est à interpréter comme défini dans la [RFC2434].

IKE effectue l'authentification mutuelle entre deux parties et établit une association de sécurité (SA) IKE qui inclut les informations de secret partagé qui peuvent être utilisées pour établir efficacement des SA pour l'encapsulation de charge utile de sécurité (ESP) [RFC4303] et/ou l'en-tête d'authentification (AH) [RFC4302] et un ensemble d'algorithmes cryptographiques utilisés par les SA pour protéger le trafic qu'elles portent. Dans le présent document, le terme "suite" ou "suite cryptographique" se réfère à un ensemble complet d'algorithmes utilisés pour protéger une SA. Un initiateur propose une ou plusieurs suites en faisant la liste des algorithmes pris en charge qui peuvent être combinés en suites selon un choix de combinaisons. IKE peut aussi négocier l'utilisation de la compression IP (IPComp, *IP Compression*) [RFC3173] en connexion avec une SA ESP et/ou AH. On appelle SA IKE une "IKE_SA". Les SA pour ESP et/ou AH qui sont établies au moyen de cette IKE_SA sont appelées "CHILD_SA" *(SA filles)*.

Toutes les communications IKE consistent en paires de messages : une demande et une réponse. La paire est appelée un "échange". On appelle les premiers messages qui établissent une IKE_SA des échanges IKE_SA_INIT et IKE_AUTH et

les échanges IKE suivants les échanges CREATE_CHILD_SA ou INFORMATIONAL. Dans le cas le plus courant, il y a un seul échange IKE_SA_INIT et un seul échange IKE_AUTH (un total de quatre messages) pour établir la IKE_SA et la première CHILD_SA. Dans des cas exceptionnels, il peut y avoir plus qu'un de chacun de ces échanges. Dans tous les cas, tous les échanges IKE_SA_INIT DOIVENT s'achever avant tout autre type d'échange, puis tous les échanges IKE_AUTH DOIVENT s'achever, et ensuite un nombre quelconque d'échanges CREATE_CHILD_SA et INFORMATIONAL peut survenir dans n'importe quel ordre. Dans certains scénarios, une seule CHILD_SA est nécessaire entre les points d'extrémité IPsec, et donc, il n'y aura pas d'échanges supplémentaires. Des échanges ultérieurs PEUVENT être utilisés pour établir des CHILD_SA supplémentaires entre la même paire de points d'extrémité authentifiés et pour effectuer d'entretien.
Le flux de messages IKE consiste toujours en une demande suivie par une réponse. Il est de la responsabilité du demandeur de s'assurer de leur fiabilité. Si la réponse n'est pas reçue dans un certain intervalle de temporisation, le demandeur doit réémettre la demande (ou abandonner la connexion).
La première demande/réponse d'une session IKE (IKE_SA_INIT) négocie les paramètres de sécurité pour la IKE_SA, envoie les noms occasionnels *(nonce)*, et envoie les valeurs Diffie-Hellman.
La seconde demande/réponse (IKE_AUTH) transmet les identités, prouve sa connaissance des secrets correspondants aux deux identités, et établit une SA pour la première (et souvent seule) CHILD_SA AH et/ou ESP.
Les types d'échanges ultérieurs sont CREATE_CHILD_SA (qui crée une CHILD_SA) et INFORMATIONAL (qui supprime une SA, rapporte des conditions d'erreur, ou effectue d'autres tâches d'entretien). Chaque demande exige une réponse. Une demande INFORMATIONAL sans charge utile (autre que la charge utile vide Encrypted *(chiffrée)* exigée par la syntaxe) est communément utilisée comme un contrôle d'existence. Ces échanges ultérieurs ne peuvent pas être utilisés avant l'achèvement des échanges initiaux.
Dans la description qui suit, on suppose qu'aucune erreur ne survient. Les modifications au flux qui surviennent en cas d'erreurs sont décrites au paragraphe 2.21.

1.1 Scénarios d'utilisation

IKE est prévu pour la négociation des SA ESP et/ou AH dans un certain nombre de scénarios différents, dont chacun a ses propres exigences particulières.

1.1.1 Tunnel de passerelle de sécurité à passerelle de sécurité

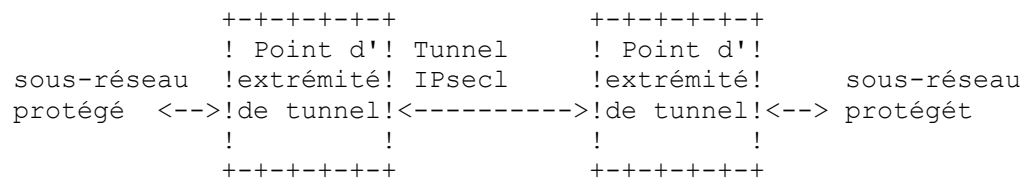


Figure 1 : Tunnel de passerelle de sécurité à passerelle de sécurité

Dans ce scénario, aucun point d'extrémité de la connexion IP ne met en œuvre IPsec, mais les nœuds de réseau entre eux protègent le trafic sur une partie du chemin. La protection est transparente pour les points d'extrémité, et dépend de l'acheminement ordinaire pour envoyer les paquets à travers les points d'extrémité du tunnel pour le traitement. Chaque point d'extrémité annoncera l'ensemble des adresses "derrière" lui, et les paquets seront envoyés en mode tunnel là où l'en-tête IP interne contient les adresses IP des points d'extrémité réels.

1.1.2 Transport de point d'extrémité à point d'extrémité

Figure 2 : Point d'extrémité à point d'extrémité

Dans ce scénario, les deux points d'extrémité de la connexion IP mettent en œuvre IPsec, comme il est exigé des hôtes dans la [RFC4301]. Le mode transport sera habituellement utilisé sans en-tête IP interne. Si il y a un en-tête IP interne, les adresses internes seront les mêmes que les adresses externes. Une seule paire d'adresses sera négociée pour les paquets à

protéger par cette SA. Ces points d'extrémité PEUVENT mettent en œuvre des contrôles d'accès de couche application sur la base des identités authentifiées IPsec des participants. Ce scénario permet la sécurité de bout en bout qui a été un principe de base de l'Internet depuis les [RFC1958] [RFC2775], et une méthode pour limiter les problèmes inhérents à la complexité dans les réseaux notés par la [RFC3439]. Bien que ce scénario puisse n'être pas entièrement applicable à l'IPv4 Internet, il a été développé avec succès dans des scénarios spécifiques au sein d'intranets utilisant IKEv1. Il devrait être plus largement activé durant la transition vers IPv6 et avec l'adoption de IKEv2.

Il est possible dans ce scénario qu'un des points d'extrémité protégés ou les deux se trouve derrière un nœud de traduction d'adresse réseau (NAT, *network address translation*), auquel cas les paquets tunnelés devront être encapsulés en UDP de sorte que les numéros d'accès dans les en-têtes UDP puissent être utilisés pour identifier les points d'extrémité individuels "derrière" le NAT (voir au paragraphe 2.23).

1.1.3 Tunnel de point d'extrémité à passerelle de sécurité

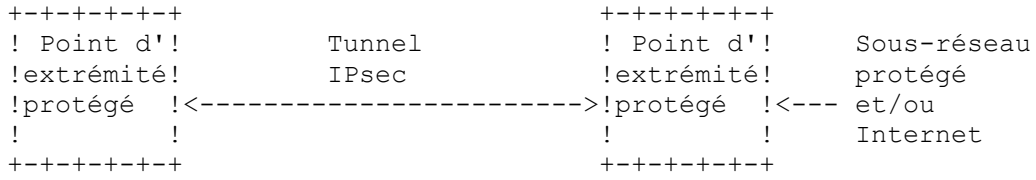


Figure 3 : Tunnel de point d'extrémité à passerelle de sécurité

Dans ce scénario, un point d'extrémité protégé (normalement un ordinateur portable en déplacement) se connecte à son réseau d'entreprise au moyen d'un tunnel protégé par IPsec. Il ne peut utiliser ce tunnel que pour accéder à des informations sur le réseau d'entreprise, ou il peut tunneler tout son trafic sur le réseau d'entreprise afin de tirer parti de la protection fournie par un pare-feu d'entreprise contre les attaques fondées sur l'Internet. Dans tous les cas, le point d'extrémité protégé voudra une adresse IP associée à la passerelle de sécurité de sorte que les paquets qui lui sont retournés aillent à la passerelle de sécurité et lui soient tunnelés en retour. Cette adresse IP peut être statique ou peut être allouée de façon dynamique par la passerelle de sécurité. À l'appui de ce dernier cas, IKEv2 inclut un mécanisme pour que l'initiateur demande une adresse IP possédée par la passerelle de sécurité à utiliser pour la durée de sa SA.

Dans ce scénario, les paquets vont utiliser le mode tunnel. Sur chaque paquet provenant du point d'extrémité protégé, l'en-tête IP externe contiendra l'adresse IP de source associée à sa localisation actuelle (c'est-à-dire, l'adresse qui obtiendra que le trafic soit acheminé directement au point d'extrémité) alors que l'en-tête IP interne contiendra l'adresse IP de source allouée par la passerelle de sécurité (c'est-à-dire, l'adresse qui obtiendra le trafic acheminé à la passerelle de sécurité pour transmission au point d'extrémité). L'adresse de destination externe sera toujours celle de la passerelle de sécurité, alors que l'adresse de destination interne sera celle de la destination ultime du paquet.

Dans ce scénario, il est possible que le point d'extrémité protégé soit derrière un NAT. Dans ce cas, l'adresse IP telle que vue par la passerelle de sécurité ne sera pas la même que l'adresse IP envoyée par le point d'extrémité protégé, et les paquets devront être encapsulés dans UDP afin d'être acheminés correctement.

1.1.4 Autres scénarios

D'autres scénarios sont possibles, comme le sont des combinaisons incorporées de ceux ci-dessus. Un exemple notable combine les aspects des paragraphes 1.1.1 et 1.1.3. Un sous-réseau peut faire tous les accès externes à travers une passerelle de sécurité distante en utilisant un tunnel IPsec, avec les adresses sur le sous-réseau acheminées à la passerelle de sécurité par le reste de l'Internet. Un exemple pourrait être celui du réseau de rattachement de quelqu'un qui serait virtuellement sur l'Internet avec des adresses IP statiques bien que la connectivité soit fournie par un ISP qui alloue une seule adresse IP de façon dynamique à la passerelle de sécurité de l'utilisateur (les adresses IP statiques et un relais IPsec sont fournis par un tiers localisé ailleurs).

1.2 Les échanges initiaux

Les communications utilisant IKE commencent toujours par les échanges IKE_SA_INIT et IKE_AUTH (appelés Phase 1 dans IKEv1). Ces échanges initiaux consistent normalement en quatre messages, bien que dans certains scénarios ce nombre puisse être supérieur. Toutes les communications utilisant IKE consistent en paires de demande/réponse. Nous décrirons d'abord l'échange de base, puis les variantes. La première paire de messages (IKE_SA_INIT) négocie les algorithmes cryptographiques, les échanges de noms occasionnels *(nonce)*, et fait un échange Diffie-Hellman [DH].

La seconde paire de messages (IKE_AUTH) authentifie les messages précédents, échange les identités et certificats, et établit la première CHILD_SA. Des parties de ces messages sont chiffrées et protégées en intégrité avec des clés établies à travers l'échange IKE_SA_INIT, de sorte que les identités sont cachées aux espions et tous les champs dans tous les messages sont authentifiés.

Dans les descriptions suivantes, les charges utiles contenues dans le message sont indiquées par les noms dont la liste figure ci-dessous.

Notation	**Charge utile**
AUTH	*(Authentication)* authentification
CERT	*(Certificate)* certificat
CERTREQ	*(Certificate Request)* demande de certificat
CP	Configuration
D	*(Delete)* supprime
E	*(Encrypted)* chiffré
EAP	*(Extensible Authentication)* authentification extensible
HDR	*(IKE Header)* en-tête IKE
IDi	*(Identification – Initiateur)* identification de l'initiateur
IDr	*(Identification – Répondant)* identification du répondant
KE	*(Key Exchange)* échange de clés
Ni, Nr	*(Nonce)* nom occasionnel
N	*(Notify)* notifie
SA	*(Security Association)* association de sécurité
TSi	*(Traffic Selector – Initiateur)* sélecteur de trafic de l'initiateur
TSr	*(Traffic Selector – Répondant)* sélecteur de trafic du répondant
V	*(Vendor ID)* identifiant de fabricant

Les détails des contenus de chaque charge utile sont décrits à la section 3. Les charges utiles qui peuvent apparaître facultativement seront indiquées entre crochets, telles que [CERTREQ], qui indique qu'une charge utile de demande de certificat peut facultativement être incluse.

Les échanges initiaux sont comme suit :

Initiateur **Répondant**

HDR, SAi1, KEi, Ni →

HDR contient les indices des paramètres de sécurité (SPI, *Security Parameter Indexes*), les numéros de version, et les fanions de diverses sortes. La charge utile SAi1 établit les algorithmes cryptographiques que l'initiateur prend en charge pour la IKE_SA. La charge utile KE envoie la valeur Diffie-Hellman de l'initiateur. Ni est le nom occasionnel de l'initiateur.

← HDR, SAr1, KEr, Nr, [CERTREQ]

Le répondant choisit une suite cryptographique d'après les choix offerts par l'initiateur et exprime ce choix dans la charge utile SAr1, complète l'échange Diffie-Hellman avec la charge utile KEr, et envoie son nom occasionnel dans la charge utile Nr.

À ce point de la négociation, chaque partie peut générer un SKEYSEED, à partir duquel toutes les clés sont déduites pour cette IKE_SA. Tous les messages qui suivent sont chiffrés et protégés en intégrité sauf les en-têtes. Les clés utilisées pour le chiffrement et la protection d'intégrité sont déduites du SKEYSEED et sont appelées SK_e (chiffrement) et SK_a (authentification ou protection d'intégrité selon le cas). Une SK_e et SK_a séparée est calculée pour chaque direction. En plus des clés SK_e et SK_a déduites de la valeur DH pour la protection de la IKE_SA, une autre quantité SK_d est déduite et utilisée pour la dérivation des matériaux de clés ultérieurs pour les CHILD_SA. La notation SK { ... } indique que ces charges utiles sont chiffrées et protégées en intégrité en utilisant les SK_e et SK_a de cette direction.

HDR, SK {IDi, [CERT,] [CERTREQ,] [IDr,] AUTH, SAi2, TSi, TSr} →

L'initiateur affirme son identité avec la charge utile IDi, prouve sa connaissance du secret correspondant à IDi et protège en intégrité le contenu du premier message en utilisant la charge utile AUTH (voir au paragraphe 2.15). Il peut aussi envoyer son ou ses certificats dans la ou les charges utiles CERT et une liste de ses ancres de confiance dans la ou les charges utiles CERTREQ. Si une charge utile CERT est incluse, le premier certificat fourni DOIT contenir la clé publique utilisée pour vérifier le champ AUTH. La charge utile facultative IDr permet à l'initiateur de spécifier à laquelle des identités du répondant il veut parler. Ceci est utile lorsque la machine sur laquelle fonctionne le répondant héberge

plusieurs identités à la même adresse IP. L'initiateur commence la négociation d'une CHILD_SA en utilisant la charge utile SAi2. Les champs finaux (commençant par SAi2) sont décrits avec l'échange CREATE_CHILD_SA.

← HDR, SK {IDr, [CERT,] AUTH, SAr2, TSi, TSr}

Le répondant affirme son identité avec la charge utile IDr, envoie facultativement un ou plusieurs certificats (à nouveau avec le certificat qui contient la clé publique utilisée pour vérifier la AUTH qui figure en début sur la liste) authentifie son identité et protège l'intégrité du second message avec la charge utile AUTH, et achève la négociation d'une CHILD_SA avec les champs supplémentaires décrits ci-dessous dans l'échange CREATE_CHILD_SA.

Les receveurs des messages 3 et 4 DOIVENT vérifier que toutes les signatures et MAC sont calculés correctement et que les noms dans les charges utiles ID correspondent aux clés utilisées pour générer la charge utile AUTH.

1.3 L'échange CREATE_CHILD_SA

Cet échange consiste en une seule paire demande/réponse, et était appelé échange de phase 2 dans IKEv1. Il PEUT être initié par l'une ou l'autre extrémité de la IKE_SA après l'achèvement des échanges initiaux.

Tous les messages qui suivent l'échange initial sont protégés cryptographiquement en utilisant les algorithmes cryptographiques et les clés négociées dans les deux premiers messages de l'échange IKE. Ces messages suivants utilisent la syntaxe de la charge utile chiffrée décrite au paragraphe 3.14. Tous les messages ultérieurs incluent une charge utile chiffrée, même si dans le texte ils sont mentionnés comme "vides".

L'un ou l'autre point d'extrémité peut initier un échange CREATE_CHILD_SA, aussi dans ce paragraphe, le terme "initiateur" se réfère au point d'extrémité qui prend l'initiative de cet échange.

Une CHILD_SA est créée par l'envoi d'une demande CREATE_CHILD_SA. La demande CREATE_CHILD_SA PEUT facultativement contenir une charge utile KE pour un échange Diffie-Hellman supplémentaire afin de permettre des garanties plus fortes de secret de transmission pour la CHILD_SA. Le matériel de clés pour la CHILD_SA est une fonction de SK_d constituée durant l'établissement de la IKE_SA, des noms occasionnels échangés durant l'échange CREATE_CHILD_SA, et de la valeur Diffie-Hellman (si les charges utiles KE sont incluses dans l'échange CREATE_CHILD_SA).

Dans la CHILD_SA créée au titre de l'échange initial, on NE DOIT PAS envoyer de seconde charge utile KE ni de second nom occasionnel. Les noms occasionnels de l'échange initial sont utilisés dans le calcul des clés pour la CHILD_SA.

La demande CREATE_CHILD_SA contient :

Initiateur **Répondant**
HDR, SK {[N], SA, Ni, [KEi], [TSi, TSr]} →

L'initiateur envoie une ou des offres de SA dans la charge utile SA, un nom occasionnel dans la charge utile Ni, facultativement une valeur Diffie-Hellman dans la charge utile KEi, et les sélecteurs de trafic proposés dans les charges utiles TSi et TSr. Si cet échange CREATE_CHILD_SA renouvelle les clés d'une SA existante autre que la IKE_SA, la charge utile N de tête de type REKEY_SA DOIT identifier la SA dont les clés sont renouvelées. Si cet échange CREATE_CHILD_SA n'est pas de renouvellement de clés d'une SA existante, la charge utile N DOIT être omise. Si l'offre de la SA inclut des groupes Diffie-Hellman différents, KEi DOIT être un élément du groupe que l'initiateur s'attend qu'accepte le répondant. Si il se trompe, l'échange CREATE_CHILD_SA va échouer, et il devra faire un nouvel essai avec une KEi différente.

Le message qui suit l'en-tête est chiffré et le message incluant l'en-tête est protégé en intégrité en utilisant les algorithmes cryptographiques négociés pour la IKE_SA.

La réponse CREATE_CHILD_SA contient :

← HDR, SK {SA, Nr, [KEr], [TSi, TSr]}

Le répondant réplique (en utilisant le même identifiant de message pour répondre) avec l'offre acceptée dans une charge utile SA, et une valeur Diffie-Hellman dans la charge utile KEr si KEi était inclus dans la demande et si la suite cryptographique choisie inclut ce groupe. Si le répondant choisit une suite cryptographique avec un groupe différent, il DOIT rejeter la demande. L'initiateur DEVRAIT répéter la demande, mais cette fois avec une charge utile KEi provenant

du groupe choisi par le répondant.

Les sélecteurs de trafic pour le trafic à envoyer sur cette SA sont spécifiés dans la charge utile TS, qui peut être un sous-ensemble de ce que proposait l'initiateur de la CHILD_SA. Les sélecteurs de trafic sont omis si cette demande CREATE_CHILD_SA est utilisée pour changer la clé de la IKE_SA.

1.4 L'échange INFORMATIONAL

À divers moments du fonctionnement d'une IKE_SA, les homologues peuvent désirer s'envoyer l'un l'autre des messages de commande concernant des erreurs ou des notifications de certains événements. Pour ce faire, IKE définit un échange INFORMATIONAL. Les échanges INFORMATIONAL ne DOIVENT survenir qu'après les échanges initiaux et sont protégés cryptographiquement avec les clés négociées.

Les messages de commande qui appartiennent à une IKE_SA DOIVENT être envoyés sous cette IKE_SA. Les messages de commande qui appartiennent aux CHILD_SA DOIVENT être envoyés sous la protection de la IKE_SA qui les a générés (ou son successeur si la IKE_SA a été remplacée pour les besoins d'un renouvellement de clés).

Les messages d'un échange INFORMATIONAL contiennent zéro, une ou plusieurs charges utiles Notification, Delete, et Configuration. Le receveur d'une demande d'échange INFORMATIONAL DOIT envoyer une réponse (autrement l'envoyeur va supposer que le message a été perdu dans le réseau et va le retransmettre). Cette réponse PEUT être un message sans charge utile. Le message de demande dans un échange INFORMATIONAL PEUT aussi ne pas contenir de charge utile. C'est la façon prévue par laquelle un point d'extrémité peut demander à un autre point d'extrémité de vérifier qu'il est en vie.

Les SA ESP et AH existent toujours par paire, avec une SA dans chaque direction. Lorsque une SA est fermée, les deux membres de la paire DOIVENT être fermés. Lorsque les SA sont incorporées, comme lorsque des données (et les en-têtes IP si on est en mode tunnel) sont encapsulées d'abord avec IPComp, puis avec ESP, et finalement avec AH entre la même paire de points d'extrémité, toutes les SA DOIVENT être supprimées ensemble. Chaque point d'extrémité DOIT clore ses SA entrantes et permettre à l'autre point d'extrémité de clore l'autre SA dans chaque paire. Pour supprimer une SA, un échange INFORMATIONAL avec une ou plusieurs charges utiles Delete est envoyé avec la liste des SPI (comme elles devraient se trouver dans les en-têtes des paquets entrants) des SA à supprimer. Le receveur DOIT clore les SA désignées. Normalement, la réponse de l'échange INFORMATIONAL va contenir les charges utiles Delete pour les SA appariées allant dans l'autre direction. Il y a une exception. Si par hasard les deux extrémités d'un ensemble de SA décident indépendamment de les clore, chacun peut envoyer une charge utile Delete et les deux demandes peuvent se croiser dans le réseau. Si un nœud reçoit une demande de suppression pour des SA pour lesquelles il a déjà produit une demande Delete, il DOIT supprimer les SA sortantes tout en traitant la demande et les SA entrantes tout en traitant la réponse. Dans ce cas, les réponses NE DOIVENT PAS inclure les charges utiles Delete pour les SA supprimées, car il en résulterait une duplication de la suppression et pourrait en théorie supprimer la mauvaise SA.

Un nœud DEVRAIT regarder les connexions à moitié closes comme des anomalies et faire un audit de leur existence si elles devaient persister. Noter que la présente spécification ne spécifie nulle part de temporisations, aussi il appartient aux points d'extrémité individuels de décider du temps d'attente. Un nœud PEUT refuser d'accepter des données entrantes sur des connexions demi closes mais NE DOIT PAS les clore unilatéralement et réutiliser les SPI. Si l'état de connexion devient suffisamment embrouillé, un nœud PEUT clore la IKE_SA ; le faire clôt implicitement toutes les SA négociées sous elle. Il peut alors reconstruire les SA dont il a besoin sur une base saine sous une nouvelle IKE_SA.

L'échange INFORMATIONAL se définit comme :

Initiateur	**Répondant**
HDR, SK {[N,] [D,] [CP,] ...} →	
	← HDR, SK {[N,] [D,] [CP], ...}

Le traitement d'un échange INFORMATIONAL est déterminé par ses composants de charge utile.

1.5 Messages d'information en-dehors d'une IKE_SA

Si un paquet IKE chiffré arrive sur l'accès 500 ou 4500 avec un indice SPI non reconnu, cela pourrait être parce que le nœud de réception a eu une panne récente et perdu l'état ou à cause d'un dysfonctionnement ou d'une attaque d'un autre système. Si le nœud de réception a une IKE_SA active à l'adresse IP d'où est venu le paquet, il PEUT envoyer une notification du paquet capricieux sur cette IKE_SA dans un échange INFORMATIONAL. S'il n'a pas une telle IKE_SA, il

PEUT envoyer un message d'information sans protection cryptographique à l'adresse IP de source. Un tel message ne fait pas partie d'un échange informationnel, et le nœud de réception NE DOIT PAS y répondre. Le faire pourrait causer un message en boucle.

2 Détails du protocole IKE et variantes

Normalement, IKE écoute et envoie sur l'accès UDP 500, bien que les messages IKE puissent aussi être reçus sur l'accès UDP 4500 avec un format légèrement différent (voir au paragraphe 2.23). Comme UDP est un protocole de datagrammes (non fiable) IKE inclut dans sa définition la récupération des erreurs de transmission, y compris la perte de paquet, la répétition de paquet, et le faux paquet. IKE est conçu pour fonctionner tant que (1) au moins un d'une série de paquets retransmis atteint sa destination avant l'expiration de la temporisation ; et (2) le canal n'est pas rempli de paquets faux et répétés au point d'épuiser les capacités du réseau ou de CPU à l'un ou l'autre des points d'extrémité. Même en l'absence de ces exigences minimales de performance, IKE est conçu pour avoir des échecs propres (comme si le réseau était cassé).

Bien que les messages IKEv2 soient prévus pour être courts, ils contiennent des structures sans limite supérieure de taille (en particulier, les certificats X.509), et IKEv2 lui-même n'a as de mécanisme pour fragmenter les grands messages. IP définit un mécanisme pour la fragmentation des messages UDP sur-dimensionnés, mais la taille maximale de message acceptée varie selon les mises en œuvre. De plus, l'utilisation de la fragmentation sur IP livre les mises en œuvre aux attaques de déni de service [KPS03]. Finalement, certaines mises en œuvre de NAT et/ou pare-feu peuvent bloquer les fragments IP.

Toutes les mises en œuvre IKEv2 DOIVENT être capables d'envoyer, recevoir, et traiter les messages IKE qui vont jusqu'à 1280 octets de long, et elles DEVRAIENT être capables d'envoyer, recevoir, et traiter les messages qui vont jusqu'à 3000 octets de long. Les mises en œuvre de IKEv2 DEVRAIENT connaître la taille maximale de message UDP acceptée et PEUVENT raccourcir les messages en laissant de côté certains certificats ou propositions de suite cryptographique si cela permet de garder les messages en dessous du maximum. L'utilisation des formats "Hachage et URL" plutôt que d'inclure les certificats dans les échanges lorsque c'est possible peut éviter la plupart des problèmes. Les mises en œuvre et les configurations devraient cependant garder présent à l'esprit que si les recherches d'URL ne sont possibles qu'après l'établissement de la SA IPsec, des questions de récurrence pourraient empêcher cette technique de fonctionner.

2.1 Utilisations des temporisateurs de retransmission

Tous les messages existent en paires dans IKE : une demande et une réponse. L'établissement d'une IKE_SA consiste normalement en deux paires de demande/réponse. Une fois la IKE_SA établie, une des deux associations de sécurité peut initier les demandes à tout moment, et il peut y avoir de nombreuses demandes et réponses "en l'air" à tout moment. Mais chaque message est étiqueté comme demande ou réponse, et pour chaque paire demande/réponse une extrémité de l'association de sécurité est l'initiateur et l'autre est le répondant.

Pour chaque paire de messages IKE, l'initiateur est responsable de la retransmission pour le cas où surviendrait une fin de temporisation. Le répondant NE DOIT jamais retransmettre une réponse si il n'a pas reçu une retransmission de la demande. Dans ce cas, le répondant DOIT ignorer la demande retransmise sauf dans la mesure où il déclenche une retransmission de la réponse. L'initiateur DOIT se souvenir de chaque demande jusqu'à ce qu'il reçoive la réponse correspondante. Le répondant DOIT se souvenir de chaque réponse jusqu'à ce qu'il reçoive une demande dont le numéro de séquence soit supérieur au numéro de séquence dans la réponse plus sa taille de fenêtre (voir au paragraphe 2.3).

IKE est un protocole fiable, dans ce sens que l'initiateur DOIT retransmettre une demande jusqu'à ce qu'il reçoive une réponse correspondante OU qu'il estime que l'association de sécurité IKE a échoué et qu'il élimine tous les états associés à la IKE_SA et toutes les CHILD_SA négociées en utilisant cette IKE_SA.

2.2 Utilisation des numéros de séquence pour l'identifiant de message

Chaque message IKE contient un identifiant de message au titre de son en-tête fixe. Cet identifiant de message est utilisé pour faire correspondre demandes et réponses, et pour identifier les retransmissions des messages.

L'identifiant de message est une quantité de 32 bits, qui est zéro pour la première demande IKE dans chaque direction. Les messages d'établissement initial de IKE_SA seront toujours numérotés 0 et 1. Chaque point d'extrémité dans l'association de sécurité IKE maintient deux identifiants de message "courant" : le prochain à utiliser pour une demande qu'il initie et le

prochain qu'il s'attend à voir dans une demande provenant de l'autre extrémité. Ces compteurs s'incrémentent lorsque les demandes sont générées et reçues. Les réponses contiennent toujours le même identifiant de message que la demande correspondante. Cela signifie qu'après l'échange initial, chaque entier n peut apparaître comme l'identifiant de message dans quatre messages distincts : la n^{ème} demande provenant de l'initiateur IKE original, la réponse correspondante, la n^{ème} demande provenant du répondant IKE original, et la réponse correspondante. Si les deux extrémités rendent très différents les numéros des demandes, les identifiants de message dans les deux directions peuvent être très différents. Il n'y a cependant pas d'ambiguïté dans les messages, parce que les bits (I)nitiateur et (R)épondant dans l'en-tête de message spécifient lequel des quatre messages est un message particulier.

Noter que les identifiants de message sont protégés cryptographiquement et fournissent une protection contre la répétition de message. Dans le cas peu vraisemblable où les identifiants de message deviendraient trop grands pour tenir sur 32 bits, la IKE_SA DOIT être close. Le renouvellement de clés d'une IKE_SA remet à zéro les numéros de séquence.

2.3 Taille de fenêtre pour les demandes en chevauchement

Afin de maximiser le débit d'IKE, un point d'extrémité IKE PEUT produire plusieurs demandes avant d'obtenir une réponse à aucune d'entre elles si l'autre point d'extrémité a indiqué sa capacité à traiter de telles demandes. Pour simplifier, une mise en œuvre IKE PEUT choisir de traiter les demandes strictement dans l'ordre et/ou attendre une réponse à une demande avant d'en produire une autre. Certaines règles doivent être suivies pour s'assurer de l'interopérabilité entre des mises en œuvre utilisant des stratégies différentes.

Après l'établissement d'une IKE_SA, l'une ou l'autre extrémité peut initier une ou plusieurs demandes. Ces demandes peuvent passer l'une après l'autre sur le réseau. Un point d'extrémité IKE DOIT être prêt à accepter et traiter une demande alors qu'il a une demande en cours afin d'éviter un blocage dans cette situation. Un point d'extrémité IKE DEVRAIT être prêt à accepter et traiter plusieurs demandes alors qu'il a une demande en cours.

Un point d'extrémité IKE DOIT attendre une réponse à chacun de ses messages avant d'envoyer le message suivant sauf s'il a reçu un message Notify SET_WINDOW_SIZE de son homologue l'informant que l'homologue est prêt à maintenir l'état pour plusieurs messages en cours afin de permettre un plus gros débit.

Un point d'extrémité IKE NE DOIT PAS excéder la taille de fenêtre affichée pour l'homologue pour les demandes IKE transmises. En d'autres termes, si le répondant déclare que sa taille de fenêtre est N, lorsque l'initiateur a besoin de faire une demande X, il DOIT attendre jusqu'à ce qu'il ait reçu les réponses à toutes les demandes jusqu'à la demande X-N. Un point d'extrémité IKE DOIT garder une copie de (ou être capable de régénérer exactement) chaque demande qu'il a envoyée jusqu'à ce qu'il reçoive la réponse correspondante. Un point d'extrémité IKE DOIT garder copie (ou être capable de régénérer exactement) du nombre de réponses précédentes égal à sa taille de fenêtre déclarée pour le cas où sa réponse serait perdue et où l'initiateur demanderait sa retransmission en retransmettant la demande.

Un point d'extrémité IKE prenant en charge une taille de fenêtre supérieure à un DEVRAIT être capable de traiter les demandes entrantes qui ne sont pas à leur ordre afin de maximiser les performances en vue de défaillances du réseau ou de réordonnancement des paquets.

2.4 Synchronisation d'état et fin de temporisation de connexion

Un point d'extrémité IKE peut oublier tous ses états associés à une IKE_SA et la collection des CHILD_SA correspondantes à tout moment. C'est le comportement prévu dans le cas de défaillance et redémarrage d'un point d'extrémité. Il est important, lorsque un point d'extrémité a une défaillance ou réinitialise son état, que l'autre point d'extrémité détecte ces conditions et ne continue pas à gâcher de la bande passante en envoyant des paquets sur des SA éliminées et les voir tomber dans un trou noir.

Comme IKE est conçu pour fonctionner en dépit des attaques de déni de service (DoS) provenant du réseau, un point d'extrémité NE DOIT PAS conclure que l'autre point d'extrémité est défaillant sur la base d'informations d'acheminement (par exemple, de messages ICMP) ou de messages IKE qui arrivent sans protection cryptographique (par exemple, messages Notify se plaignant de SPI inconnus). Un point d'extrémité ne DOIT conclure que l'autre point d'extrémité a une défaillance que lorsque des tentatives répétées de le contacter sont restées sans réponse pendant une période de temporisation ou lorsque une notification INITIAL_CONTACT protégée cryptographiquement est reçue sur une IKE_SA différente sur la même identité authentifiée. Un point d'extrémité DEVRAIT suspecter que l'autre point d'extrémité a une défaillance sur la base des informations d'acheminement et initier une demande pour voir si l'autre point d'extrémité est vivant. Pour vérifier si l'autre côté est toujours en vie, IKE spécifie un message INFORMATIONAL vide qui exige

(comme toutes les demandes IKE) un accusé de réception (noter que dans le contexte d'une IKE_SA, un message "vide" consiste en un en-tête IKE suivi par une charge utile Encrypted qui ne contient pas de charge utile). Si un message protégé cryptographiquement a été reçu récemment de l'autre côté, les notifications non protégées PEUVENT être ignorées. Les mises en œuvre DOIVENT limiter le débit auquel elles entreprennent des actions sur la base de messages non protégés.

Le nombre des répétitions et la durée des temporisations ne sont pas traitées dans la présente spécification parce qu'elles n'affectent pas l'interopérabilité. Il est suggéré que les messages soient retransmis au moins une douzaine de fois sur une période de plusieurs minutes avant d'abandonner une SA, mais des environnements différents peuvent exiger des règles différentes. Pour être un bon citoyen sur le réseau, les temps entre les retransmissions DOIVENT avoir une croissance exponentielle pour éviter de noyer le réseau et rendre pire une situation d'encombrement existante. Si il y a seulement eu du trafic sortant sur toutes les SA associées à une IKE_SA, il est essentiel de confirmer que l'autre point d'extrémité est toujours en vie pour éviter les trous noirs. Si aucun message protégé cryptographiquement n'a été reçu sur une IKE_SA ou aucune de ses CHILD_SA récemment, le système doit faire une vérification de vie afin de prévenir l'envoi de messages sur un homologue mort. La réception d'un message protégé cryptographiquement frais sur une IKE_SA ou une de ses CHILD_SA assure de la vie de la IKE_SA et de toutes ses CHILD_SA. Noter que cela fait peser des exigences sur les modes de défaillance d'un point d'extrémité IKE. Une mise en œuvre NE DOIT PAS continuer d'envoyer sur une SA si quelque défaillance l'empêche de recevoir sur toutes les SA associées. Si les CHILD_SA peuvent avoir des défaillances indépendamment les unes des autres sans que la IKE_SA associée soit capable d'envoyer un message Delete, elles DOIVENT alors être négociées par des IKE_SA séparées.

Il y a une attaque de déni de service contre l'initiateur d'une IKE_SA qui peut être évitée si l'initiateur prend les bonnes précautions. Comme les deux premiers messages d'un établissement de SA ne sont pas protégés cryptographiquement, un attaquant pourrait répondre au message de l'initiateur avant le répondant authentique et empoisonner la tentative d'établissement de la connexion. Pour empêcher cela, l'initiateur PEUT vouloir accepter plusieurs réponses à son premier message, traiter chacune comme potentiellement légitime, lui répondre, puis éliminer toutes les connexions semi ouvertes invalides lorsqu'il reçoit une réponse valide protégée cryptographiquement à l'une de ses demandes. Une fois qu'une réponse cryptographiquement valide est reçue, toutes les réponses subséquentes devraient être ignorées, qu'elles soient ou non cryptographiquement valides.

Noter qu'avec ces règles, il n'y a pas de raison de négocier et se mettre d'accord sur la durée de vie d'une SA. Si IKE suppose que le partenaire est mort, sur la base d'un manque répété d'accusé de réception à un message IKE, la SA IKE et toutes les CHILD_SA établies à travers cette IKE_SA sont supprimées.

Un point d'extrémité IKE peut à tout moment supprimer les CHILD_SA inactives pour récupérer des ressources utilisées pour maintenir leur état. Si un point d'extrémité IKE choisit de supprimer des CHILD_SA, il DOIT envoyer des charges utiles Delete à l'autre extrémité en lui notifiant la suppression. Il PEUT de la même façon mettre fin à la temporisation de la IKE_SA. Clore la IKE_SA clôt implicitement toutes les CHILD_SA associées. Dans ce cas, un point d'extrémité IKE DEVRAIT envoyer une charge utile Delete indiquant qu'il a fermé la IKE_SA.

2.5 Numéros de version et rétro compatibilité

Le présent document décrit la version 2.0 de IKE, ce qui signifie que le numéro de version majeur est 2 et le numéro de version mineur est zéro. Il est vraisemblable que certaines mises en œuvre voudront accepter les deux versions 1.0 et 2.0, et à l'avenir, d'autres versions.

Le numéro de version majeur ne devrait être incrémenté que si les formats de paquet ou les actions requises ont changé à tel point qu'un nœud de version plus ancienne ne serait pas capable d'interopérer avec un nœud de version plus récente si il devait simplement ignorer les champs qu'il ne comprend pas et entreprendre les actions spécifiées dans l'ancienne spécification. Le numéro de version mineur indique de nouvelles capacités, et DOIT être ignoré par un nœud ayant un plus petit numéro de version mineur, mais utilisé pour des besoins d'information par le nœud ayant le plus grand numéro de version mineur. Par exemple, il peut indiquer la capacité à traiter un message de notification nouvellement défini. Le nœud avec le plus grand numéro de version mineur noterait simplement que son correspondant ne sera pas capable de comprendre ce message et donc, ne l'enverra pas.

Si un point d'extrémité reçoit un message avec un plus fort numéro de version majeur, il DOIT abandonner le message et DEVRAIT envoyer un message de notification non authentifié contenant le plus fort numéro de version qu'il prend en charge. Si un point d'extrémité accepte la version majeure n, et la version majeure m, il DOIT accepter toutes les versions entre n et m. Si il reçoit un message avec une version majeure qu'il accepte, il DOIT répondre avec ce numéro de version. Afin d'empêcher deux nœuds de se prendre à correspondre sur un numéro de version majeure inférieur au maximum qu'ils acceptent tous deux, IKE a un fanion qui indique qu'un nœud est capable de communiquer dans un numéro de version majeur supérieur.

Et donc, le numéro de version majeur dans l'en-tête IKE indique le numéro de version du message, et non le plus fort numéro de version qu'accepte l'émetteur. Si l'initiateur est capable de traiter les versions n, n+1, et n+2, et si le répondant est capable ode traiter les versions n et n+1, il vont négocier le traitement de n+1, et l'initiateur va mettre le fanion qui indique sa capacité à traiter une version supérieure. Si par erreur (peut-être à cause de l'envoi de messages erronés par un attaquant actif) il négocient la version n, tous deux vont alors remarquer que l'autre côté peut accepter un numéro de version plus élevé, et ils DOIVENT rompre la connexion et se reconnecter en utilisant la version n+1.

Noter que IKEv1 ne suit pas ces règles, parce qu'il n'y a pas de moyen en v1 de noter qu'on est capable de traiter un numéro de version plus élevé. Aussi, un attaquant actif peut tromper deux nœud capables de traiter la v2 pour les faire communiquer en v1. Lorsqu'un nœud capable de v2 négocie en se rabattant sur la v1, il DEVRAIT noter le fait dans ses enregistrements de journalisation.

Pour préserver la rétro compatibilité, tous les champs marqués RESERVED DOIVENT être mis à zéro par une mise en œuvre de version 2.0 et leur contenu DOIT être ignoré par une mise en œuvre de version 2.0 ("être conservateur dans ce que vous envoyez et libéral dans ce que vous recevez"). De cette façon, les futures versions du protocole pourront utiliser ces champs d'une manière dont on peut garantir qu'elle sera ignorée par les mises en œuvre qui ne les comprennent pas. De même, les types de charge utile qui ne sont pas définis sont réservés pour une utilisation future.

IKEv2 ajoute un fanion "critique" à chaque en-tête de charge utile pour plus de souplesse dans la rétro compatibilité. Si le fanion critique est mis, et si le type de charge utile n'est pas reconnu, le message DOIT être rejeté et la réponse à la demande IKE qui contient cette charge utile DOIT inclure une charge utile Notify UNSUPPORTED_CRITICAL_PAYLOAD qui indique qu'une charge utile critique non prise en charge a été incluse. Si le fanion critique n'est pas mis et si le type de charge utile n'est pas accepté, cette charge utile DOIT être ignorée.

Bien que de nouveaux types de charge utile puissent être ajoutés à l'avenir et puissent paraître entrelacés dans les champs définis par la présente spécification, les mises en œuvre DOIVENT envoyer les charges utiles définies dans la présente spécification dans l'ordre indiqué sur les figures de la section 2 et les mises en œuvre DEVRAIENT rejeter comme invalide un message ayant ces charges utiles dans tout autre ordre.

2.6 Témoins

Le terme "cookie" *(témoin)* tire son origine de Photuris de Karn et Simpson [RFC2522], proposition ancienne de gestion de clés avec IPsec, et il a persisté. L'association de sécurité Internet et le protocole de gestion de clés (ISAKMP) [RFC2408] ont fixé un en-tête de message qui inclut deux champs de huit octets intitulés "cookies", et cette syntaxe est utilisée aussi bien par IKEv1 que IKEv2 bien que dans IKEv2 ils soient désignés comme le SPI IKE et qu'il y ait un nouveau champ distinct dans une charge utile Notify qui contient le témoin. Les deux champs initiaux de huit octets dans l'en-tête sont utilisés comme identifiant de connexion au début des paquets IKE. Chaque point d'extrémité choisit un des deux SPI et DEVRAIT les choisir pour être les identifiants uniques d'une IKE_SA. Une valeur de SPI de zéro est spéciale et indique que la valeur de SPI distante n'est pas encore connue de l'envoyeur.

À la différence de ESP et AH où seul le SPI du receveur apparaît dans l'en-tête d'un message, dans IKE, le SPI de l'envoyeur est aussi envoyé dans chaque message. Comme le SPI choisi par l'initiateur original de la IKE_SA est toujours envoyé en premier, un point d'extrémité avec plusieurs IKE_SA ouvertes qui veut trouver la IKE_SA appropriée en utilisant le SPI qu'il a alloué doit regarder le bit de fanion I(nitiateur) dans l'en-tête pour déterminer si il a alloué les huit premiers ou les huit seconds octets.

Dans le premier message d'un échange initial IKE, l'initiateur ne connaîtra pas la valeur du SPI du répondant et mettra donc ce champ à zéro.

On s'attend à une attaque contre IKE à partir de l'épuisement d'état et de CPU, dans laquelle la cible est noyée sous des demandes d'initiation de session venant de fausses adresses IP. Cette attaque peut être rendue moins efficace si une mise en œuvre d'un répondant utilise une CPU minimale et n'envoie aucun état à une SA jusqu'à ce qu'il sache que l'initiateur peut recevoir des paquets à l'adresse d'où il affirme qu'il les envoie. Pour ce faire, un répondant DEVRAIT -- quand il détecte un grand nombre de IKE_SA demi-ouvertes -- rejeter les messages initiaux IKE jusqu'à ce qu'ils contiennent une charge utile Notify du type COOKIE. Il DEVRAIT à la place envoyer un message IKE non protégé en réponse et inclure une charge utile Notify COOKIE avec les données du témoin à retourner. Les initiateurs qui reçoivent de telles réponses DOIVENT réessayer le IKE_SA_INIT avec une charge utile Notify du type COOKIE contenant les données du témoin fourni par le répondant comme première charge utile et toutes les autres charges utiles inchangées. L'échange initial sera alors comme suit :

Initiateur **Répondant**

HDR(A,0), SAi1, KEi, Ni →

← -HDR(A,0), N(COOKIE)

HDR(A,0), N(COOKIE), SAi1, KEi, Ni →

← HDR(A,B), SAr1, KEr, Nr, [CERTREQ]

HDR(A,B), SK {IDi, [CERT,] [CERTREQ,] [IDr,] AUTH, SAi2, TSi, TSr} →

← HDR(A,B), SK {IDr, [CERT,] AUTH, SAr2, TSi, TSr}

Les deux premiers messages n'affectent aucun état d'initiateur ou de répondant sauf pour la communication du témoin. En particulier, les numéros de séquence de message dans les quatre premiers messages seront tous à zéro et les numéros de séquence de message dans les deux derniers messages seront à un. 'A' est le SPI alloué par l'initiateur, alors que 'B' est le SPI alloué par le répondant.

Une mise en œuvre IKE DEVRAIT mettre en œuvre la génération du témoin de répondant de telle façon qu'elle n'exige la sauvegarde d'aucun état pour reconnaître la validité de son témoin lorsque arrive le second message IKE_SA_INIT. Les algorithmes exacts et la syntaxe qu'ils utilisent pour générer les témoins n'affectent pas l'interopérabilité et donc ne sont pas spécifiés ici. Ci-après figure un exemple de la façon dont un point d'extrémité pourrait utiliser des témoins pour mettre en œuvre une protection limitée contre le déni de service.

Une bonne façon de le faire est de régler le témoin du répondant à :

Cookie = <VersionIDofSecret> | Hash(Ni | IPi | SPIi | <secret>)

où <secret> est un secret généré de façon aléatoire connu du seul répondant et changé périodiquement et | indique l'enchaînement. <VersionIDofSecret> devrait être changé à chaque fois que <secret> est régénéré. Le témoin peut être recalculé lorsque le IKE_SA_INIT arrive pour la seconde fois et est comparé au témoin reçu dans le message reçu. Si il y a correspondance, le répondant sait que le témoin a été généré depuis le dernier changement du <secret> et que IPi doit être le même que l'adresse de source qu'il a vu la première fois. Incorporer SPIi dans le calcul garantit que si plusieurs IKE_SA sont en cours d'établissement en parallèle, elles auront toutes des témoins différents (en supposant que l'initiateur choisisse un SPLi unique pour elles toutes). Incorporer Ni dans le hachage garantit qu'un attaquant qui ne voit que le message 2 ne peut pas réussir à faire un faux message 3.

Si on choisit une nouvelle valeur pour <secret> alors qu'on est engagé dans le processus d'initialisation d'une nouvelle connexion, un IKE_SA_INIT peut être retourné avec autre chose que le <VersionIDofSecret> en cours. Dans ce cas le répondant PEUT rejeter le message en envoyant une autre réponse avec un nouveau témoin ou il PEUT garder la vieille valeur de <secret> en réserve pendant un bref instant et accepter les témoins calculés à partir de l'un ou de l'autre. Le répondant NE DEVRAIT PAS accepter indéfiniment des témoins après le changement du <secret>, car cela mettrait en échec une partie de la protection contre le déni de service. Le répondant DEVRAIT changer la valeur de <secret> fréquemment, et tout spécialement s'il est soumis à des attaques.

2.7 Négociation d'algorithme cryptographique

Le type de charge utile connu comme "SA" indique une proposition d'un ensemble de choix de protocoles IPsec (IKE, ESP, et/ou AH) pour la SA ainsi que des algorithmes cryptographiques associés à chaque protocole.

Une charge utile SA consiste en une ou plusieurs propositions. Chaque proposition comporte un ou plusieurs protocoles (généralement un seul). Chaque protocole contient une ou plusieurs transformations -- chacune spécifiant un algorithme cryptographique. Chaque transformation contient zéro, un ou plusieurs attributs (les attributs ne sont nécessaires que si l'identifiant de transformation ne spécifie pas complètement l'algorithme cryptographique).

Cette structure hiérarchique a été conçue pour coder efficacement les propositions de suites cryptographiques lorsque le nombre de suites prises en charge est important parce que plusieurs valeurs sont acceptables pour plusieurs transformations. Le répondant DOIT choisir une seule suite, qui PEUT être tout sous-ensemble de la proposition SA suivant les règles ci-dessous :

Chaque proposition contient un ou plusieurs protocoles. Si une proposition est acceptée, la réponse SA DOIT contenir les mêmes protocoles dans le même ordre que la proposition. Le répondant DOIT accepter une seule proposition ou les rejeter toutes et retourner une erreur. (Par exemple : si une seule proposition contient ESP et AH et si cette proposition est acceptée, ESP et AH DOIVENT tous deux être acceptés. Si ESP et AH sont inclus dans des propositions séparées, le répondant DOIT accepter une seule d'entre elles).

Chaque proposition de protocole IPsec contient une ou plusieurs transformations. Chaque transformation contient un type transform. La suite cryptographique acceptée DOIT contenir exactement une transformation de chaque type incluse dans la proposition. Par exemple: si une proposition ESP inclut les transformations ENCR_3DES, ENCR_AES w/keysize 128, ENCR_AES w/keysize 256, AUTH_HMAC_MD5, et AUTH_HMAC_SHA, la suite acceptée DOIT contenir une des transformations ENCR_ et une des transformations AUTH_. Et donc, six combinaisons sont acceptables.

Comme l'initiateur envoie sa valeur Diffie-Hellman dans IKE_SA_INIT, il doit deviner le groupe Diffie-Hellman que le répondant va choisir dans sa liste de groupes acceptés. Si l'initiateur devine mal, le répondant répondra par une charge utile Notify de type INVALID_KE_PAYLOAD indiquant le groupe choisi. Dans ce cas, l'initiateur DOIT réessayer le IKE_SA_INIT avec le groupe Diffie-Hellman corrigé. L'initiateur DOIT à nouveau proposer l'ensemble complet de suites cryptographiques acceptables parce que le message de rejet était non authentifié et autrement, un attaquant actif pourrait tromper les points d'extrémité en négociant une suite plus faible que la plus forte préférée par tous les deux.

2.8 Renouvellement des clés

Les associations de sécurité IKE, ESP, et AH utilisent des clés secrètes qui ne DEVRAIENT être utilisées que pour une durée limitée et pour protéger une quantité de données limitée. Cela limite la durée de vie de l'association de sécurité toute entière. Lorsque la durée de vie d'une association de sécurité expire, celle-ci NE DOIT PLUS être utilisée. Si il y a une demande, de nouvelles associations de sécurité PEUVENT être établies. Le nouvel établissement d'associations de sécurité pour prendre la place de celles qui sont arrivées à expiration est appelé "renouvellement de clés" *(rekeying)*.

Pour permettre des mises en œuvre IPsec minimales, la capacité à renouveler les clés des SA sans redémarrer toute la IKE_SA est facultatif. Une mise en œuvre PEUT refuser toutes les demandes CREATE_CHILD_SA au sein d'une IKE_SA. Si une SA est arrivée à expiration ou est sur le point d'arriver à expiration et que les tentatives de renouvellement de clés en utilisant les mécanismes décrits ici échouent, une mise en œuvre DOIT clore la IKE_SA et toutes CHILD_SA associées et PEUT ensuite en commencer de nouvelles. Les mises en œuvre DEVRAIENT prendre en charge le renouvellement de clés des SA en place, car faire ainsi offre de meilleures performances et va vraisemblablement réduire le nombre de paquets perdus durant la transition.

Pour renouveler les clés d'une CHILD_SA au sein d'une IKE_SA existante, il faut créer une nouvelle SA équivalente (voir au paragraphe 2.17 ci-dessous) et quand la nouvelle est établie, supprimer l'ancienne. Pour renouveler les clés d'une IKE_SA, établir une nouvelle IKE_SA équivalente (voir au paragraphe 2.18 ci-dessous) avec l'homologue avec qui la vieille IKE_SA est partagée en utilisant un CREATE_CHILD_SA au sein de la IKE_SA existante. Une IKE_SA ainsi créée hérite de toutes les CHILD_SA de la IKE_SA originale. On utilise la nouvelle IKE_SA pour tous les messages de commandes nécessaires pour maintenir les CHILD_SA créées par la vieille IKE_SA, et supprimer celle-ci. La charge utile Delete pour se supprimer elle-même DOIT être la dernière demande envoyée sur une IKE_SA.

Les SA DEVRAIENT avoir un renouvellement de clés pro-actif, c'est-à-dire que la nouvelle SA devrait être établie avant que l'arrivée à expiration de la vieille ne la rende inutilisable. Suffisamment de temps devrait s'écouler entre le moment où la nouvelle SA est établie et celui où l'ancienne devient inutilisable afin que le trafic puisse être basculé sur la nouvelle SA.

Une différence entre IKEv1 et IKEv2 est que dans IKEv1 les durées de vie de SA étaient négociées. Dans IKEv2, chaque extrémité de la SA est chargée de mettre en application sa propre politique de durée de vie sur les SA et de renouveler les clés de la SA lorsque c'est nécessaire. Si les deux extrémités ont des politiques de durée de vie différentes, l'extrémité qui a la durée de vie la plus courte va toujours être celle qui demande le renouvellement de clés. Si un faisceau de SA a été inactif pendant un long moment et si un point d'extrémité n'initie pas SA en l'absence de trafic, le point d'extrémité PEUT choisir de clore la SA au lieu de renouveler ses clés lorsque sa durée de vie arrive à expiration. Il DEVRAIT le faire si il n'y a pas eu de trafic depuis le dernier renouvellement de clés de la SA.

Si les deux extrémités ont la même politique de durée de vie, il est possible que toutes deux initient un renouvellement de clés en même temps (d'où résulteront des SA redondantes). Pour réduire la probabilité de cet événement, la temporisation des demandes de renouvellement de clés DEVRAIT être aléatoire (retardée d'une durée aléatoire après avoir remarqué le besoin de renouvellement de clés).

Cette forme de renouvellement de clés peut temporairement résulter en plusieurs SA similaires entre les mêmes paires de nœuds. Lorsqu'il y a deux SA éligibles pour recevoir des paquets, un nœud DOIT accepter les paquets entrants par l'une et l'autre SA. Si des SA redondantes sont créées malgré une telle collision, la SA créée avec le plus faible des quatre noms occasionnels utilisés dans les deux échanges DEVRAIT être fermée par le point d'extrémité qui l'a créée.

Noter que IKEv2 permet délibérément des SA parallèles avec les mêmes sélecteurs de trafic entre des points d'extrémité

communs. Un des objets de ce comportement est de prendre en charge la différence de qualité de service (QS) du trafic parmi les SA (Voir les [RFC2474], [RFC2475], et le paragraphe 4.1 de la [RFC2983]). Et donc, à la différence de IKEv1, la combinaison des points d'extrémité et des sélecteurs de trafic peut ne pas identifier de façon univoque une SA entre ces points d'extrémité, aussi l'heuristique de renouvellement de clés de IKEv1 consistant à supprimer les SA sur la base de la duplication des sélecteurs de trafic NE DEVRAIT PAS être utilisée.

Le nœud qui a initié la SA à clés renouvelée survivante DEVRAIT supprimer la SA remplacée après que la nouvelle a été établie.

Il y a des fenêtres de temporisation – en particulier en présence de paquets perdus – où les points d'extrémité peuvent ne pas se mettre d'accord sur l'état d'une SA. Le répondant à un message CREATE_CHILD_SA DOIT être prêt à accepter des messages sur une SA avant d'envoyer sa réponse à la demande de création, afin qu'il n'y ait pas d'ambiguïté pour l'initiateur. L'initiateur PEUT commencer à envoyer sur une SA aussitôt qu'il a traité la réponse. L'initiateur ne peut cependant pas recevoir sur une SA nouvellement créée jusqu'à ce qu'il ait reçu et traité la réponse à sa demande CREATE_CHILD_SA. Comment le répondant va-t-il savoir quand il y a accord pour qu'il envoie sur la SA nouvellement créée ?

Du point de vue de la correction technique et de l'interopérabilité, le répondant PEUT commencer à envoyer sur une SA aussitôt qu'il envoie sa réponse à la demande CREATE_CHILD_SA. Dans certaines situations, cela peut cependant avoir pour résultat que des paquets soient abandonnés sans nécessité, aussi une mise en œuvre PEUT vouloir différer un tel envoi.

Le répondant peut être sûr que l'initiateur est prêt à recevoir des messages sur une SA si (1) il a reçu un message cryptographiquement valide sur une nouvelle SA, ou (2) la nouvelle SA renouvelle les clés d'une SA existante et il reçoit une demande IKE de clore la SA remplacée. Lorsqu'il renouvelle les clés d'une SA, le répondant DEVRAIT continuer à envoyer des messages sur la vieille SA jusqu'à ce qu'un de ces événements survienne. Lors de l'établissement d'une nouvelle SA, le répondant PEUT différer l'envoi de messages sur une nouvelle SA jusqu'à ce qu'il en reçoive une ou qu'une fin de temporisation survienne. Si un initiateur reçoit un message sur une SA pour laquelle il n'a pas reçu de réponse à sa demande CREATE_CHILD_SA, il DEVRAIT interpréter cela comme une perte vraisemblable de paquet et retransmettre sa demande CREATE_CHILD_SA. Un initiateur PEUT envoyer un message factice sur une SA nouvellement créée si il n'a pas de message en file d'attente afin d'assurer au répondant que l'initiateur est prêt à recevoir des messages.

2.9 Négociation de sélecteur de trafic

Lorsqu'un paquet IP est reçu par un sous-système IPsec conforme à la RFC4301 et correspond à un sélecteur "protect" dans sa base de données de politique de sécurité (SPD, *Security Policy Database*) le sous-système DOIT protéger ce paquet avec IPsec. Quand il n'existe pas encore de SA, il appartient à IKE de la créer. La maintenance de la SPD d'un système sort du domaine d'application de IKE (voir [RFC2367] pour un exemple de protocole) bien que certaines mises en œuvre puissent mettre à jour leur SPD en connexion avec le fonctionnement de IKE (pour un exemple de scénario, voir au paragraphe 1.1.3).

Les charges utiles de sélecteur de trafic (TS, *Traffic Selector*) permettent aux points d'extrémité de communiquer certaines des informations provenant de leur SPD à leurs homologues. Les charges utiles de TS spécifient les critères de choix des paquets qui seront transmis sur la SA nouvellement établie. Cela peut servir de vérification de cohérence dans certains scénarios pour s'assurer que les SPD sont cohérentes. Dans d'autres, cela sert de guide à la mise à jour dynamique de la SPD.

Deux charges utiles de TS apparaissent dans chaque message dans l'échange qui crée une paire de CHILD_SA. Chaque charge utile de TS contient un ou plusieurs sélecteurs de trafic. Chaque sélecteur de trafic consiste en une gamme d'adresses (IPv4 ou IPv6), une gamme d'accès, et un identifiant de protocole IP. À l'appui du scénario décrit au paragraphe 1.1.3, un initiateur peut demander que le répondant alloue une adresse IP et dise à l'initiateur ce qu'elle est.

IKEv2 permet au répondant de choisir un sous-ensemble du trafic proposé par l'initiateur. Cela peut arriver quand les configurations des deux points d'extrémité sont en cours de mise à jour mais qu'une seule extrémité a reçu les nouvelles informations. Comme les deux points d'extrémité peuvent être configurés par des personnes différentes, l'incompatibilité peut persister pendant un assez long moment même en l'absence d'erreurs. Cela permet aussi des configurations intentionnellement différentes, comme quand une extrémité est configurée pour tunneler toutes les adresses et dépend de l'autre extrémité pour avoir la liste mise à jour.

La première des deux charges utiles de TS est appelée TSi (sélecteur initiateur de trafic). La seconde est appelée TSr (Sélecteur de trafic répondant). TSi spécifie l'adresse de source du trafic transmis de (ou l'adresse de destination du trafic transmis à) l'initiateur de la paire de CHILD_SA. TSr spécifie l'adresse de destination du trafic transmis au (ou l'adresse de source du trafic transmis du) répondant de la paire CHILD_SA. Par exemple, si l'initiateur original demande la création d'une paire de CHILD_SA, et souhaite tunneler tout le trafic provenant du sous-réseau 192.0.1.* du côté de l'initiateur du

sous-réseau 192.0.2.* du côté du répondant, l'initiateur devra inclure un seul sélecteur de trafic dans chaque charge utile de TS. TSi spécifiera la gamme d'adresses (192.0.1.0 - 192.0.1.255) et TSr spécifiera la gamme d'adresses (192.0.2.0 - 192.0.2.255). En supposant que la proposition est acceptable pour le répondant, il renverra des charges utiles de TS identiques. (Note : La gamme d'adresses IP 192.0.2.* a été réservée pour être utilisée dans des exemples dans les RFC et documents similaires. Le présent document avait besoin de deux gammes de ce genre, et utilise donc aussi 192.0.1.*. Ceci ne devrait pas être confondu avec une adresse réelle.)

Le répondant peut rétrécir les choix en sélectionnant un sous-ensemble du trafic, par exemple en éliminant ou en réduisant la gamme d'un ou plusieurs membres de l'ensemble des sélecteurs de trafic, pourvu que l'ensemble ne devienne pas l'ensemble NUL.

Il est possible que la politique du répondant contienne plusieurs gammes plus petites, toutes mises en application par le sélecteur de trafic de l'initiateur, et que la politique du répondant soit que chacune de ces gammes soit envoyée sur une SA différente. En poursuivant l'exemple ci-dessus, le répondant pourrait avoir une politique de tunneler ces adresses de et vers l'initiateur, mais pourrait exiger que chaque paire d'adresse soit sur une CHILD_SA négociée séparément. Si l'initiateur a généré sa demande en réponse à un paquet entrant de 192.0.1.43 à 192.0.2.123, il n'y aura aucun moyen pour le répondant de déterminer quelle paire d'adresses devrait être incluse dans ce tunnel, et il faudra qu'il le devine ou rejette la demande avec un statut de SINGLE_PAIR_REQUIRED.

Pour permettre au répondant de choisir la gamme appropriée dans ce cas, si l'initiateur a demandé la SA à cause d'un paquet de données, l'initiateur DEVRAIT inclure comme premier sélecteur de trafic dans chacun de TSi et TSr un sélecteur de trafic très spécifique comportant les adresses qui étaient dans le paquet qui a déclenché la demande. Dans l'exemple, l'initiateur inclurait dans TSi deux sélecteurs de trafic : le premier contiendrait la gamme d'adresses (192.0.1.43 - 192.0.1.43) l'accès de source et le protocole IP venant du paquet, et le second contiendrait (192.0.1.0 - 192.0.1.255) avec tous les accès et protocoles IP. L'initiateur inclurait de la même manière deux sélecteurs de trafic dans TSr.

Si la politique du répondant ne lui permet pas d'accepter l'ensemble entier des sélecteurs de trafic de la demande de l'initiateur, mais lui permet d'accepter le premier sélecteur de TSi et TSr, le répondant DOIT alors réduire les sélecteurs de trafic à un sous-ensemble qui inclut les premiers choix de l'initiateur. Dans notre exemple, le répondant pourrait répondre avec TSi à (192.0.1.43 - 192.0.1.43) avec tous les accès et protocoles IP.

Si l'initiateur crée la paire de CHILD_SA en dehors d'une réponse à l'arrivée d'un paquet, mais plutôt, disons, au démarrage, il peut alors n'y avoir pas d'adresses spécifiques que préfère plus qu'un autre l'initiateur pour le tunnel initial. Dans ce cas, les premières valeurs dans TSi et TSr PEUVENT être des gammes plutôt que des valeurs spécifiques, et le répondant choisit un sous-ensemble des TSi et TSr de l'initiateur qui soient acceptables. Si plus d'un sous-ensemble est acceptable mais que leur union ne l'est pas, le répondant DOIT accepter un sous-ensemble et PEUT inclure une charge utile Notify de type ADDITIONAL_TS_POSSIBLE pour indiquer que l'initiateur peut vouloir essayer encore. Ce cas ne surviendra que lorsque l'initiateur et le répondant ont des configurations différentes les unes des autres. Si l'initiateur et le répondant sont d'accord sur la granularité des tunnels, l'initiateur ne demandera jamais un tunnel plus large que celui qu'acceptera le répondant. De tels désaccords de configuration DEVRAIENT être enregistrés dans un journal d'erreurs.

2.10 Noms occasionnels

Les messages IKE_SA_INIT contiennent chacun un nom occasionnel *(nonce)*. Ces noms occasionnels sont utilisés comme entrées aux fonctions cryptographiques. La demande CREATE_CHILD_SA et la réponse CREATE_CHILD_SA contiennent aussi des noms occasionnels. Ces noms occasionnels sont utilisés pour ajouter de la fraîcheur à la technique de déduction de clés utilisée pour obtenir des clés pour une CHILD_SA, et pour assurer la création de bits pseudo-aléatoires forts à partir de la clé Diffie-Hellman. Les noms occasionnels utilisés dans IKEv2 DOIVENT être choisis de façon aléatoire, DOIVENT être d'une taille d'au moins 128 bits, et DOITVENT être au moins de la moitié de la taille de la clé de la prf négociée ("prf" se réfère à la fonction "pseudo aléatoire", un des algorithmes cryptographiques négociés dans l'échange IKE.) Si la même source de nombres aléatoires est utilisée à la fois pour les clés et les noms occasionnels, il faut veiller à s'assurer que ce dernier usage ne compromet pas le précédent.

2.11 Adresse et agilité d'accès

IKE fonctionne sur les accès UDP 500 et 4500, et établit implicitement des associations ESP et AH pour les mêmes adresses IP que celles sur lesquelles il fonctionne. Les adresses et accès IP dans l'en-tête externe ne sont cependant pas eux-mêmes protégés cryptographiquement, et IKE est conçu pour fonctionner même à travers des boîtes de traduction d'adresse réseau (NAT, *Network Address Translation*). Une mise en œuvre DOIT accepter les demandes entrantes même si l'accès de source n'est pas 500 ou 4500, et DOIT répondre à l'adresse et à l'accès d'où la demande a été reçue. Elle DOIT

spécifier l'adresse et l'accès auxquels la demande a été reçue comme adresse de source et accès dans la réponse. IKE fonctionne de façon identique sur IPv4 ou IPv6.

2.12 Réutilisation des exponentielles de Diffie-Hellman

IKE génère du matériel de clés en utilisant un échange Diffie-Hellman éphémère afin d'obtenir la propriété de "secret de transmission parfait". Cela signifie qu'une fois qu'une connexion est close et que ses clés correspondantes sont oubliées, même quelqu'un qui aurait enregistré toutes les données de la connexion et obtenu l'accès à toutes les clés à long terme des deux points d'extrémité ne pourrait pas reconstruire les clés utilisées pour protéger la conversation sans effectuer une recherche en force (exhaustive) de l'espace de clés de la session.

Réaliser un secret de transmission parfait exige que lorsqu'une connexion est close, chaque point d'extrémité DOIVE oublier non seulement les clés utilisées par la connexion mais aussi toute information qui pourrait être utilisée pour recalculer ces clés. En particulier, il DOIT oublier les secrets utilisés dans le calcul Diffie-Hellman et tout état qui peut persister dans le générateur de nombres pseudo-aléatoires qui pourrait être utilisé pour recalculer les secrets Diffie-Hellman.

Comme le calcul des exponentielles Diffie-Hellman est coûteux en calcul, un point d'extrémité peut trouver avantageux de réutiliser ces exponentielles pour plusieurs établissements de connexion. Il y a plusieurs stratégies raisonnables pour faire cela. Un point d'extrémité pourrait ne choisir une nouvelle exponentielle que périodiquement bien qu'il puisse en résulter un secret de transmission moins parfait si une connexion dure moins que la durée de vie de l'exponentielle. Ou il pourrait garder trace des exponentielles utilisées pour chaque connexion et ne supprimer les informations associées à l'exponentielle que lorsque la connexion correspondante est close. Cela permettrait que l'exponentielle soit réutilisée sans perdre le secret de transmission parfait pour le prix du maintient de plus d'états.

Les décisions quant à la réutilisation des exponentielles Diffie-Hellman et du moment de le faire sont privées au sens où cela n'affectera pas l'interopérabilité. Une mise en œuvre qui réutilise les exponentielles PEUT choisir de se souvenir des exponentielles utilisées par l'autre point d'extrémité dans les échanges passés, et si l'un d'eux est réutilisé, pour éviter la deuxième moitié du calcul.

2.13 Génération du matériel de clés

Dans le contexte de IKE_SA, quatre algorithmes cryptographiques sont négociés : un algorithme de chiffrement, un algorithme de protection d'intégrité, un groupe Diffie-Hellman, et une fonction pseudo-aléatoire (prf, *pseudo-random function*). La fonction pseudo-aléatoire est utilisée pour la construction du matériel de clés pour tous les algorithmes cryptographiques utilisés dans la IKE_SA et les CHILD_SA.

On suppose que chaque algorithme de chiffrement et de protection d'intégrité utilise une clé de taille fixe et que chaque valeur choisie de façon aléatoire de cette taille fixe peut servir de clé appropriée. Pour les algorithmes qui acceptent une longueur de clé variable, une taille de clé fixe DOIT être spécifiée au titre de la transformée cryptographique négociée. Pour les algorithmes pour lesquels toutes les valeurs ne sont pas des clés valides (comme DES ou 3DES avec parité de clés) l'algorithme par lequel les clés sont déduites de valeurs arbitraires DOIT être spécifié par la transformation cryptographique. Pour les fonctions de protection d'intégrité fondées sur le hachage de code d'authentification de message (HMAC, *Hashed Message Authentication Code*) la taille de clé fixe est la taille du résultat de la fonction de hachage sous-jacente. Lorsque la fonction prf prend une clé de longueur variable, des données de longueur variable, et produit un résultat de longueur fixe (par exemple, en utilisant HMAC) les formules du présent document s'appliquent. Lorsque la clé pour la fonction prf a une longueur fixe, les données fournies comme clé sont tronquées ou bourrées avec des zéros en tant que de besoin sauf si un traitement exceptionnel est préconisé par la formule.

Le matériel de clés sera toujours déduit comme résultat de l'algorithme prf négocié .Comme la quantité de matériel de clés nécessaire peut être supérieure à la taille du résultat de l'algorithme prf, on utilisera la prf itérativement. On utilisera la terminologie prf+ pour décrire la fonction qui donne un flux pseudo-aléatoire sur la base des entrées d'une prf comme suit : (où | indique la concaténation)

prf+ (K,S) = T1 | T2 | T3 | T4 | ...

où :
T1 = prf (K, S | 0x01)
T2 = prf (K, T1 | S | 0x02)
T3 = prf (K, T2 | S | 0x03)
T4 = prf (K, T3 | S | 0x04)

qui se poursuit comme nécessaire pour calculer toutes les clés requises. Les clés sont tirées de la chaîne de résultat sans considération des frontières (par exemple, si les clés requises sont des clés de 256 bits de la norme de chiffrement évolué (AES) et une clé de 160 bits HMAC, et que la fonction prf génère 160 bits, la clé AES viendra de T1 et du début de T2, tandis que la clé HMAC viendra du reste de T2 et du commencement de T3).

La constante concaténée à la fin de chaque chaîne qui alimente la prf est un seul octet. prf+ dans le présent document n'est pas définie au-delà de 255 fois la taille du résultat de la prf.

2.14 Génération du matériel de clés pour la IKE_SA

Les clés partagées sont calculées comme suit : une quantité appelée SKEYSEED est calculée à partir des noms occasionnels échangés durant l'échange IKE_SA_INIT et le secret Diffie-Hellman partagé établi durant cet échange. SKEYSEED est utilisé pour calculer sept autres secrets : SK_d utilisé pour déduire de nouvelles clés pour les CHILD_SA établies avec cette IKE_SA ; SK_ai et SK_ar utilisés comme clés pour l'algorithme de protection d'intégrité pour authentifier les messages composants les échanges ultérieurs ; SK_ei et SK_er utilisés pour le chiffrement (et bien sûr, le déchiffrement) de tous les échanges ultérieurs; et SK_pi et SK_pr, qui sont utilisés lors de la génération de la charge utile AUTH.

SKEYSEED et ses dérivés sont calculés comme suit :

SKEYSEED = prf(Ni | Nr, g^ir)

{SK_d | SK_ai | SK_ar | SK_ei | SK_er | SK_pi | SK_pr } = prf+ (SKEYSEED, Ni | Nr | SPIi | SPIr)

(qui indique que les quantités SK_d, SK_ai, SK_ar, SK_ei, SK_er, SK_pi, et SK_pr sont prises dans l'ordre des bits générés de la prf+). g^ir est le secret partagé tiré de l'échange Diffie-Hellman éphémère. g^ir est représenté comme une chaîne d'octets en ordre gros boutien bourrés avec des zéros si nécessaire pour arriver à la longueur du module. Ni et Nr sont les noms occasionnels, dépouillés de tout en-tête. Si la prf négociée prend une clé de longueur fixée et si les longueurs de Ni et Nr n'ajoutent rien à cette longueur, la moitié des bits doit venir de Ni et l'autre moitié de Nr, en prenant les premiers bits de chaque.

Les deux directions de flux de trafic utilisent des clés différentes. Les clés utilisées pour protéger les messages provenant de l'initiateur d'origine sont SK_ai et SK_ei. Les clés utilisées pour protéger les messages dans l'autre direction sont SK_ar et SK_er. Chaque algorithme prend un nombre fixé de bits de matériel de clés, qui est spécifié au titre de l'algorithme. Pour les algorithmes d'intégrité fondés sur un hachage de clés, la taille de clé est toujours égale à la longueur du résultat de la fonction de hachage sous-jacente.

2.15 Authentification de la IKE_SA

Lorsqu'on n'utilise pas l'authentification extensible (voir au paragraphe 2.16) les homologues sont authentifiés en ayant un bloc de données pour chaque signe (ou MAC utilisant un secret partagé comme clé). Pour le répondant, les octets à signer commencent par le premier octet du premier SPI dans l'en-tête du second message et se terminent par le dernier octet de la dernière charge utile dans le second message. Ajouté à cela (pour les besoins du calcul de signature) il y a le nom occasionnel Ni de l'initiateur (juste la valeur, pas la charge utile qui la contient), et la valeur prf(SK_pr,IDr') où IDr' est la charge utile d'identifiant du répondant, en-tête fixe exclu. Noter que ni le nom occasionnel Ni ni la valeur prf(SK_pr,IDr') ne sont transmises. De même, l'initiateur signe le premier message, en commençant par le premier octet du premier SPI dans l'en-tête et terminant par le dernier octet de la dernière charge utile. Ajouté à cela (pour les besoins du calcul de la signature) se trouvent le nom occasionnel Nr du répondant, et la valeur prf(SK_pi,IDi'). Dans le calcul ci-dessus, IDi' et IDr' sont les charge utiles d'identifiant entières, en-tête fixe exclu. Il est critique pour la sécurité de l'échange que chaque côté signe le nom occasionnel de l'autre côté.

Noter que toutes les charges utiles sont incluses sous la signature, y compris tous types de charge utile non définis dans le présent document. Si le premier message de l'échange est envoyé deux fois (la deuxième fois avec un témoin du répondant et/ou un groupe Diffie-Hellman différent) c'est la seconde version du message qui est signée.

Facultativement, les messages 3 et 4 PEUVENT inclure un certificat, ou une chaîne de certificats donnant la preuve que la clé utilisée pour calculer la signature numérique appartient au nom dans la charge utile d'identifiant. La signature ou MAC sera calculée en utilisant les algorithmes impliqués par le type de clé utilisée par le signataire, et spécifié par le champ Auth Method *(méthode d'authentification)* dans la charge utile Authentification. Il n'est pas exigé que l'initiateur et le répondant signent avec les mêmes algorithmes cryptographiques. Le choix des algorithmes cryptographiques dépend du type de clés

qu'a chacun. En particulier, l'initiateur peut utiliser une clé partagée alors que le répondant a une clé et un certificat de signature publique. Ce sera souvent le cas (mais pas obligatoirement) qu'un secret partagé soit utilisé pour l'authentification et que la même clé soit utilisée dans les deux directions. Noter que c'est une pratique courante mais typiquement à risque d'avoir une clé partagée dérivée seulement d'un mot de passe choisi par l'usager sans incorporer aucune autre source d'aléa.

C'est typiquement à risque parce que les mots de passe choisis par les usagers n'ont vraisemblablement pas l'imprévisibilité suffisante pour résister aux attaques de dictionnaire et ces attaques ne sont pas empêchées par la méthode d'authentification. (Les applications qui utilisent l'authentification fondée sur le mot de passe pour l'amorçage et IKE_SA devraient utiliser la méthode d'authentification du paragraphe 2.16, qui est conçue pour empêcher les attaques de dictionnaire hors ligne.) La clé pré-partagée DEVRAIT contenir autant d'imprévisibilité que la plus forte clé négociée. Dans le cas d'une clé pré-partagée, la valeur AUTH est calculée par :

AUTH = prf(prf(Shared Secret,"Key Pad for IKEv2"), <msg octets>)

où la chaîne "Key Pad for IKEv2" est de 17 caractères ASCII sans terminaison nulle. Le secret partagé peut être de longueur variable. La chaîne de bourrage est ajoutée de telle sorte que si le secret partagé est déduit d'un mot de passe, la mise en œuvre IKE n'ait pas besoin de mémoriser le mot de passe en clair, mais puisse plutôt mémoriser la valeur prf(Shared Secret,"Key Pad for IKEv2"), qui ne pourrait pas être utilisée comme un équivalent de mot de passe pour des protocoles autres que IKEv2. Comme noté ci-dessus, déduire le secret partagé d'un mot de passe n'est pas sûr. Cette construction est utilisée parce qu'on se doute bien que les gens le feront quand même. L'interface de gestion par laquelle est fourni le secret partagé DOIT accepter les chaînes ASCII d'au moins 64 octets et NE DOIT PAS ajouter de terminaison nulle avant de les utiliser comme secrets partagés. Elle DOIT aussi accepter un codage HEX du secret partagé. L'interface de gestion PEUT accepter d'autres codages si l'algorithme de traduction du codage en chaîne binaire est spécifié. Si la prf négociée prend une clé de taille fixe, le secret partagé DOIT être de cette taille fixe.

2.16 Méthodes de protocole d'authentification extensible

En plus de l'authentification utilisant les signatures de clés publiques et les secrets partagés, IKE prend en charge l'authentification utilisant les méthodes définies dans la RFC 3748 [RFC3748]. Normalement, ces méthodes ont asymétriques (conçues par l'authentification d'un usager auprès d'un serveur), et elles peuvent n'être pas mutuelles. Pour cette raison, ces protocoles sont normalement utilisés pour authentifier l'initiateur auprès du répondant et DOIVENT être utilisées en conjonction avec une authentification fondée sur la signature de clé publique du répondant auprès de l'initiateur. Ces méthodes sont souvent associées aux mécanismes appelés "authentification traditionnelle".

Alors que le présent mémoire se réfère à [RFC3748] avec l'idée d'ajouter de nouvelles méthodes à l'avenir sans mettre à jour la présente spécification, certaines variantes plus simples sont exposées ici et au paragraphe 3.16. La [RFC3748] définit un protocole d'authentification qui exige un nombre variable de messages. L'authentification extensible est mise en œuvre dans IKE sous forme d'échanges IKE_AUTH supplémentaires qui DOIVENT être menés à bien afin d'initialiser la IKE_SA.

Un initiateur indique le désir d'utiliser l'authentification extensible en laissant la charge utile AUTH en dehors du message 3. En incluant une charge utile IDi mais pas la charge utile AUTH, l'initiateur a déclaré une identité mais ne l'a pas prouvée. Si le répondant veut utiliser une méthode d'authentification extensible, il va placer une charge utile de protocole d'authentification extensible (EAP, *Extensible Authentication Protocol*) dans le message 4 et différer l'envoi de SAr2, TSi, et TSr jusqu'à l'achèvement de l'authentification de l'initiateur dans un échange IKE_AUTH ultérieur. Dans le cas d'une authentification extensible minimale, l'établissement initial de SA va apparaître comme suit :

Initiateur	**Répondant**
HDR, SAi1, KEi, Ni →	
	← HDR, SAr1, KEr, Nr, [CERTREQ]
HDR, SK {IDi, [CERTREQ,] [IDr,] SAi2, TSi, TSr} →	
	← HDR, SK {IDr, [CERT,] AUTH, EAP }
HDR, SK {EAP} →	
	← HDR, SK {EAP (succès)}
HDR, SK {AUTH} →	
	← HDR, SK {AUTH, SAr2, TSi, TSr }

Pour les méthodes EAP qui créent une clé partagée comme effet secondaire de l'authentification, cette clé partagée DOIT être utilisée à la fois par l'initiateur et le répondant pour générer des charges utiles AUTH dans les messages 7 et 8 en utilisant la syntaxe des secrets partagés du paragraphe 2.15. La clé partagée provenant d'EAP est le champ de la

spécification EAP nommé MSK. La clé partagée générée durant un échange IKE NE DOIT PAS être utilisée pou un autre objet.

Les méthodes EAP qui n'établissent pas une clé partagée NE DEVRAIENT PAS être utilisées, car elles sont sujettes à un certain nombre d'attaques par interposition [EAPMITM] si ces méthodes EAP sont utilisées dans d'autres protocoles qui ne se servent pas d'un tunnel authentifié par le serveur. Prière de se reporter à la section Considérations sur la sécurité pour les détails. Si les méthodes EAP qui ne génèrent pas une clé partagée sont utilisées, les charges utiles AUTH dans les messages 7 et 8 DOIVENT être générées en utilisant respectivement SK_pi et SK_pr.

L'initiateur d'une IKE_SA utilisant EAP DEVRAIT être capable d'étendre l'échange de protocole initial à au moins dix échanges IKE_AUTH pour le cas où le répondant envoie des messages de notification et/ou réessaye l'invite d'authentification. Une fois que l'échange de protocole défini par la méthode d'authentification EAP choisie s'est achevé avec succès, le répondant DOIT envoyer une charge utile EAP contenant le message Success. De même, si la méthode d'authentification a échoué, le répondant DOIT envoyer une charge utile EAP contenant le message Failure. Le répondant PEUT terminer à tout moment l'échange IKE en envoyant une charge utile EAP contenant le message Failure.

Après un tel échange étendu, les charges utiles EAP AUTH DOIVENT être incluses dans les deux messages qui suivent celui qui contient les message EAP Success.

2.17 Génération du matériel de clés pour les CHILD_SA

Une seule CHILD_SA est créée par l'échange IKE_AUTH, et des CHILD_SA supplémentaires peuvent facultativement être créées dans les échanges CREATE_CHILD_SA. Le matériel de clés pour elles est généré comme suit :

KEYMAT = prf+(SK_d, Ni | Nr)

Où Ni et Nr sont les noms occasionnels provenant de l'échange IKE_SA_INIT si cette demande est la première CHILD_SA créée, ou les Ni et Nr frais provenant de l'échange CREATE_CHILD_SA si c'est une création ultérieure.

Pour les échanges CREATE_CHILD_SA qui incluent un échange Diffie-Hellman facultatif, le matériel de clés est défini par :

KEYMAT = prf+(SK_d, g^ir (new) | Ni | Nr)

où g^ir (new) est le secret partagé provenant de l'échange Diffie-Hellman éphémère de cet échange CREATE_CHILD_SA (représenté par une chaîne d'octets en ordre gros boutien bourré avec des zéros dans les bits de plus fort poids si nécessaire pour arriver à la longueur du module).

Une seule négociation CHILD_SA peut résulter en plusieurs associations de sécurité. Les SA ESP et AH existent par paires (une dans chaque direction), et quatre SA peuvent être créées dans une seule négociation CHILD_SA si une combinaison de ESP et AH est négociée.

Le matériel de clés DOIT être pris à partir du KEYMAT étendu dans l'ordre suivant :

- Toutes les clés pour les SA qui portent des données de l'initiateur au répondant sont prises avant les SA qui vont dans la direction inverse.
- Si plusieurs protocoles IPsec sont négociés, le matériel de clés est pris dans l'ordre dans lequel les en-têtes de protocole vont apparaître dans le paquet encapsulé.
- Si un seul protocole a des clés à la fois de chiffrement et d'authentification, la clé de chiffrement est prise des premiers octets de KEYMAT et la clé d'authentification est prise des octets suivants.

Chaque algorithme cryptographique prend un nombre fixe de bits de matériel de clé spécifié par l'algorithme.

2.18 Renouvellement des clés des IKE_SA en utilisant l'échange CREATE_CHILD_SA

L'échange CREATE_CHILD_SA peut être utilisé pour renouveler les clés d'une IKE_SA existante (voir paragraphe 2.8). Les nouveaux SPI d'initiateur et de répondant sont fournis dans les champs de SPI. Les charges utiles de TS sont omises lors du renouvellement de clés d'une IKE_SA. Le SKEYSEED pour la nouvelle IKE_SA est calculé en utilisant le SK_d provenant de la IKE_SA existante comme suit :

SKEYSEED = prf(SK_d (old), [g^ir (new)] | Ni | Nr)

où g^ir (new) est le secret partagé provenant de l'échange Diffie-Hellman éphémère de cet échange CREATE_CHILD_SA (représenté comme une chaîne d'octets en ordre gros boutien bourrée avec des zéros si nécessaire pour arriver à la longueur du module) et Ni et Nr sont les deux noms occasionnels débarrassés de tout en-tête.

La nouvelle IKE_SA DOIT remettre son compteur de messages à 0.

SK_d, SK_ai, SK_ar, SK_ei, et SK_er sont calculés à partir du SKEYSEED comme spécifié au paragraphe 2.14.

2.19 Demande d'une adresse interne sur un réseau distant

Comme il arrive le plus couramment dans le scénario de point d'extrémité à passerelle de sécurité, un point d'extrémité peut d'avoir besoin d'une adresse IP dans le réseau protégée par la passerelle de sécurité et que cette adresse soit allouée de façon dynamique. Une telle demande d'adresse temporaire peut être incluse dans toute demande de création d'une CHILD_SA (y compris la demande implicite dans le message 3) en incluant une charge utile CP.

Cette fonction fournit une allocation d'adresse à un client IPsec d'accès distant (IRAC, *IPsec Remote Access Client*) qui essaye de tunneler dans un réseau protégé par un serveur IPsec d'accès distant (IRAS, *IPsec Remote Access Server*). Comme l'échange IKE_AUTH crée une IKE_SA et une CHILD_SA, l'IRAC DOIT demander l'adresse contrôlées par l'IRAS (et facultativement d'autres informations concernant le réseau protégé) dans l'échange IKE_AUTH. L'IRAS peut procurer une adresse pour l'IRAC à partir d'un certain nombre de sources telles qu'un serveur DHCP/BOOTP ou de son propre réservoir d'adresses.

Initiateur	**Répondant**
HDR, SK {IDi, [CERT,] [CERTREQ,] [IDr,] AUTH, CP(CFG_REQUEST), SAi2, TSi, TSr} →	
	← HDR, SK {IDr, [CERT,] AUTH, CP(CFG_REPLY), SAr2, TSi, TSr}

Dans tous les cas, la charge utile CP DOIT être insérée avant la charge utile SA. Dans des variantes du protocole où il y a plusieurs échanges IKE_AUTH, la charge utile CP DOIT être insérée dans les messages contenant les charges utiles SA.

CP(CFG_REQUEST) DOIT contenir au moins un attribut INTERNAL_ADDRESS (IPv4 ou IPv6) mais PEUT contenir tout nombre d'attributs supplémentaires que l'initiateur veut retourner dans la réponse.

Par exemple, un message d'initiateur à répondant :

CP(CFG_REQUEST)=
INTERNAL_ADDRESS(0.0.0.0)
INTERNAL_NETMASK(0.0.0.0)
INTERNAL_DNS(0.0.0.0)
TSi = (0, 0-65535,0.0.0.0-255.255.255.255)
TSr = (0, 0-65535,0.0.0.0-255.255.255.255)

Note : Les sélecteurs de trafic contiennent (protocole, gamme d'accès, gamme d'adresse).

Message du répondant à l'initiateur :

CP(CFG_REPLY)=
INTERNAL_ADDRESS(192.0.2.202)
INTERNAL_NETMASK(255.255.255.0)
INTERNAL_SUBNET(192.0.2.0/255.255.255.0)
TSi = (0, 0-65535,192.0.2.202-192.0.2.202)
TSr = (0, 0-65535,192.0.2.0-192.0.2.255)

Toutes les valeurs retournées vont dépendre de la mise en œuvre. Comme on peut le voir dans l'exemple ci-dessus, l'IRAS PEUT aussi envoyer d'autres attributs qui n'étaient pas inclus dans CP(CFG_REQUEST) et PEUT ignorer les attributs non obligatoires qu'il ne prend pas en charge.

Le répondant NE DOIT PAS envoyer de CFG_REPLY sans avoir d'abord reçu une CP(CFG_REQUEST) de l'initiateur, parce qu'on ne veut pas que l'IRAS effectue de bouclage de configuration inutile si l'IRAC ne peut pas traiter la REPLY. Dans le cas où la configuration de l'IRAS exige que CP soit utilisé pour une identité IDi donnée, mais où l'IRAC a échoué à envoyer un CP(CFG_REQUEST), l'IRAS DOIT faire échouer la demande, et terminer l'échange IKE avec une erreur FAILED_CP_REQUIRED.

2.20 Demande de la version de l'homologue

Un homologue IKE souhaitant s'enquérir de la version du logiciel IKE de l'autre homologue PEUT utiliser la méthode suivante. C'est un exemple de demande de configuration au sein d'un échange INFORMATIONAL, après la création de la IKE_SA et de la première CHILD_SA.

Une mise en œuvre IKE PEUT refuser de divulguer ses informations de version avant l'authentification ou même après l'authentification pour empêcher la contagion au cas ou certaines mises en œuvre seraient réputées avoir des failles dans leur sécurité. Dans ce cas, elle DOIT retourner une chaîne vide ou pas de charge utile CP si CP n'est pas pris en charge.

Initiateur **Répondant**

HDR, SK{CP(CFG_REQUEST)} →

← HDR, SK{CP(CFG_REPLY)}

CP(CFG_REQUEST)=APPLICATION_VERSION("")

CP(CFG_REPLY) APPLICATION_VERSION("foobar v1.3beta, (c) Foo Bar Inc.")

2.21 Traitement des erreurs

De nombreuses sortes d'erreurs peuvent survenir pendant le traitement IKE. Si une demande mal formatée est reçue ou si elle est inacceptable pour des raisons de politique (par exemple, pas d'algorithme cryptographique qui corresponde) la réponse DOIT contenir une charge utile Notify qui indique l'erreur. Si une erreur survient en dehors du contexte d'une demande IKE (par exemple, le nœud obtient des messages ESP sur un SPI qui n'existe pas), le nœud DEVRAIT initier un échange INFORMATIONAL avec une charge utile Notify décrivant le problème.
Les erreurs qui surviennent avant l'établissement d'une IKE_SA protégée cryptographiquement doivent être traitées avec beaucoup de prudence. Il y a un compromis à faire entre vouloir aider à diagnostiquer un problème et y répondre et vouloir éviter d'être la dupe d'une attaque de déni de service fondée sur des messages falsifiés.
Si un nœud reçoit un message sur l'accès UDP 500 ou 4500 en dehors du contexte d'une IKE_SA connue de lui (et non une demande d'en commencer une) cela peut être le résultat d'une défaillance récente du nœud. Si le message est marqué comme réponse, le nœud PEUT auditer l'événement suspect mais NE DOIT PAS répondre. Si le message est marqué comme demande, le nœud PEUT auditer l'événement suspect et PEUT envoyer une réponse. Si une réponse est envoyée, la réponse DOIT être envoyée à l'adresse et accès IP d'où elle est venue avec les mêmes SPI IKE et l'identifiant de message copié. La réponse NE DOIT PAS être protégée cryptographiquement et DOIT contenir une charge utile Notify indiquant INVALID_IKE_SPI.
Un nœud qui reçoit une telle charge utile Notify non protégée NE DOIT PAS répondre et NE DOIT changer l'état d'aucune des SA existantes. Le message peut être une falsification ou peut être une réponse que le véritable correspondant a été conduit par fraude à envoyer. Un nœud DEVRAIT traiter un tel message (et aussi un message réseau comme une destination inaccessible ICMP) comme l'indication qu'il pourrait y avoir des problèmes avec des SA à cette adresse IP et DEVRAIT lancer une vérification de vie pour toute IKE_SA de cette sorte. Une mise en œuvre DEVRAIT limiter la fréquence de telles vérifications pour éviter de se faire manipuler dans une participation à une attaque de déni de service.
Un nœud qui reçoit un message suspect d'une adresse IP avec laquelle il a une IKE_SA PEUT envoyer une charge utile IKE Notify dans un échange IKE INFORMATIONAL sur cette SA. Le receveur NE DOIT changer l'état d'aucune des SA à la suite de cela mais DEVRAIT faire un audit de l'événement pour aider au diagnostic des dysfonctionnements. Un nœud DOIT limiter le débit auquel il va envoyer des messages en réponse à des messages non protégés.

2.22 IPComp

L'utilisation de la compression IP [RFC3173] peut être négociée au titre de l'établissement d'une CHILD_SA. Alors que la compression IP implique un en-tête supplémentaire dans chaque paquet et un indice de paramètre de compression (CPI), l'association de compression virtuelle n'a pas de vie en dehors de la SA ESP ou AH qui la contient. Les associations de compression disparaissent lorsque la SA ESP ou AH correspondante s'en va. Elle n'est pas mentionnée explicitement dans une charge utile DELETE.

La négociation de la compression IP est séparée de la négociation des paramètres cryptographiques associés à une CHILD_SA. Un nœud qui demande une CHILD_SA PEUT faire connaître son soutien à un ou plusieurs algorithmes de compression à travers une ou plusieurs charges utiles Notify de type IPCOMP_SUPPORTED. La réponse PEUT indiquer l'acceptation d'un seul algorithme de compression avec une charge utile Notify du type IPCOMP_SUPPORTED. Ces

charges utiles NE DOIVENT PAS survenir dans des messages qui ne contiennent pas de charge utile SA.

Bien qu'il y ait eu des discussions sur la question de savoir si plusieurs algorithmes de compression doivent être acceptés et si on peut avoir des algorithmes de compression différents disponibles pour les deux directions d'une CHILD_SA, les mises en œuvre de la présente spécification NE DOIVENT PAS accepter un algorithme IPComp qui n'a pas été proposé, NE DOIVENT PAS en accepter plus d'un, et NE DOIVENT PAS compresser en utilisant un algorithme autre que celui proposé et accepté dans l'établissement de la CHILD_SA.

Un effet collatéral de la séparation de la négociation de IPComp et des paramètres cryptographiques est qu'il n'est pas possible de proposer plusieurs suites cryptographiques et de proposer la compression IP avec certaines mais pas avec les autres.

2.23 Traversée de NAT

Les passerelles de traduction d'adresse réseau (NAT, *Network Address Translation*) sont un sujet controversé. Ce paragraphe décrit brièvement pourquoi elles le sont et comment elles vont vraisemblablement agir sur le trafic IKE. Beaucoup croient que les NAT sont diaboliques et que nous devrions concevoir nos protocoles de façon à les faire mieux fonctionner. IKEv2 spécifie effectivement quelques règles de traitement objectives afin que les NAT aient plus de chances de fonctionner.

Les NAT existent principalement à cause de la pénurie d'adresses IPv4, bien qu'il y ait aussi d'autres raisons. Les nœuds IP qui sont "derrière" un NAT ont des adresses IP qui ne sont pas uniques au monde, mais sont plutôt allouées à partir d'un espace qui est unique au sien du réseau derrière le NAT mais seront vraisemblablement réutilisées par des nœuds derrière d'autres NAT. Généralement, les nœuds derrière des NAT peuvent communiquer avec d'autres nœuds derrière le même NAT et avec des nœuds avec des adresses uniques au monde, mais pas avec des nœuds derrière d'autres NAT. Il y a des exceptions à cette règle. Lorsque ces nœuds font des connexions avec des nœuds sur l'Internet réel, la passerelle NAT "traduit" l'adresse IP de source en une adresse qui sera réacheminée sur la passerelle. Les messages pour la passerelle venant de l'Internet ont leurs adresses de destination "traduites" en une adresse interne qui va acheminer le paquet au bon nœud terminal.

Les NAT sont conçus pour être "transparents" pour les nœuds terminaux. Le logiciel sur le nœud qui est derrière le NAT ni le nœud sur l'Internet n'exigent de modification pour communiquer à travers le NAT. Réaliser cette transparence est plus difficile avec certains protocoles qu'avec d'autres. Les protocoles qui incluent les adresses IP des points d'extrémité au sein des charges utiles du paquet vont échouer sauf si la passerelle NAT comprend le protocole et modifie les références internes ainsi que celles dans les en-têtes. Un tel savoir est par nature non fiable, c'est une violation de la couche réseau, et il en résulte souvent des problèmes subtils.

Ouvrir une connexion IPsec à travers un NAT introduit des problèmes particuliers. Si la connexion fonctionne en mode transport, changer les adresses IP sur les paquets va causer l'échec des sommes de contrôle et le NAT ne peut pas corriger les sommes de contrôle parce qu'elles sont protégées cryptographiquement. Même en mode tunnel, il y a des problèmes d'acheminement parce que la traduction transparente des adresses des paquets AH et ESP exige une logique particulière dans le NAT et cette logique est heuristique et non fiable par nature. Pour cette raison, IKEv2 peut négocier l'encapsulation UDP des paquets IKE et ESP. Ce codage est légèrement moins efficace mais il est plus facile à traiter pour les NAT. De plus, le pare-feu peut être configuré pour passer le trafic IPsec sur UDP mais pas ESP/AH ou vice versa.

Il est de pratique courante que les NAT traduisent les numéros d'accès TCP et UDP aussi bien que les adresses et utilisent les numéros d'accès des paquets entrants pour décider quel nœud interne va obtenir tel paquet. Pour cette raison, même si les paquets IKE DOIVENT être envoyés de et vers l'accès UDP 500, ils DOIVENT être acceptés en provenance de tous les accès et les réponses DOIVENT être envoyées à l'accès d'où ils proviennent. C'est parce que les accès peuvent être modifiés lorsque les paquets passent à travers les NAT. De même, les adresses IP des points d'extrémité IKE ne sont généralement pas incluses dans les charges utiles IKE parce que les charges utiles sont protégées cryptographiquement et pourraient n'être pas modifiées de façon transparente par les NAT.

L'accès 4500 est réservé à ESP et IKE encapsulé en UDP. Lorsqu'on travaille à travers un NAT, il est généralement préférable de passer les paquets IKE sur l'accès 4500 parce que certains NAT plus anciens traitent habilement le trafic IKE sur l'accès 500 pour essayer d'établir des connexions IPsec de façon transparente entre des poin0ts d'extrémité qui ne traitent pas eux-mêmes la traversée de NAT. De tels NAT peuvent interférer avec la traversée de NAT directe envisagée dans le présent document, aussi un point d'extrémité IPsec qui découvre un NAT entre lui et son correspondant DOIT envoyer tout le trafic ultérieur de et vers l'accès 4500, que les NAT ne devraient pas traiter de façon particulière (comme ils le font avec l'accès 500).

Les exigences spécifiques de la prise en charge de la traversée de NAT [RFC3715] figurent ci-dessous. La prise en charge de la traversée de NAT est facultative. Dans ce seul paragraphe, les exigences énumérées DOIVENT s'appliquer aux seules mises en œuvre qui prennent en charge la traversée de NAT.

IKE DOIT écouter sur l'accès 4500 ainsi que sur l'accès 500. IKE DOIT répondre à l'adresse et accès IP d'où sont arrivés les paquets.

L'initiateur et le répondant IKE DOIVENT tous deux inclure dans leurs paquets IKE_SA_INIT des charges utiles Notify du type NAT_DETECTION_SOURCE_IP et NAT_DETECTION_DESTINATION_IP. Ces charges utiles peuvent être utilisées pour détecter si il y a un NAT entre les hôtes, et quelle extrémité est derrière le NAT. La localisation des charges utiles dans les paquets IKE_SA_INIT est juste après les charge utiles Ni et Nr (avant la charge utile facultative CERTREQ).

Si aucune des charges utiles NAT_DETECTION_SOURCE_IP reçues ne correspond au hachage de la source et accès IP trouvés dans l'en-tête IP du paquet qui contient la charge utile, cela signifie que l'autre extrémité est derrière un NAT (c'est-à-dire que quelqu'un le long de la route a changé l'adresse de source du paquet d'origine pour correspondre à l'adresse du NAT). Dans ce cas, cette extrémité devrait permettre une mise à jour dynamique de l'adresse IP de l'autre extrémité, comme il sera décrit plus loin.

Si la charge utile NAT_DETECTION_DESTINATION_IP reçue ne correspond pas au hachage de la destination et accès IP trouvés dans l'en-tête IP du paquet qui contient la charge utile, cela signifie que cette extrémité est derrière un NAT. Dans ce cas, cette extrémité DEVRAIT commencer à envoyer des paquets Garder-en-vie comme expliqué dans la [RFC3948].

L'initiateur IKE DOIT vérifier ces charges utiles si elles sont présentes et si elles ne correspondent pas aux adresses dans le paquet externe, il DOIT tunneler tous les futurs paquets IKE et ESP associés à cette IKE_SA sur l'accès UDP 4500.

Pour tunneler les paquets IKE sur l'accès UDP 4500, l'en-tête IKE a quatre octets de zéros ajoutés en tête et le résultat suit immédiatement l'en-tête UDP. Pour tunneler les paquets ESP sur l'accès UDP 4500, l'en-tête ESP suit immédiatement l'en-tête UDP. Comme les quatre premiers octets de l'en-tête ESP contiennent le SPI, et que le SPI ne peut pas être à zéro et valide, il est toujours possible de distinguer les messages ESP et IKE.

Les adresses IP de source et de destination originales nécessaires pour le dispositif de somme de contrôle de paquet TCP et UDP en mode transport (voir la [RFC3948]) sont obtenues des sélecteurs de trafic associés à l'échange. Dans le cas de traversée de NAT, les sélecteurs de trafic DOIVENT contenir exactement une adresse IP, qui est alors utilisée comme l'adresse IP d'origine.

Il y a des cas où un NAT décide de retirer les transpositions qui sont toujours actives (par exemple, l'intervalle de garder-en-vie est trop long, ou le NAT est réinitialisé). Pour récupérer les adresses dans ces cas là, les hôtes qui ne sont pas derrière un NAT DEVRAIENT envoyer tous les paquets (y compris les paquets de retransmission) à l'adresse et l'accès IP provenant du dernier paquet valide authentifié de l'autre extrémité (c'est-à-dire, mettre à jour l'adresse de façon dynamique). Un hôte derrière un NAT NE DEVRAIT PAS faire cela parce qu'il ouvre une possibilité d'attaque de DoS. Tout paquet IKE authentifié ou tout paquet ESP encapsulé en UDP authentifié peut être utilisé pour détecter que l'adresse ou l'accès IP a changé.

Noter que des actions similaires mais probablement pas identiques seront vraisemblablement nécessaires pour faire fonctionner IKE avec IP mobile, mais un tel traitement n'est pas visé par le présent document.

2.24 Notification explicite d'encombrement (ECN)

Lorsque des tunnels IPsec se comportent comme spécifié à l'origine dans la [RFC2401], l'usage d'ECN n'est pas approprié pour les en-têtes IP externes parce que le traitement de désencapsulation de tunnel élimine les indications d'encombrement ECN au détriment du réseau. La prise en charge par ECN des tunnels IPsec pour IPsec fondé sur IKEv1 exige plusieurs modes de fonctionnement et négociations (voir la [RFC3168]). IKEv2 simplifie cette situation en exigeant que ECN soit utilisable dans les en-têtes IP externes de toutes les SA IPsec en mode tunnel créées par IKEv2. En particulier, les encapsuleurs et désencapsuleurs de tunnel pour toutes les SA en mode tunnel créées par IKEv2 DOIVENT prendre en charge l'option ECN de pleine fonctionnalité pour les tunnels spécifiée dans la [RFC3168] et DOIVENT mettent en œuvre le traitement d'encapsulation et désencapsulation de tunnel spécifiée dans la [RFC4301] pour empêcher l'élimination des indications d'encombrement ECN.

3 Formats d'en-tête et de charge utile

3.1 L'en-tête IKE

Les messages IKE utilisent les accès UDP 500 et/ou 4500, avec un message IKE par datagramme UDP. Les informations provenant du début du paquet à travers l'en-tête UDP sont largement ignorées, à l'exception des adresses IP et accès UDP provenant des en-têtes qui sont inversés et utilisés pour les paquets de retour. Lorsqu'ils sont envoyés sur l'accès UDP 500, les messages IKE commencent immédiatement après l'en-tête UDP. Lorsqu'ils sont envoyés sur l'accès UDP 4500, les messages IKE ont quatre octets de zéros ajoutés en tête. Ces quatre octets de zéros ne font pas partie du message IKE et ne sont inclus dans aucun des champs de longueur ou de somme de contrôle définis par IKE. Chaque message IKE commence par l'en-tête IKE, noté HDR dans le présent mémoire. Suivant l'en-tête se trouvent une ou plusieurs charges utiles IKE, chacune identifiée par un champ "Prochaine charge utile" dans la charge utile précédente. Les charges utiles sont traitées dans l'ordre dans lequel elles apparaissent dans un message IKE en invoquant la routine de traitement appropriée conformément au champ "Prochaine charge utile" dans l'en-tête IKE et ensuite conformément au champ "Prochaine charge utile" dans la charge utile IKE elle-même jusqu'à ce qu'un champ "Prochaine charge utile" de zéro indique qu'aucune charge utile ne suit. Si une charge utile de type "Chiffrée" est trouvée, cette charge utile est déchiffrée et son contenu est analysé comme des charges utiles supplémentaires. Une charge utile Chiffrée DOIT être la dernière charge utile dans un paquet et une charge utile Chiffrée NE DOIT PAS contenir d'autre charge utile Chiffrée.

Le SPI du receveur dans l'en-tête identifie une instance d'association de sécurité IKE. Il est donc possible à une seule instance de IKE de multiplexer des sessions distinctes avec plusieurs homologues.

Tous les champs multi octets représentant des entiers sont disposés en ordre gros boutien (à savoir l'octet de plus fort poids d'abord, ou dans l'ordre des octets du réseau).

Le format de l'en-tête IKE est donné à la Figure 4.

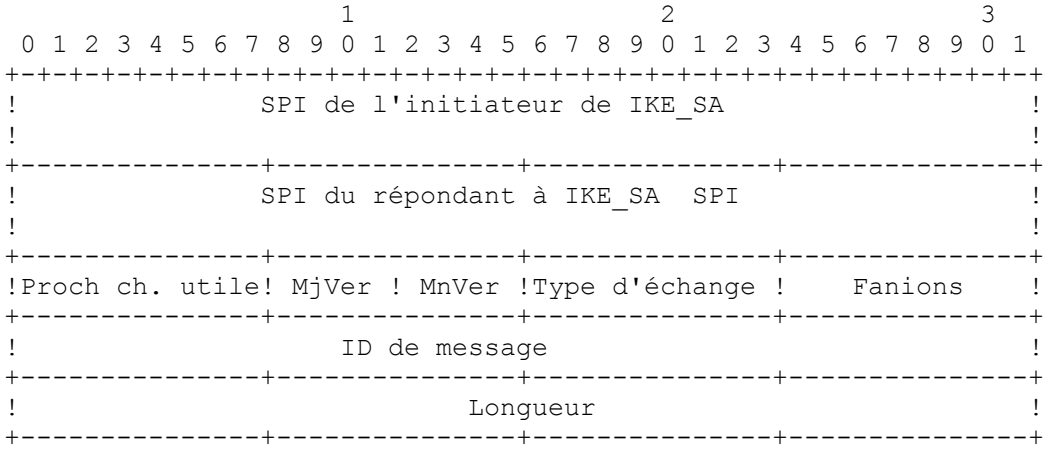


Figure 4 : Format d'en-tête IKE

- SPI de l'initiateur (8 octets) - Valeur choisie par l'initiateur pour identifier une association de sécurité IKE unique. Cette valeur NE DOIT PAS être zéro.
- SPI du répondant (8 octets) - Valeur choisie par le répondant pour identifier une association de sécurité IKE unique. Cette valeur DOIT être zéro dans le premier message d'un échange initial IKE (y compris les répétitions de ce message comportant un témoin) et NE DOIT PAS être zéro dans tout autre message.
- Prochaine charge utile (1 octet) – Indique le type de charge utile qui suit immédiatement l'en-tête. Le format et la valeur de chaque charge utile sont définis ci-dessous.
- Version majeure *(MjVer)* (4 bits) – Indique la version majeure du protocole IKE utilisé. Les mises en œuvre fondées sur cette version de IKE DOIVENT mettre Version majeure à 2. Les mises en œuvre fondées sur des versions précédentes de IKE et ISAKMP DOIVENT mettre Version majeure à 1. Les mises en œuvre fondées sur la présente version de IKE DOIVENT rejeter ou ignorer les messages qui contiennent un numéro de version supérieur à 2.
- Version mineure *(MnVer)* (4 bits) – Indique la version mineure du protocole IKE utilisé. Les mises en œuvre fondées sur cette version de IKE DOIVENT mettre Version mineure à 0. Elles DOIVENT ignorer le numéro de version mineur des messages reçus.
- Type d'échange (1 octet) – Indique le type d'échange utilisé. Ceci constitue une contrainte pour les charges utiles envoyées dans chaque message et l'ordre des messages dans un échange.

Type d'échange	Valeur
Réservé	0-33
IKE_SA_INIT	34
IKE_AUTH	35
CREATE_CHILD_SA	36
Information	37
Réservé à l'IANA	38-239
Réservé pour utilisation privée	240-255

- o Fanions (1 octet) – Indique les options spécifiques qui sont établies pour le message. La présence d'options est indiquée par l'établissement du bit approprié dans le champ de fanions. Les bits sont définis avec le bit de plus fort poids d'abord, ainsi le bit 0 serait le bit de moindre poids de l'octet Fanions. Dans la description ci-dessous, un bit mis signifie que sa valeur est '1', alors que ôté signifie que sa valeur est '0'.
  - -- X(réservé) (bits 0-2) - Ces bits DOIVENT être ôtés lors de l'envoi et DOIVENT être ignorés à réception.
  - -- I(nitiateur) (bit 3 de Fanions) - Ce bit DOIT être mis dans les messages envoyés par l'initiateur original de la IKE_SA et DOIT être ôté dans les messages envoyés par le répondant d'origine. Il est utilisé par le receveur pour déterminer quels sont les huit octets du SPI qui ont été générés par le receveur.
  - -- V(ersion) (bit 4 de Fanions) - Ce bit indique que l'émetteur est capable de traiter un numéro de version majeur du protocole supérieur à celui indiqué par le champ numéro de version majeur. Les mises en œuvre de IKEv2 DOIVENT ôter ce bit à l'envoi et DOIVENT l'ignorer dans les messages entrants.
  - -- R(éponse) (bit 5 de Fanions) - Ce bit indique que ce message est une réponse à un message contenant le même identifiant de message. Ce bit DOIT être ôté dans tous les messages de demande et DOIT être mis dans toutes les réponses. Un point d'extrémité IKE NE DOIT PAS générer une réponse à un message marqué comme étant une réponse.
  - -- X(réservé) (bits 6-7 de Fanions) - Ces bits DOIVENT être ôtés à l'envoi et DOIVENT être ignorés à réception.

- o identifiant de message (4 octets) – Identifiant de message utilisé pour contrôler la retransmission des paquets perdus et faire correspondre demandes et réponses. Il est essentiel à la sécurité du protocole parce qu'il est utilisé pour empêcher les attaques en répétition de message. Voir les paragraphes 2.1 et 2.2.
- o Longueur (4 octets) - Longueur du message total (en-tête + charges utiles) en octets.

3.2 En-tête générique de charge utile

Chaque charge utile IKE définit des paragraphes 3.3 à 3.16 commence par un en-tête générique de charge utile, montré à la Figure 5. Les figures pour chaque charge utile ci-dessous incluent l'en-tête générique de charge utile, qui sera omis pour abréger la description de chaque champ.

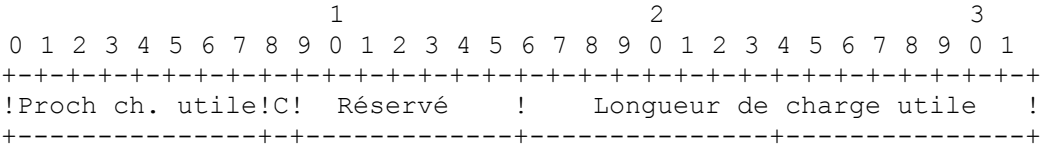


Figure 5 : En-tête générique de charge utile

Les champs d'en-tête générique de charge utile sont définis comme suit :

- o Prochaine charge utile (1 octet) – Identifiant du type de charge utile de la prochaine charge utile dans le message. Si la charge utile en cours est la dernière du message, ce champ sera alors 0. Ce champ donne une capacité de "chaînage" grâce à laquelle des charges utiles supplémentaires peuvent être ajoutées à un message en les accrochant à la fin du message et en réglant le champ "Prochaine charge utile" de la charge utile précédente de façon à indiquer le type de la nouvelle charge utile. Une charge utile Chiffrée, qui doit toujours être la dernière charge utile d'un message, est une exception. Elle contient les structures des données dans le format des charges utiles additionnelles. Dans l'en-tête d'une charge utile Chiffrée, le champ Prochaine charge utile est réglé au type de charge utile de la première charge utile contenue (au lieu de 0).

Valeurs de type de charge utile

Prochain type de charge utile	Notation	Valeur
Pas de prochaine charge utile		0
Réservé		1-32
Association de sécurité	SA	33

Échange de clé	KE	34
Identification – Initiateur	IDi	35
Identification – Répondant	IDr	36
Certificat	CERT	37
Demande de certificat	CERTREQ	38
Authentification	AUTH	39
Nom occasionnel	Ni, Nr	40
Notifier	N	41
SupprimerDelete	D	42
ID de fabricant	V	43
Sélecteur de trafic – Initiateur	TSi	44
Sélecteur de trafic – Répondant	TSr	45
Chiffré	E	46
Configuration	CP	47
Authentification extensible	EAP	48
Réservé À IANA		49-127
Usage privé		128-255

Les valeurs de type de charge utile de 1 à 32 ne devraient pas être utilisées de façon qu'il n'y ait pas de chevauchement avec les allocations de code de IKEv1. Les valeurs de type de charge utile de 49 à 127 sont réservées à l'IANA pour des allocations futures dans IKEv2 (voir la section 6). Les valeurs de type de charge utile de 128 à 255 sont pour un usage privé entre des parties mutuellement consentantes.

- C(ritique) (1 bit) - DOIT être mis à zéro si l'envoyeur veut que le receveur saute cette charge utile si il ne comprend pas le code de type de charge utile dans le champ Prochaine charge utile de la charge utile précédente. DOIT être mis à un si l'envoyeur veut que le receveur rejette la totalité du message si il ne comprend pas le type de charge utile. DOIT être ignoré par le receveur si il comprend le code de type de charge utile. DOIT être mis à zéro pour les types de charge utile définis dans ce document. Noter que le bit critique s'applique à la charge utile en cours plutôt qu'à la prochaine charge utile dont le code de type apparaît dans le premier octet. Le raisonnement derrière le non établissement du bit critique pour les charges utiles définies dans le présent document est que toutes les mises en œuvre DOIVENT comprendre tous les types de charges utiles définis dans ce document et doivent donc ignorer la valeur du bit Critique. Les charges utiles sautées sont supposées avoir des champs Prochaine charge utile et Longueur de charge utile valides.

- Réservé (7 bits) - DOIT être envoyé à zéro ; DOIT être ignoré à réception.

- Longueur de charge utile (2 octets) - Longueur en octets de la charge utile en cours, y compris l'en-tête générique de charge utile.

3.3 Charge utile d'association de sécurité

La charge utile d'association de sécurité, notée SA dans le présent mémoire, est utilisée pour négocier les attributs d'une association de sécurité. L'assemblage des charges utiles d'association de sécurité exige une grande sérénité. Une charge utile de SA PEUT contenir plusieurs propositions. S'il y en a plus d'une, elles DOIVENT être ordonnées par préférence décroissante. Chaque proposition peut contenir plusieurs protocoles IPsec (un protocole IKE, ESP, ou AH). Chaque protocole PEUT contenir plusieurs transformations, et chaque transformation PEUT contenir plusieurs attributs. Quand elle analyse une SA, une mise en œuvre DOIT vérifier que la longueur de charge utile totale est cohérente avec les longueurs internes et comptes de charges utiles. Propositions, transformations, et attributs ont chacun leurs propres codages de longueur de variable. Ils sont incorporés de telle sorte que la longueur de charge utile d'une SA inclut le contenu combiné des informations de SA, de proposition, de transformation, et d'attribut. La longueur d'une proposition inclut les longueurs de toutes les transformations et attributs qu'elle contient. La longueur d'une transformation inclut les longueurs de tous les attributs qu'elle contient.

La syntaxe des associations de sécurité, propositions, transformations, et attributs est fondée sur ISAKMP ; cependant, la sémantique est un peu différente. La raison de la complexité et de la hiérarchie est de permettre qu'il soit possible de coder plusieurs combinaisons d'algorithmes dans une seule SA. Parfois il y a un choix entre plusieurs algorithmes, alors que d'autres fois il y a une combinaison d'algorithmes. Par exemple, un initiateur peut vouloir proposer d'utiliser (AH avec MD5 et ESP avec 3DES) OU (ESP avec MD5 et 3DES).

Une des raisons du changement de la sémantique des charges utiles de SA depuis ISAKMP et IKEv1 est de rendre les codages plus compacts dans les cas courants.

La structure Proposal contient en son sein un numéro de proposition et un identifiant de protocole IPsec. Chaque structure DOIT avoir le même n° de proposition que le précédent ou lui être supérieur de un (1). La première proposition DOIT avoir un n° de proposition de un (1). Si deux structures successives ont le même numéro de proposition, cela signifie que la proposition consiste en la première structure ET la seconde. Ainsi la proposition de AH ET ESP aurait deux structures de proposition, une pour AH et une pour ESP et les deux auraient Proposal n°1. Une proposition de AH OU ESP aurait deux structures de proposition, une pour AH avec Proposal n°1 et une pour ESP avec Proposal n°2.

Chaque structure de proposition/protocole est suivie par une ou plusieurs structures de transformation. Le nombre de transformations différentes est généralement déterminé par le protocole. Généralement, AH a une seule transformation : un algorithme de vérification d'intégrité. Généralement, ESP en a deux : un algorithme de chiffrement et un algorithme de vérification d'intégrité. Généralement, IKE a quatre transformations : un groupe Diffie-Hellman, un algorithme de vérification d'intégrité, un algorithme de prf, et un algorithme de chiffrement. Si un algorithme qui combine chiffrement et protection d'intégrité est proposé, il DOIT être proposé comme algorithme de chiffrement et un algorithme de protection d'intégrité NE DOIT PAS être proposé. Pour chaque protocole, l'ensemble des transformations permises est celui des numéros d'identifiant de transformations alloués, qui apparaissent dans l'en-tête de chaque transformation.

Si il y a plusieurs transformations avec le même type Transform, ces transformations constituent un groupe dans lequel exactement une transformation sera choisie. Si il y a plusieurs de ces groupes, la proposition est un ET des choix parmi ces différents groupes. Par exemple, pour proposer ESP avec (3DES ou IDEA) et (HMAC_MD5 ou HMAC_SHA), la proposition ESP contiendrait deux candidats Transform de type 1 (un pour 3DES et un pour IDEA) et deux candidats Transform de type 2 (un pour HMAC_MD5 et un pour HMAC_SHA). Cela propose effectivement quatre combinaisons d'algorithmes. Si l'initiateur voulait proposer seulement un sous-ensemble de ceux-ci, par exemple (3DES et HMAC_MD5) ou (IDEA et HMAC_SHA), il n'y a aucun moyen de coder cela comme plusieurs transformations au sein d'une seule proposition. À la place, l'initiateur devrait construire deux propositions différentes, chacune avec deux transformations.

Une transformation donnée PEUT avoir un ou plusieurs attributs. Les attributs sont nécessaires quand la transformation peut être utilisée de plus d'une façon, comme quand un algorithme de chiffrement a une taille de clé variable. La transformation spécifierait l'algorithme et l'attribut spécifierait la taille de clé. La plupart des transformations n'ont pas d'attribut. Une transformation NE DOIT PAS avoir plusieurs attributs du même type. Pour proposer des valeurs de remplacement pour un attribut (par exemple, plusieurs tailles de clé pour l'algorithme de chiffrement AES) et une mise en œuvre DOIT inclure plusieurs transformations avec le même type Transform, chacun avec un seul attribut.

Noter que la sémantique des transformations et des attributs est assez différente de celle de IKEv1. Dans IKEv1, une seule transformation portait plusieurs algorithmes pour un protocole dont un porté dans Transform et les autres portés dans les attributs.

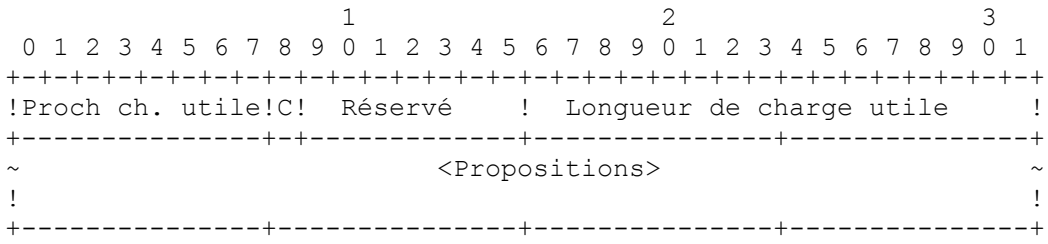
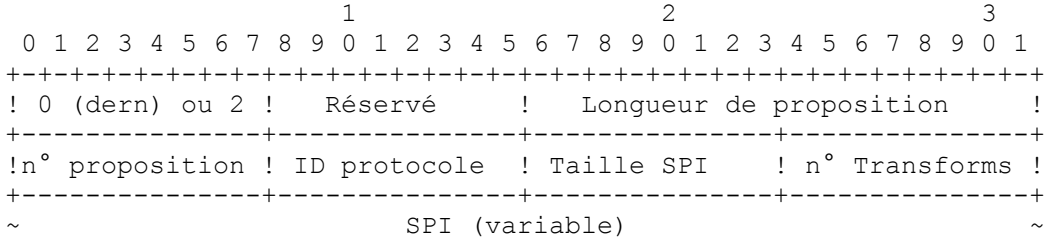


Figure 6 : Charge utile Association de sécurité

o Propositions (variable) - Une ou plusieurs sous-structures de propositions.

Le type de charge utile pour la charge utile Association de sécurité est trente trois (33).

3.3.1 Sous-structure de proposition



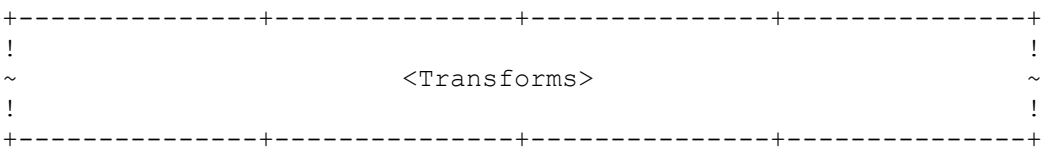


Figure 7 : Sous-structure de proposition

- 0 (dernière) ou 2 (plus) (1 octet) – Spécifie si c'est la dernière sous-structure de proposition dans la SA. Cette syntaxe est héritée de ISAKMP, mais n'est pas nécessaire parce que la dernière proposition pourrait être identifiée à partir de la longueur de la SA. La valeur (2) correspond à un type de charge utile de proposition dans IKEv1, et les quatre premiers octets de la structure Proposition sont conçus pour ressembler un peu à l'en-tête d'une charge utile.
- Réservé (1 octet) - DOIT être envoyé à zéro ; DOIT être ignoré à réception.
- Longueur de proposition (2 octets) – Longueur de cette proposition, y compris toutes les transformations et attributs qui suivent.
- N° de proposition (1 octet) – Lorsque une proposition est faite, la première proposition dans une charge utile de SA DOIT être n° 1, et les propositions suivantes DOIVENT soit être les mêmes que la proposition précédente (indiquant un ET des deux propositions) ou un de plus que la proposition précédente (indiquant un OU des deux propositions). Lorsqu'une proposition est acceptée, tous les numéros de proposition dans la charge utile de SA DOIVENT être les mêmes et DOIVENT correspondre au numéro de la proposition envoyée qui a été accepté.
- Identifiant de protocole (1 octet) – Spécifie l'identifiant de protocole IPsec pour la négociation en cours. Les valeurs définies sont :

Protocole	**Identifiant de protocole**
Réservé	0
IKE	1
AH	2
ESP	3
Réservé à IANA	4-200
Usage privé	201-255

- Taille de SPI (1 octet) - Pour une négociation de IKE_SA initiale, ce champ DOIT être à zéro ; le SPI est obtenu de l'en-tête externe. Durant les négociations ultérieures, sa taille en octets est égale à celle du SPI du protocole correspondant (8 pour IKE, 4 pour ESP et AH).
- n° de transformation (1 octet) - Spécifie le nombre de transformations dans cette proposition.
- SPI (variable) – SPI de l'entité qui envoie. Même si la taille de SPI n'est pas un multiple de 4 octets, aucun bourrage n'est appliqué à la charge utile. Lorsque le champ Taille de SPI est zéro, ce champ n'est pas présent dans la charge utile Association de sécurité.

- Transformations (variable) - Une ou plusieurs sous-structures de transformations.

3.3.2 Sous-structure de transformation

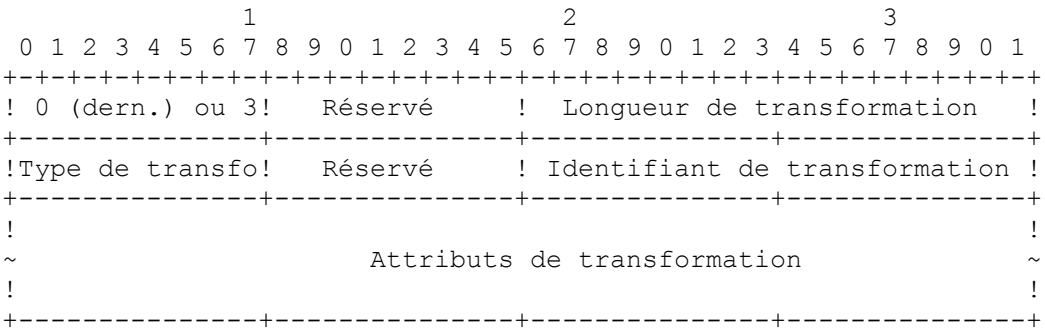


Figure 8 : Sous-structure de transformation

- 0 (dernière) ou 3 (plus) (1 octet) – Spécifie si c'est la dernière sous-structure de transformation dans la proposition. Cette syntaxe est héritée de ISAKMP, mais est inutile parce que la dernière transformation pourrait être identifiée à partir de la longueur de la proposition. La valeur (3) correspond à un type de charge utile de Transform dans IKEv1, et les quatre premiers octets de la structure Transform sont conçus pour ressembler un peu à l'en-tête d'une charge utile.

- Réservé - DOIT être mis à zéro ; DOIT être ignoré en réception.

- Longueur de transformation – Longueur (en octets) de la sous-structure Transformation y compris l'en-tête et les attributs.

- Type de transformation (1 octet) - Type de transformation spécifié dans cette transformation. Différents protocoles prennent en charge différents types de transformation. Pour certains protocoles, certaines des transformations peuvent être facultatives. Si une transformation est facultative et si l'initiateur souhaite proposer que la transformation soit omise, aucune transformation du type donné n'est incluse dans la proposition. Si l'initiateur souhaite rendre la transformation facultative pour le répondant, il inclut une sous-structure de transformation avec un identifiant de transformation = 0 comme une des options.

- Identifiant de transformation (2 octets) - Instance spécifique du type de transformation proposé.

Valeurs de type de transformation

	Type de transformation	**Utilisé dans**
Réservé	0	
Algorithme de chiffrement (ENCR)	1	(IKE et ESP)
Fonction pseudo-aléatoire (PRF)	2	(IKE)
Algorithme d'intégrité (INTEG)	3	IKE, AH, facultatif en ESP)
Groupe Diffie-Hellman (D-H)	4	IKE, facultatif en AH & ESP)
Numéro de séquence étendu (ESN)	5	(AH et ESP)
Réservé à IANA	6-240	
Usage privé	241-255	

Pour la transformation de type 1 (Algorithme de chiffrement), les identifiants de transformation définis sont :

Nom	**Numéro**	**Défini dans**
Réservé	0	
ENCR_DES_IV64	1	(RFC 1827)
ENCR_DES	2	(RFC 2405), [DES]
ENCR_3DES	3	(RFC 2451)
ENCR_RC5	4	(RFC 2451)
ENCR_IDEA	5	(RFC 2451), [IDEA]
ENCR_CAST	6	(RFC 2451)
ENCR_BLOWFISH	7	(RFC 2451)
ENCR_3IDEA	8	(RFC 2451)
ENCR_DES_IV32	9	
Réservé	10	
ENCR_NULL	11	(RFC 2410)
ENCR_AES_CBC	12	(RFC 3602)
ENCR_AES_CTR	13	(RFC 3664)

Les valeurs 14 à 1023 sont réservées à l'IANA. Les valeurs 1 024 à 65 535 sont à usage privé entre parties mutuellement consentantes.

Pour le type 2 de transformation (Fonction pseudo-aléatoire), les identifiants de transformation définis sont :

Nom	**Numéro**	**Défini dans**
Réservé	0	
PRF_HMAC_MD5	1	(RFC 2104), [RFC1321]
PRF_HMAC_SHA1	2	(RFC 2104), [SHA]
PRF_HMAC_TIGER	3	(RFC 2104)
PRF_AES128_XCBC	4	(RFC 3664)

Les valeurs 5 à 1023 sont réservées à l'IANA. Les valeurs 1 024 à 65 535 sont à usage privé entre parties mutuellement consentantes.

Pour le type 3 de transformation (Algorithme d'intégrité), les identifiants de transformation définis sont :

Nom	**Numéro**	**Défini dans**
Aucun	0	
AUTH_HMAC_MD5_96	1	(RFC 2403)
AUTH_HMAC_SHA1_96	2	(RFC 2404)
AUTH_DES_MAC	3	

AUTH_KPDK_MD5	4	(RFC 1828
AUTH_AES_XCBC_96	5	(RFC 3566)

Les valeurs 6 à 1023 sont réservées à l'IANA. Les valeurs 1024-65535 sont à usage privé entre parties mutuellement consentantes.

Pour le type 4 de transformation (Groupe Diffie-Hellman), les identifiants de transformation définis sont :

Nom	**Numéro**
Aucun	0
Défini à l'appendice B	1 – 2
Réservé	3 – 4
Défini dans [RFC3526]	5
Réservé à l'IANA	6 – 13
Défini dans [RFC3526]	14 – 18
Réservé à l'IANA	19 - 1023
Usage privé	1024 – 65 535

Pour le type 5 de transformation (Numéro de séquence étendu), les identifiants de transformation définis sont :

Nom	**Numéro**
Pas de numéro de séquence étendu	0
Numéro de séquence étendu	1
Réservé	2 – 65535

3.3.3 Types de transformation valides par protocole

Le nombre et le type de transformations qui accompagnent une charge utile de SA dépendent du protocole de la SA elle-même. Une charge utile de SA qui propose l'établissement d'une SA a les types de transformations obligatoires et facultatifs suivants. Une mise en œuvre conforme DOIT comprendre tous les types obligatoires et facultatifs pour chaque protocole qu'elle accepte (bien qu'elle n'ait pas besoin d'accepter les propositions avec des suites inacceptables). Une proposition PEUT omettre les types facultatifs si la seule valeur qu'elle puisse en accepter est Aucun.

Protocole	**Types obligatoires**	**Types facultatifs**
IKE	ENCR, PRF, INTEG, D-H	
ESP	ENCR, ESN	INTEG, D-H
AH	INTEG, ESN	D-H

3.3.4 Identifiants de transformations obligatoires

La spécification des suites qui DOIVENT et DEVRAIENT être prises en charge pour l'interopérabilité a été retirée du présent document parce qu'elles vont vraisemblablement changer plus rapidement que le document lui-même.

Une importante leçon tirée de IKEv1 est qu'aucun système ne devrait seulement mettre en œuvre les algorithmes obligatoires et s'attendre à ce qu'ils constituent le meilleur choix pour tous les consommateurs. Par exemple, au moment de la rédaction du présent document, de nombreux utilisateurs de IKEv1 commençaient à migrer vers AES en mode de chaînage de bloc de chiffrement (CBC, *Cipher Block Chaining*) pour les applications de réseau privé virtuel (VPN, *Virtual Private Network*). De nombreux systèmes IPsec fondés sur IKEv2 mettront en œuvre AES, des groupes Diffie-Hellman supplémentaires, et des algorithmes de hachage supplémentaires, et certains utilisateurs d'IPsec exigent déjà ces algorithmes en plus de ceux figurant sur la liste ci-dessus.

Il est vraisemblable que l'IANA va ajouter des transformations supplémentaires à l'avenir, et certains utilisateurs peuvent vouloir utiliser des suites privées, en particulier pour IKE où les mises en œuvre devraient être capables de prendre en charge différents paramètres, jusqu'à certaines limites de taille. À l'appui de cet objectif, toutes les mises en œuvre de IKEv2 DEVRAIENT inclure des facilités de gestion qui permettent la spécification (par un utilisateur ou administrateur de système) de paramètres Diffie-Hellman (DH) (le générateur, le module, et les longueurs et valeurs d'exposant) pour les nouveaux groupes DH. Les mises en œuvre DEVRAIENT fournir une interface de gestion au moyen de laquelle ces paramètres et les identifiants de transformations associées puissent être entrés (par un utilisateur ou administrateur de système) pour permettre de négocier de tels groupes.

Toutes les mises en œuvre de IKEv2 DOIVENT inclure une facilité de gestion qui permette à un utilisateur ou administrateur de système de spécifier les suites dont l'utilisation avec IKE est acceptable. À réception d'une charge utile

avec un ensemble d'identifiants de transformations, la mise en œuvre DOIT comparer les identifiants de transformation transmis à ceux configurés localement par les contrôles de gestion, pour vérifier que la suite proposée est acceptable sur la base d'une politique locale. La mise en œuvre DOIT rejeter les propositions de SA qui ne sont pas autorisées par ces contrôles de suite IKE. Noter que les suites cryptographiques qui DOIVENT être mises en œuvre n'ont pas besoin d'être configurées comme acceptables pour la politique locale.

3.3.5 Attributs de transformation

Chaque transformation dans une charge utile d'association de sécurité peut inclure des attributs qui modifient ou complètent la spécification de la transformation. Ces attributs sont des paires de type/valeur et sont définis ci-dessous. Par exemple, si un algorithme de chiffrement a une clé de longueur variable, la longueur de clé à utiliser peut être spécifiée comme un attribut. Les attributs peuvent avoir une valeur d'une longueur fixe de deux octets ou une valeur de longueur variable. Pour cette dernière, l'attribut est codé comme type/longueur/valeur (TLV).

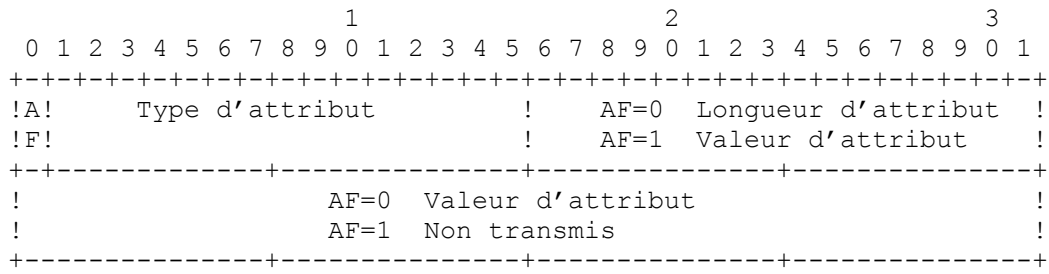


Figure 9 : Attributs de données

- Type d'attribut (2 octets) – Identifiant unique pour chaque type d'attribut (voir ci-dessous).
  Le bit de plus fort poids de ce champ est le bit Format d'attribut (AF). Il indique si les attributs de données suivent le format de TLV ou un format raccourci de type/valeur (TV). Si le bit AF est à zéro (0), les attributs de données sont alors de la forme TLV. Si le bit AF est à un (1), les attributs de données sont alors de la forme type/valeur.

- Longueur d'attribut (2 octets) - Longueur en octets de la valeur de l'attribut. Quand le bit AF est à un (1), la valeur de l'attribut est seulement de 2 octets et le champ Longueur d'attribut n'est pas présent.

- Valeur d'attribut (longueur variable) – Valeur de l'attribut associé au type d'attribut. Si le bit AF est à zéro (0), ce champ a une longueur de variable définie par le champ Longueur d'attribut. Si le bit AF est a un (1), la valeur d'attribut a une longueur de deux octets.

Noter qu'un seul type d'attribut (Longueur de clé) est défini, et il est de longueur fixe. La spécification de codage de longueur variable n'est incluse que pour les extensions futures. Les seuls algorithmes définis dans le présent document qui acceptent des attributs sont le chiffrement fondé sur AES, l'intégrité, et les fonctions pseudo-aléatoires, qui requièrent un seul attribut spécifiant la largeur de clé.

Les attributs décrits comme étant de base NE DOIVENT PAS être codés en utilisant le codage de longueur variable. Les attributs de longueur variable NE DOIVENT PAS être codés comme étant de base même si leur valeur peut tenir en deux octets. NOTE : Ceci est un changement par rapport à IKEv1, où une souplesse accrue peut avoir facilité la composition des messages mais a certainement compliqué l'analyse.

Type d'attribut	**Valeur**	**Format d'attribut**
Réservé	0 - 13	
Longueur de clé (en bits)	14	TV
Réservé	15 - 17	
Réservé à l'IANA	18 – 16 383	
Usage privé	16 384 – 32 767	

Les valeurs 0 à 13 et 15 à 17 étaient utilisées dans un contexte similaire dans IKEv1 et ne devraient pas être allouées sauf à des valeurs correspondantes. Les valeurs 18 à 16 383 sont réservées à l'IANA. Les valeurs 16 384 à 32 767 sont pour usage privé entre des parties mutuellement consentantes.

- Longueur de clé

Lorsqu'on utilise un algorithme de chiffrement qui a une clé de longueur variable, cet attribut spécifie la longueur de clé en bits (DOIT utiliser l'ordre des octets du réseau). Cet attribut NE DOIT PAS être utilisé lorsque l'algorithme de chiffrement spécifié utilise une clé de longueur fixe.

3.3.6 Négociation d'attributs

Durant la négociation d'association de sécurité, les initiateurs présentent des offres aux répondants. Les répondants DOIVENT choisir un seul ensemble complet de paramètres parmi les offres (ou rejeter toutes les offres si aucune n'est acceptable). Si il y a plusieurs propositions, le répondant DOIT choisir un seul numéro de proposition et retourner toutes les sous-structures de proposition avec ce numéro de proposition. S'il y a plusieurs transformations du même type, le répondant DOIT en choisir une seule. Tout attribut d'une transformation choisie DOIT être retourné non modifié. L'initiateur d'un échange DOIT vérifier que l'offre acceptée est cohérente avec une de ses propositions, et sinon, cette réponse DOIT être rejetée.

La négociation des groupes Diffie-Hellman présente quelques défis particuliers. Les offres de SA comportent les attributs proposés et un numéro public Diffie-Hellman (KE) dans le même message. Si dans l'échange initial l'initiateur offre d'utiliser un groupe Diffie-Hellman parmi plusieurs, il DEVRAIT prendre celui que le répondant va le plus vraisemblablement accepter et inclure un KE correspondant à ce groupe. Si l'hypothèse s'avère être fausse, le répondant indiquera le groupe correct dans la réponse et l'initiateur DEVRAIT prendre un élément de ce groupe comme valeur de son KE lorsqu'il réessaye le premier message. Cependant, il DEVRAIT continuer de proposer l'ensemble des groupes qu'il prend pleinement en charge afin d'empêcher une attaque de dégradation par interposition.

Note de mise en œuvre :

Certains attributs négociables peuvent avoir des gammes de valeurs acceptables ou pourraient avoir plusieurs valeurs acceptables. Parmi elles figure la longueur de clé d'un chiffrement symétrique à longueur de clé variable. Pour l'interopérabilité future et pour prendre en charge de façon indépendante la remise à niveau des points d'extrémité, les développeurs de ce protocole DEVRAIENT accepter les valeurs dont ils pensent qu'elles fournissent une plus grande sécurité. Par exemple, si un homologue est configuré pour accepter un chiffrement de longueur variable avec une longueur de clé de X bits et s'il lui est offert un chiffrement avec une plus grande longueur de clé, la mise en œuvre DEVRAIT accepter l'offre si elle prend en charge la plus longue clé.

La prise en charge de cette capacité permet à une mise en œuvre d'exprimer un concept de "au moins" un certain niveau de sécurité -- "une longueur de clé de _au_moins_ X bits pour le chiffrement Y".

3.4 Charge utile d'échange de clés

La charge utile d'échange de clés, notée KE dans le présent mémoire, est utilisée pour échanger des nombres publics Diffie-Hellman au titre d'un échange de clés Diffie-Hellman. La charge utile d'échange de clés consiste en un en-tête de charge utile générique IKE suivi par la valeur publique Diffie-Hellman elle-même.

Figure 10 : Format de charge utile d'échange de clés

Une charge utile d'échange de clés est construite en copiant une valeur publique Diffie-Hellman dans la portion "Données d'échange de clés" de la charge utile. La longueur de la valeur publique Diffie-Hellman DOIT être égale à la longueur du principal module sur lequel l'exponentiation a été effectuée, en ajoutant des bits à zéro à l'avant de la valeur si nécessaire.

Le n° de groupe DH identifie le groupe Diffie-Hellman dans lequel les données d'échange de clés ont été calculées (voir au paragraphe 3.3.2). Si la proposition choisie utilise un groupe Diffie-Hellman différent, le message DOIT être rejeté avec une charge utile Notify du type INVALID_KE_PAYLOAD.

Le type de charge utile pour la charge utile d'échange de clés est trente quatre (34).

3.5 Charges utiles d'identification

Les charges utiles d'identification, notées IDi et IDr dans le présent mémoire, permettent aux homologues d'attester mutuellement de leur identité. Cette identité peut être utilisée pour une recherche de politique, mais n'a pas nécessairement à correspondre à quelque chose dans la charge utile CERT ; les deux champs peuvent être utilisés par une mise en œuvre pour prendre les décisions de contrôle d'accès.

Note : Dans IKEv1, deux charges utiles d'identifiant étaient utilisées dans chaque direction pour détenir les informations de sélecteurs de trafic (TS) pour les données passant sur la SA. Dans IKEv2, ces informations sont portées dans les charges utiles de TS (voir au paragraphe 3.13).

La charge utile d'identification consiste en un en-tête générique de charge utile IKE suivi par les champs d'identification :

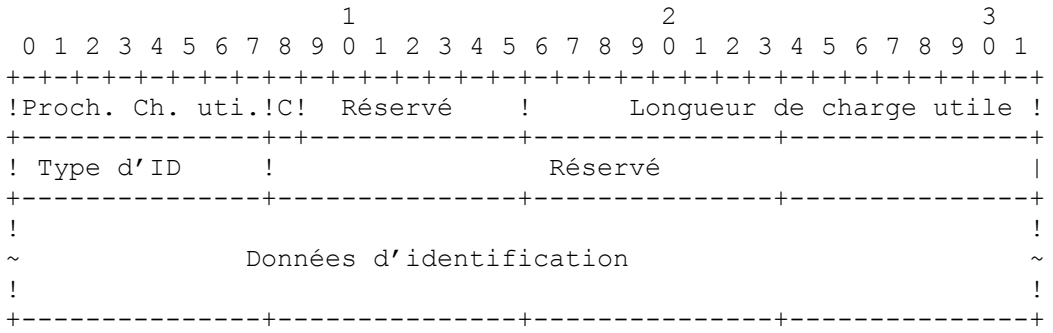


Figure 11 : Format de charge utile d'identification

- Type d'identifiant (1 octet) – Spécifie le type d'identification utilisée.
- Réservé - DOIT être envoyé à zéro ; DOIT être ignoré à réception.
- Données d'identification (longueur variable) - Valeur, comme indiquée par le type d'identification. La longueur des données d'identification est calculée à partir de la taille dans l'en-tête de charge utile d'identification.

Les types de charge utile pour la charge utile d'identification sont trente cinq (35) pour IDi et trente six (36) pour IDr.

Le tableau ci-après fait la liste des valeurs allouées pour le champ Type d'identification, suivi par une description des données d'identification qui viennent après :

Type d'identification	**Valeur**	**Description des données**
Réservé	0	
ID_IPV4_ADDR	1	Une seule adresse IPv4 de quatre (4) octets
ID_FQDN	2	Chaîne de nom de domaine pleinement qualifié. Un exemple de aID_FQDN est "exemple.com". La chaîne DOIT ne contenir aucune terminaison (par exemple, NULL, CR, etc.).
ID_RFC822_ADDR	3	Chaîne d'adresse de messagerie électronique pleinement qualifiée de la RFC 822. Un exemple de ID_RFC822_ADDR est "jsmith@exemple.com". La chaîne DOIT ne contenir aucune terminaison.
Réservé à l'IANA	4	
ID_IPV6_ADDR	5	Une seule adresse IPv6 de seize (16) octets.
Réservé à l'IANA	6 – 8	
ID_DER_ASN1_DN	9	Le codage binaire des règles de codage distinctif (DER) d'un nom distinctif X.500 en ASN.1 X.500 [X.501].
ID_DER_ASN1_GN	10	Le codage binaire DER d'un nom général X.500 en ASN.1 [X.509].
ID_KEY_ID	11	Flux d'octets opaque qui peut être utilisé pour passer les informations spécifiques du fabricant nécessaires pour faire certains types d'identification brevetés.
Réservé à l'IANA	12-200	
Réservé à usage privé	201-255	

Deux mises en œuvre ne vont interopérer que si chacune peut générer un type d'ID acceptable pour l'autre. Pour assurer une interopérabilité maximale, les mises en œuvre DOIVENT être configurables pour envoyer au moins une des

ID_IPV4_ADDR, ID_FQDN, ID_RFC822_ADDR, ou ID_KEY_ID, et DOIVENT être configurables pour accepter tous ces types. Les mises en œuvre DEVRAIENT être capables de générer et d'accepter tous ces types. Les mises en œuvre capables de IPv6 DOIVENT en plus être configurables pour accepter ID_IPV6_ADDR. Les mises en œuvre qui n'acceptent que IPv6 PEUVENT être configurables pour envoyer seulement ID_IPV6_ADDR.

3.6 Charge utile de certificat

La charge utile de certificat, notée CERT dans le présent mémoire, donne le moyen de transporter des certificats ou d'autres informations en relation avec l'authentification via IKE. Les charges utiles de certificat DEVRAIENT être incluses dans un échange si les certificats sont disponibles pour l'envoyeur à moins que l'homologue n'ait indiqué sa capacité à restituer ces informations à partir d'ailleurs en utilisant une charge utile Notify HTTP_CERT_LOOKUP_SUPPORTED. Noter que le terme "Charge utile de certificat" est quelque peu trompeur, parce que tous les mécanismes d'authentification n'utilisent pas des certificats et que des données autres que des certificats peuvent être passées dans cette charge utile.

La charge utile de certificat est définie comme suit :

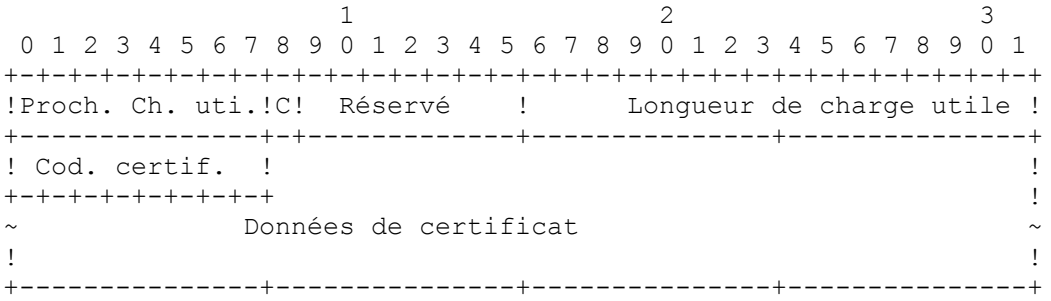


Figure 12 : Format de charge utile de certificat

o Codage de certificat (1 octet) – Ce champ indique le type de certificat ou d'informations se rapportant au certificat contenu dans le champ Données de certificat.

Codage de certificat	**Valeur**
Réservé	0
Certificat X.509 enveloppé dans PKCS n°7	1
Certificat PGP	2
Clé signée DNS	3
Certificat – Signature X.509	4
Jeton Kerberos	6
Liste de révocation de certificats (CRL)	7
Liste de révocation d'autorité (ARL)	8
Certificat SPKI	9
Certificat – Attribut X.509	10
Clé RSA brute	11
Hachage et URL de certificat X.509	12
Hachage et URL de faisceau X.509	13
Réservé à l'IANA	14 - 200
Usage privé	201 - 255

o Données de certificat (longueur variable) – Codage réel des données de certificat. Le type de certificat est indiqué par le champ Codage de certificat.

Le type de charge utile pour la charge utile de certificat est trente sept (37).

La syntaxe spécifique de certains des codes de type de certificat ci-dessus n'est pas définie dans le présent document. Les types dont la syntaxe est définie ici sont :

Certificat - Signature X.509 (4) contient un certificat X.509 codé en DER dont la clé publique est utilisée pour valider la charge utile AUTH de l'envoyeur.

Liste de révocation de certificat (7) contient une liste de révocation de certificats X.509 codée en DER.

Clé RSA brute (11) contient une clé RSA codée en PKCS n° 1 (voir [RSA] et [RFC3447]).

Les codages de hachage et d'URL (12-13) permettent aux message IKE de rester courts en remplaçant les longues structures de données par un hachage SHA-1 de 20 octets (voir [SHA]) de la valeur remplacée suivi par un URL de longueur variable qui se résout en la structure de données codées en DER elle-même. Cela améliore l'efficacité lorsque les points d'extrémité ont les données de certificat en antémémoire et rend IKE moins sujet aux attaques de déni de service qui deviennent plus faciles à monter quand les messages IKE sont assez gros pour exiger la fragmentation IP [KPS03].

Utiliser la définition ASN.1 suivante pour un faisceau X.509 :

```
CertBundle
        { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
          id-mod-cert-bundle(34) }

        DEFINITIONS EXPLICIT TAGS ::=
        BEGIN

        IMPORTS
          Certificate, CertificateList
          FROM PKIX1Explicit88
            { iso(1) identified-organization(3) dod(6)
              internet(1) security(5) mechanisms(5) pkix(7)
              id-mod(0) id-pkix1-explicit(18) } ;

       CertificateOrCRL ::= CHOICE {
         cert [0] Certificate,
         crl  [1] CertificateList }

       CertificateBundle ::= SEQUENCE OF CertificateOrCRL
       END
```

Les mises en œuvre DOIVENT être capables d'être configurées pour envoyer et accepter jusqu'à quatre certificats X.509 à l'appui de l'authentification, et DOIVENT aussi être capables d'être configurées pour envoyer et accepter les deux premiers formats de hachage et d'URL (avec des URL HTTP). Les mises en œuvre DEVRAIENT être capables d'être configurées pour envoyer et accepter des clés RSA brutes. Si plusieurs certificats sont envoyés, le premier certificat DOIT contenir les clés publiques utilisées pour signer la charge utile AUTH. Les autres certificats peuvent être envoyés dans n'importe quel ordre.

3.7 Charge utile de demande de certificat

La charge utile Demande de certificat, notée CERTREQ dans le présent mémoire, fournit le moyen de demander les certificats préférés via IKE et peut apparaître dans la réponse IKE_INIT_SA et/ou la demande IKE_AUTH. Les charges utiles de demande de certificat PEUVENT être incluses dans un échange lorsque l'envoyeur a besoin d'obtenir le certificat du receveur. Si plusieurs autorités de certification sont de confiance et si le codage de certificat ne permet pas une liste, plusieurs charges utiles de demande de certificat DEVRAIENT être transmises.

La charge utile Demande de certificat se définit comme suit :

Figure 13 : Format de charge utile Demande de certificat

o Codage de certificat (1 octet) - Contient un codage du type ou format du certificat demandé. La liste des valeurs figure au paragraphe 3.6.

o Autorité de certification (longueur variable) - Contient un codage d'une autorité de certification acceptable pour le type de certificat demandé.

Le type de charge utile pour la charge utile Demande de certificat est trente huit (38).

Le champ codage de certificat a les mêmes valeurs que celles définies au paragraphe 3.6. Le champ Autorité de certification contient un indicateur des autorités de confiance pour ce type de certificat. La valeur de Autorité de certification est une liste concaténée de hachages SHA-1 des clés publiques des autorités de certification (CA) de confiance. Chacune est codée comme hachage SHA-1 de l'élément Information de clé publique sujette *(Subject Public Key Info)* (voir au paragraphe 4.1.2.7 de la RFC 3280) à partir de chaque certificat d'ancre de confiance. Les hachages de vingt octets sont enchaînés et inclus sans autre formatage.

Noter que le terme "Demande de certificat" est quelque peu trompeur, en ce que des valeurs autres que de certificats sont définies dans une charge utile de "Certificat" et les demandes pour ces valeurs peuvent être présentes dans une charge utile de demande de certificat. La syntaxe de la charge utile de demande de certificat pour de tels cas n'est pas définie ici.

La charge utile de demande de certificat est traitée en inspectant le champ "Codage de certificat" pour déterminer si le processeur dispose de certificats de ce type. Si c'est le cas, le champ "Autorité de certification" est inspecté pour déterminer si le processeur dispose d'un certificat qui puisse être validé jusqu'à une des autorités de certification spécifiées. Cela peut être une chaîne de certificats.

Si un certificat d'entité terminale existe, qui satisfasse aux critères spécifiés dans le CERTREQ, un certificat ou chaîne de certificats DEVRAIT être renvoyé au demandeur de certificat si le receveur du CERTREQ :

- est configuré pour utiliser l'authentification de certificat,
- est autorisé à envoyer une charge utile CERT,
- a une politique d'autorité de certificat de confiance qui gouverne la négociation en cours, et
- a au moins un chaînage de certificat d'entité terminale approprié en temps et en usage avec une autorité de certificat fournie dans le CERTREQ.

La vérification des révocations de certificat doit être envisagée durant le processus de chaînage utilisé pour choisir un certificat. Noter que même si deux homologues sont configurés pour utiliser deux autorités de certificat différentes, les relations de certification croisée devraient être acceptées par une logique de choix appropriée.

Le but n'est pas d'empêcher la communication par la stricte adhésion au choix d'un certificat fondé sur CERTREQ, alors qu'un certificat de remplacement pourrait être choisi par l'envoyeur qui permettrait aussi au receveur de réussir à le valider en confiance au moyen de la certification croisée, des CRL, ou d'autres moyens configurés hors bande. Et donc, le traitement d'un CERTREQ devrait être vu comme la suggestion d'un choix de certificat plutôt qu'une obligation. S'il n'existe pas de certificat, le CERTREQ est alors ignoré. Ceci n'est pas une condition d'erreur du protocole. Il peut y avoir des cas où il y a une CA préférée envoyée dans le CERTREQ, mais une autre pourrait être acceptable (peut-être après appel à un opérateur humain).

3.8 Charge utile d'authentification

La charge utile d'authentification, notée AUTH dans le présent mémo, contient des données utilisées pour les besoins de l'authentification. La syntaxe des données d'authentification varie conformément à la méthode d'authentification comme spécifié ci-dessous.

La charge utile d'authentification est définie comme suit :

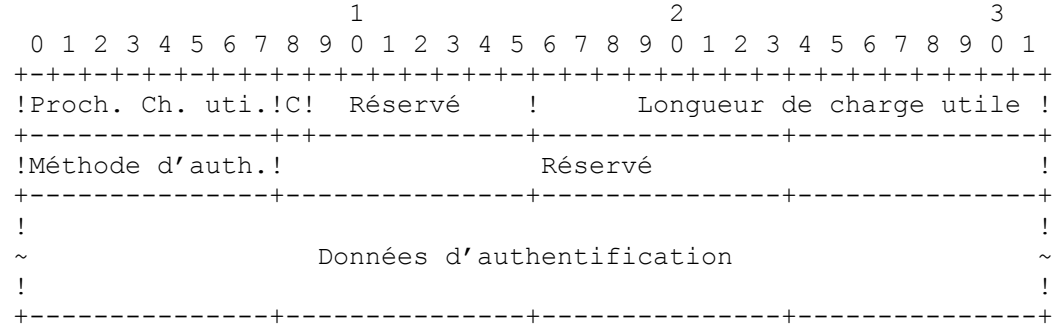


Figure 14 : Format de charge utile d'authentification

o Méthode d'authentif. (1 octet) – Spécifie la méthode d'authentification utilisée. Les valeurs définies sont :

Signature numérique RSA (1) – Calculée comme spécifié au paragraphe 2.15 en utilisant une clé privée RSA sur un hachage bourré de PKCS n°1 (VOIR [RSA] et [RFC3447]).

Code d'intégrité de message à clé partagée (2) – Calculé comme spécifié au paragraphe 2.15 en utilisant la clé partagée associée à l'identité dans la charge utile d'identifiant et la fonction prf négociée.

Signature numérique DSS (3) – Calculée comme spécifié au paragraphe 2.15 en utilisant une clé privée DSS (voir [DSS]) sur un hachage SHA-1.

Les valeurs 0 et 4 à 200 sont réservées à l'IANA. Les valeurs 201 à 255 sont disponibles pour usage privé.

o Données d'authentification (longueur variable) - voir au paragraphe 2.15.

Le type de charge utile pour la charge utile d'authentification est trente neuf (39).

3.9 Charge utile de nom occasionnel

La charge utile de nom occasionnel, notée Ni et Nr dans le présent mémoire pour le nom occasionnel respectivement de l'initiateur et du répondant, contient des données aléatoires utilisées pour garantir le maintien en vie durant un échange et protéger contre les attaques en répétition.

La charge utile de nom occasionnel se définit comme suit :

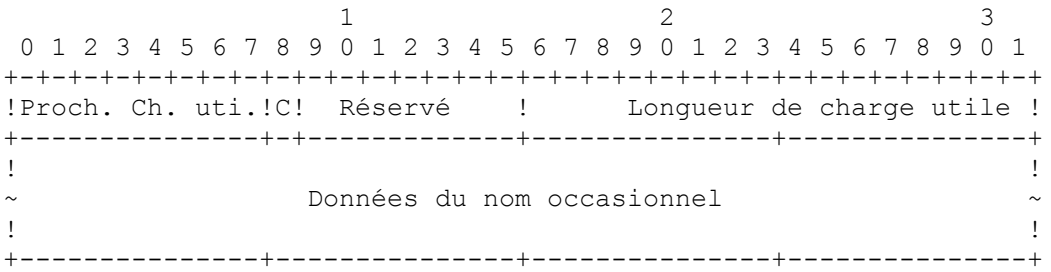


Figure 15 : Format de charge utile de nom occasionnel

o Données de nom occasionnel (longueur variable) – Contient les données aléatoires générées par l'entité émettrice.

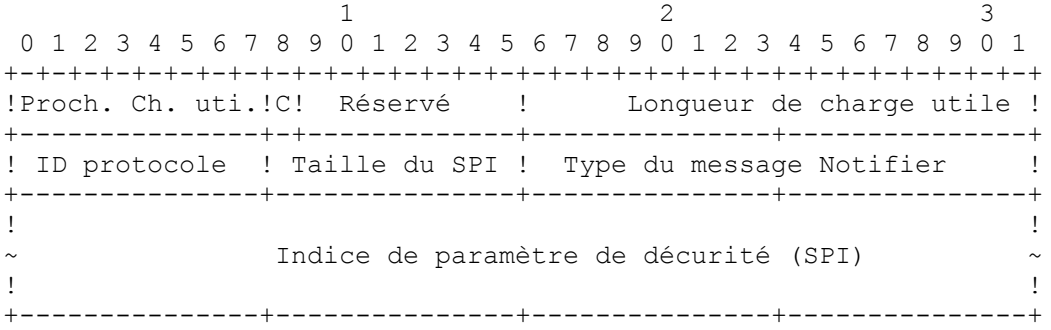
Le type de charge utile pour la charge utile de nom occasionnel est quarante (40).

La taille d'un nom occasionnel DOIT être entre 16 et 256 octets inclus. Les valeurs de nom occasionnel NE DOIVENT PAS être réutilisées.

3.10 Charge utile de Notification

La charge utile Notification, notée N dans le présent document, est utilisée pour transmettre des données d'information, telles que les conditions d'erreur et les transitions d'état, à un homologue IKE. Une charge utile Notification peut apparaître dans un message de réponse (spécifiant habituellement pourquoi une demande a été rejetée), dans un échange INFORMATIONAL (pour rapporter une erreur en dehors d'une demande IKE), ou dans tout autre message pour indiquer les capacités de l'envoyeur ou pour modifier la signification de la demande.

La charge utile Notification se définit comme suit :



```
!                                                               !
~                  Données de notification                      ~
!                                                               !
+---------------+---------------+---------------+---------------+
```

Figure 16 : Format de charge utile Notification

- o ID de protocole (1 octet) – Si cette notification concerne une SA existante, ce champ indique le type de cette SA. Pour les notifications IKE_SA, ce champ DOIT être un (1). Pour les notifications concernant les SA IPsec, ce champ DOIT contenir (2) pour indiquer AH ou (3) pour indiquer ESP. Pour les notifications qui ne se rapportent pas à une SA existante, ce champ DOIT être mis à zéro et DOIT être ignoré à réception. Toutes les autres valeurs de ce champ sont réservées à l'IANA pour de futures allocations.

- o Taille de SPI (1 octet) – Longueur en octets du SPI tel que défini par l'identifiant de protocole IPsec ou zéro si aucun SPI n'est applicable. Pour une notification concernant la IKE_SA, la taille de SPI DOIT être zéro.

- o Type de Message Notify (2 octets) – Spécifie le type de message de notification.

- o SPI (longueur variable) –I ndice de paramètre de sécurité.

- o Données de notification (longueur variable) – Données d'information ou d'erreur transmises en plus du type de message Notify. Les valeurs pour ce champ sont spécifiques du type (voir ci-dessous).

Le type de charge utile pour la charge utile de notification est quarante et un (41).

3.10.1 Types de message Notify

Les informations de notification peuvent être des messages d'erreur qui spécifient pourquoi une SA ne pourrait être établie. Elles peuvent aussi être des données d'état qu'un processus de gestion d'une base de données de SA souhaite communiquer à un processus homologue. Le tableau ci-dessous fait la liste des messages de notification et de leurs valeurs correspondantes. Le numéro des différents états d'erreur a été considérablement réduit depuis IKEv1 à la fois dans un souci de simplification et pour éviter de donner des informations de configuration d'information aux sondeurs.

Les types qui sont dans la gamme de 0 à 16 383 sont destinés à rapporter les erreurs. Une mise en œuvre recevant une charge utile Notify avec un de ces types qu'elle ne reconnaît pas dans une réponse DOIT supposer que la demande correspondante a échoué entièrement. Les types d'erreur non reconnus dans une demande et les types d'état dans une demande ou réponse DOIVENT être ignorés sauf qu'ils DEVRAIENT être enregistrés dans un journal.

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, elles sont utilisées pour négocier les paramètres non cryptographiques.

Messages NOTIFY – Types d'erreur	**Valeur**
Réservé	0
UNSUPPORTED_CRITICAL_PAYLOAD *(charge utile critique non acceptée)*	1 Envoyé si la charge utile a le bit "critique" mis et si le type de charge utile n'est pas reconnu. Les données de notification contiennent le type de charge utile d'un octet.
INVALID_IKE_SPI *(SPI IKE invalide)*	4 Indique qu'un message IKE a été reçu avec un SPI de destination inconnue. Cela indique habituellement que le receveur s'est réinitialisé et a oublié l'existence d'une IKE_SA.
INVALID_MAJOR_VERSION *(version majeure invalide)*	5 Indique que le receveur ne peut pas traiter la version d'IKE spécifiée dans l'en-tête. Le plus proche numéro de version que le receveur peut accepter sera dans l'en-tête de réponse.

INVALID_SYNTAX *(syntaxe invalide)*	7 Indique que le message IKE reçu est invalide parce que un type, longueur, ou valeur est hors gamme ou parce que la demande a été rejetée pour des raisons de politique. Pour éviter une attaque de déni de service utilisant des messages falsifiés, cet état ne peut être retourné que pour et dans un paquet chiffré si l'ID de message et la somme de contrôle cryptographique étaient valides. Pour éviter des fuites d'informations au profit d'un sondeur du nœud, cet état DOIT être envoyé en réponse à toute erreur non couverte par un des autres types d'état. Pour aider à la recherche d'erreurs, des informations d'erreur détaillées DEVRAIENT être écrites à la console ou dans le journal de bord.
INVALID_MESSAGE_ID *(identifiant de message invalide)*	9 Envoyé lorsque est reçu un identifiant de message IKE en dehors de la fenêtre acceptée. Cette notification NE DOIT PAS être envoyée dans une réponse ; il NE DOIT PAS y avoir d'accusé de réception de la demande invalide. On informe à la place l'autre côté en initialisant un échange INFORMATIONAL avec les données de notification qui contiennent les quatre octets d'identifiant du message invalide. L'envoi de cette notification est facultatif, et les notifications de ce type DOIVENT être limitées en débit.
INVALID_SPI *(indice de paramètre de sécurité invalide)*	11 PEUVENT être envoyés dans un échange IKE INFORMATIONAL lorsque un nœud reçoit un paquet ESP ou AH avec un SPI invalide. Les données de notification contiennent le SPI du paquet invalide. Cela indique habituellement qu'un nœud s'est réinitialisé et a oublié une SA. Si ce message d'information est envoyé en dehors du contexte d'une IKE_SA, il ne devrait être utilisé par le receveur que comme une "alerte" sur une anomalie possible (parce que ce pourrait facilement être une manipulation).
NO_PROPOSAL_CHOSEN *(aucune proposition choisie)*	14 Aucune des suites de chiffrement proposées n'est acceptable.
INVALID_KE_PAYLOAD *(charge utile KE invalide)*	17 Le champ n° de groupe D-H dans la charge utile KE n'est pas le n° de groupe choisi par le répondant pour cet échange. Il y a deux octets de données associées à cette notification : le n° de groupe D-H accepté, en ordre gros boutien.
AUTHENTICATION_FAILED *(échec d'authentification)*	24 Envoyé dans la réponse à un message IKE_AUTH lorsque l'authentification a échoué pour une raison quelconque. Il n'y a pas de données associées.
SINGLE_PAIR_REQUIRED *(une seule paire exigée)*	34 Cette erreur indique qu'une demande CREATE_CHILD_SA est inacceptable parce que son envoyeur veut accepter seulement des sélecteurs de trafic spécifiant une seule paire d'adresses. On s'attend à ce que le demandeur réponde en demandant une SA pour le seul trafic spécifique qu'il essaye de transmettre.
NO_ADDITIONAL_SAS *(pas de SA supplémentaire)*	35 Cette erreur indique qu'une demande CREATE_CHILD_SA est inacceptable parce que le répondant ne veut pas accepter d'autres CHILD_SA sur cette IKE_SA. Certaines mises en œuvre minimales peuvent n'accepter l'établissement que d'une seule CHILD_SA dans le contexte d'un échange IKE initial et rejeter toute tentative d'ajout ultérieur.
INTERNAL_ADDRESS_FAILURE *(échec d'adresse interne)*	36 Indique une erreur d'allocation d'adresse interne (c'est-à-dire, INTERNAL_IP4_ADDRESS ou INTERNAL_IP6_ADDRESS) durant le traitement d'une charge utile de configuration par un répondant. Si cette erreur est générée au sein d'un échange IKE_AUTH, aucune CHILD_SA ne sera créée.
FAILED_CP_REQUIRED *(exige échec de charge utile de config.)*	37 Envoyé par le répondant dans le cas où CP(CFG_REQUEST) est attendu mais pas reçu, et donc en conflit avec la politique locale configurée. Il n'y a pas de données associées.
TS_UNACCEPTABLE *(sélecteur de trafic inacceptable)*	38 Indique qu'aucune des adresses/protocoles/accès des sélecteurs de trafic fournis n'est acceptable.

INVALID_SELECTORS *(sélecteurs invalides)*	39 PEUT être envoyé dans un échange IKE INFORMATIONAL lorsqu'un nœud reçoit un paquet ESP ou AH dont les sélecteurs de trafic ne correspondent pas à ceux de la SA sur laquelle il a été livré (et cela a causé l'abandon du paquet). Les données de notification contiennent le début du paquet en cause (comme dans les messages ICMP) et le champ SPI de la notification est réglé de façon à correspondre au SPI de la SA IPsec.
Réservé à l'IANA – Types d'erreur	40 – 8 191
Usage privé – Erreurs	8 192 – 16 383
Messages NOTIFY – Types d'état	**Valeur**
INITIAL_CONTACT	16 384 Cette notification affirme que cette IKE_SA est la seule IKE_SA actuellement active entre les identités authentifiées. Elle PEUT être envoyée quand une IKE_SA est établie après une défaillance, et le receveur PEUT utiliser cette information pour supprimer toutes les autres IKE_SA qu'il a pour la même identité authentifiée sans attendre une fin de temporisation. Cette notification NE DOIT PAS être envoyée par une entité qui peut être dupliquée (par exemple, des accréditifs d'utilisateur en itinérance lorsque l'usager est autorisé à se connecter au pare-feu d'entreprise à partir de deux systèmes distants au même moment).
SET_WINDOW_SIZE	16 385 Cette notification affirme que le point d'extrémité d'envoi est capable de garder l'état pour plusieurs échanges en cours, permettant au receveur d'envoyer plusieurs demandes avant d'obtenir une réponse à la première. Les données associées à une notification SET_WINDOW_SIZE DOIVENT être longues de 4 octets et contenir la représentation en gros boutien du nombre de messages que l'envoyeur promet de garder. La taille de fenêtre est toujours de un jusqu'à la fin de l'échange initial.
ADDITIONAL_TS_POSSIBLE	16 386 Cette notification affirme que le point d'extrémité d'envoi a rétréci les sélecteurs de trafic proposés mais que d'autres sélecteurs de trafic seraient aussi acceptables, quoique seulement dans une SA distincte (voir au paragraphe 2.9). Il n'y a pas de données associées à ce type de notification. Elle ne peut être envoyée que comme charge utile supplémentaire dans un message incluant les TS acceptés.
IPCOMP_SUPPORTED	16 387 Cette notification ne peut être incluse que dans un message contenant une charge utile de SA négociant une CHILD_SA et indique la volonté de l'envoyeur d'utiliser IPComp sur cette SA. Les données associées à cette notification incluent un CPI IPComp de deux octets suivi par un identifiant de transformation d'un octet suivi facultativement par des attributs dont la longueur et le format sont définis par cet identifiant de transformation. Un message proposant une SA peut contenir plusieurs notifications IPCOMP_SUPPORTED pour indiquer plusieurs algorithmes acceptés. Un message acceptant une SA peut en contenir au plus une.

Les identifiants de transformation actuellement définis sont :

Nom	**Numéro**	**Défini dans**
Réservé	0	
IPCOMP_OUI	1	
IPCOMP_DEFLATE	2	RFC 2394
IPCOMP_LZS	3	RFC 2395
IPCOMP_LZJH	4	RFC 3051

Les valeurs 5 à 240 sont réservées à l'IANA.

Les valeurs 241 à 255 sont pour utilisation privée entre des parties mutuellement consentantes.

NAT_DETECTION_SOURCE_IP *(détection de NAT sur la source IP)*	16 388 Cette notification est utilisée par son receveur pour déterminer si la source est derrière un NAT. Les données associées à cette notification sont un résumé SHA-1 des SPI (dans l'ordre dans lequel ils apparaissent dans l'en-tête), de l'adresse IP, et de l'accès sur lequel ce paquet a été envoyé. Il PEUT y avoir plusieurs charges utiles Notify de ce type dans un message si l'envoyeur ne sait pas lequel des rattachements réseau sera utilisé pour envoyer le paquet. Le receveur de cette notification PEUT comparer la valeur fournie au hachage SHA-1 des SPI, adresse IP de source, et accès, et si ils ne correspondent pas, il DEVRAIT activer la traversée de NAT (voir au paragraphe 2.23). Autrement, il PEUT rejeter la tentative de connexion si la traversée de NAT n'est pas acceptée.
NAT_DETECTION_DESTINATION_IP *(détection de NAT sur la destination IP)*	16 389 Cette notification est utilisée par son receveur pour déterminer si elle est derrière un NAT. Les données associées à cette notification sont un résumé SHA-1 des SPI (dans l'ordre dans lequel ils apparaissent dans l'en-tête), de l'adresse IP, et de l'accès auquel le paquet est envoyé. Le receveur de cette notification PEUT comparer la valeur fournie à un hachage des SPI, de l'adresse de destination IP, et de l'accès, et si ils ne correspondent pas, il DEVRAIT invoquer la traversée de NAT (voir au paragraphe 2.23). Si il s ne correspondent pas, cela signifie que cette extrémité est derrière un NAT et cette extrémité DEVRAIT commencer l'envoi de paquets de maintien en vie comme défini dans la [RFC3948]. Autrement, il PEUT rejeter la tentative de connexion si la traversée de NAT n'est pas acceptée.
COOKIE *(témoin)*	16 390 Cette notification PEUT être incluse dans une réponse IKE_SA_INIT. Elle indique que la demande devrait être réessayée avec une copie de cette notification comme première charge utile. Cette notification DOIT être incluse dans un nouvel essai de demande IKE_SA_INIT si une notification COOKIE était incluse dans la réponse initiale. Les données associées à cette notification DOIVENT être entre 1 et 64 (inclus) octets de long.
USE_TRANSPORT_MODE *(utiliser le mode transport)*	16 391 Cette notification PEUT être incluse dans un message de demande qui inclut aussi une charge utile SA demandant une CHILD_SA. Elle demande que la CHILD_SA utilise le mode transport plutôt que le mode tunnel pour la SA créée. Si la demande est acceptée, la réponse DOIT aussi inclure une notification du type USE_TRANSPORT_MODE. Si le répondant refuse la demande, la CHILD_SA sera établie en mode tunnel. Si ce n'est pas acceptable pour l'initiateur, celui-ci DOIT supprimer la SA. Note : Excepté lors de l'utilisation de cette option pour négocier le mode transport, toutes les CHILD_SA vont utiliser le mode tunnel. Note : Les modifications de désencapsulation ECN spécifiées dans la [RFC4301] DOIVENT être effectuées pour chaque SA en mode tunnel créée par IKEv2.
HTTP_CERT_LOOKUP_SUPPORTED *(recherche de certificat HTTP acceptée)*	16 392 Cette notification PEUT être incluse dans tout message qui peut inclure une charge utile CERTREQ et indiquer que l'envoyeur est capable de rechercher des certificats sur la base d'URL fondés sur HTTP (et donc supposé préférer recevoir des spécifications de certificat dans ce format).
REKEY_SA *(renouvellement de clés de SA)*	16 393 Cette notification DOIT être incluse dans un échange CREATE_CHILD_SA si l'objet de l'échange est de remplacer une SA ESP ou AH existante. Le champ SPI identifie la SA dont on renouvelle les clés. Il n'y a pas de données.
ESP_TFC_PADDING_NOT_SUPPORTED *(bourrage TFC EFC non accepté)*	16 394 Cette notification affirme que le point d'extrémité d'envoi NE VA PAS accepter de paquets contenant le bourrage de confidentialité de flux (TFC).
NON_FIRST_FRAGMENTS_ALSO *(aussi des fragments non premiers)*	16 395 Utilisé pour le contrôle de fragmentation. Voir l'explication dans la [RFC4301].

Réservé à l'IANA – Types d'état	16 396 – 40 959
Usage privé - Types d'état	40960 – 65535

3.11 Charge utile Delete

La charge utile Delete *(supprimer)*, notée D dans le présent mémoire, contient un identifiant d'association de sécurité spécifique du protocole que l'envoyeur a retiré de sa base de données d'association de sécurité et n'est donc plus valide. La Figure 17 montre le format de la charge utile Delete. Il est possible d'envoyer plusieurs SPI dans une charge utile Delete ; cependant, chaque SPI DOIT être pour le même protocole. Le mélange d'identifiants de protocole NE DOIT PAS être effectué dans une charge utile Delete. Il est cependant permis d'inclure plusieurs charges utiles Delete dans un seul échange INFORMATIONAL où chaque charge utile Delete fait la liste des SPI pour un protocole différent.

La suppression de la IKE_SA est indiquée par un ID de protocole de 1 (IKE) mais pas de SPI. La suppression d'une CHILD_SA, telle que ESP ou AH, contiendra l'ID de protocole IPsec de ce protocole (2 pour AH, 3 pour ESP), et le SPI est celui que le point d'extrémité d'envoi attendrait dans les paquets ESP ou AH entrants.

La charge utile Delete se définit comme suit :

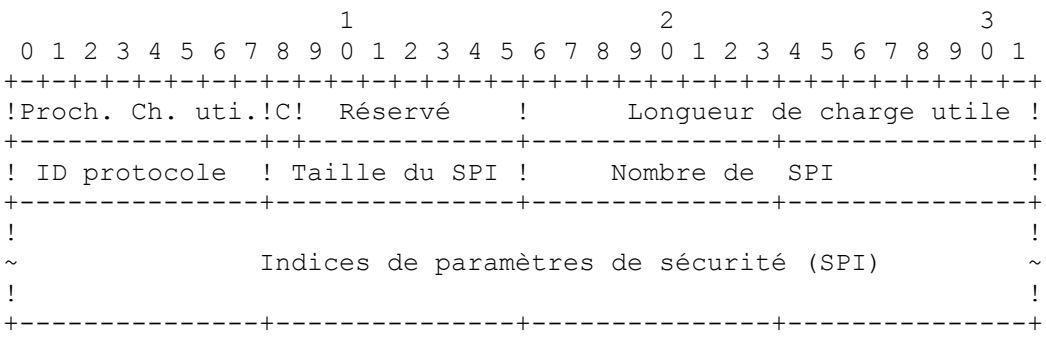


Figure 17 : Format de charge utile Delete

- ID de protocole (1 octet) - Doit être 1 pour une IKE_SA, 2 pour AH, ou 3 pour ESP.

- Taille de SPI (1 octet) – Longueur en octets du SPI comme défini par l'ID du protocole. Il DOIT être zéro pour IKE (le SPI est dans l'en-tête de message) ou quatre pour AH et ESP.

- Nombre de SPI (2 octets) - Le nombre de SPI contenus dans la charge utile Delete. La taille de chaque SPI est définie par le champ Taille de SPI.

- Indice(s) de paramètre de sécurité (SPI) (longueur variable) – Identifie la ou les associations de sécurité spécifiques à supprimer. La longueur de ce champ est déterminée par les champs Taille de SPI et Nombre de SPI.

Le type de charge utile pour la charge utile Delete est quarante deux (42).

3.12 Charge utile d'identifiant de fabricant

La charge utile ID de fabricant, notée V dans le présent mémoire, contient une constante définie par le fabricant. La constante est utilisée par le fabricant pour identifier et reconnaître à distance ses instances de mise en œuvre. Ce mécanisme permet à un fabricant d'expérimenter de nouveaux dispositifs tout en maintenant la rétro compatibilité.
Une charge utile ID de fabricant PEUT annoncer que l'envoyeur est capable d'accepter certaines extensions du protocole, ou elle PEUT simplement identifier la mise en œuvre pour une aide à la recherche d'erreurs.
Plusieurs charges utiles ID de fabricant PEUVENT être envoyées. Une mise en œuvre N'EST PAS OBLIGÉE d'envoyer du tout de charge utile ID de fabricant.
Une charge utile ID de fabricant peut être envoyée au titre de tout message. La réception d'une charge utile ID de fabricant familière permet à une mise en œuvre de faire usage des numéros à usage privé décrits tout au long du présent mémoire -- charges utiles privées, échanges privés, notifications privées, etc. Les identifiants de fabricant non familiers DOIVENT être ignorés.
Les auteurs de projets Internet qui souhaitent étendre le présent protocole DOIVENT définir une charge utile ID de fabricant pour annoncer la capacité à mettre en œuvre l'extension dans le projet Internet. Il est prévu que les projets Internet qui sont acceptés et sont normalisés recevront des "numéros magiques" hors de la gamme Utilisation future de la part de l'IANA, et l'exigence d'utiliser un identifiant de fabricant sera abandonnée.

Les champs de charge utile ID de fabricant sont définis comme suit :

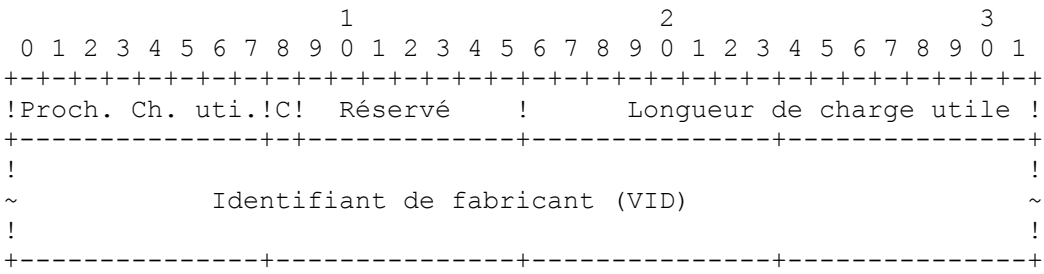


Figure 18 : Format de charge utile d'identifiant de fabricant

o Identifiant de fabricant (longueur variable) – Il est de la responsabilité de la personne qui choisit l'identifiant de fabricant de s'assurer de son unicité en dépit de l'absence de tout registre central des identifiants. Il est de bonne pratique d'inclure le nom de la compagnie, un nom de personne, ou quelque chose comme cela. Si vous voulez épater, vous pouvez inclure la latitude, la longitude et l'heure qu'il était quand vous avez choisi l'identifiant et un nombre aléatoire. Un résumé de message en une longue chaîne unique est préférable à la longue chaîne unique elle-même.

Le type de charge utile pour la charge utile d'identifiant de fabricant est quarante trois (43).

3.13 Charge utile de sélecteur de trafic

La charge utile de sélecteur de trafic, notée TS dans le présent mémoire, permet aux homologues d'identifier les flux de paquet pour le traitement par les services de sécurité IPsec. La charge utile de sélecteur de trafic comporte l'en-tête générique IKE de charge utile suivi par les sélecteurs de trafic individuels comme suit :

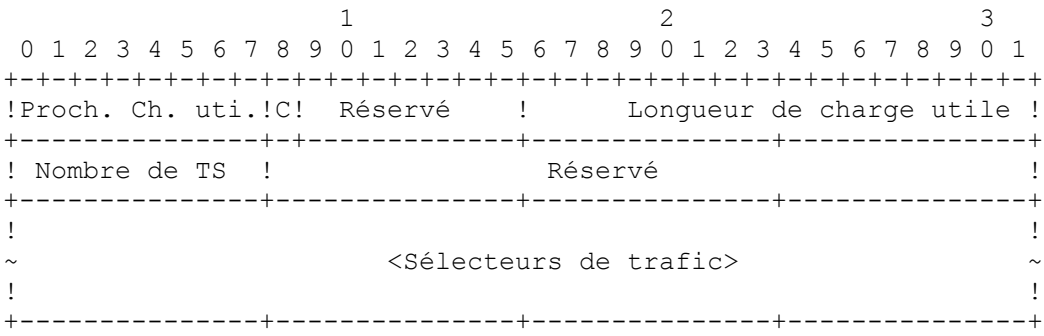


Figure 19 : Format de charge utile de sélecteur de trafic

o Nombre de TS (1 octet) – Nombre de sélecteurs de trafic fournis.

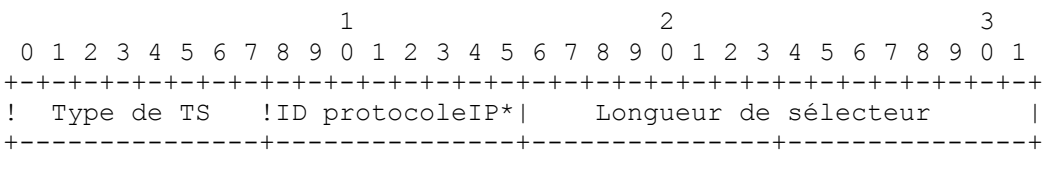
o Réservé – Ce champ DOIT être envoyé à zéro et DOIT être ignoré à réception.

o Sélecteurs de trafic (longueur variable) - Un ou plusieurs sélecteurs de trafic individuels.

La longueur de la charge utile de sélecteur de trafic inclut l'en-tête TS et tous les sélecteurs de trafic.

Le type de charge utile pour la charge utile de sélecteur de trafic est quarante quatre (44) pour les adresses du côté initiateur de la SA et quarante cinq (45) pour les adresses du côté répondant.

3.13.1 Sélecteur de trafic



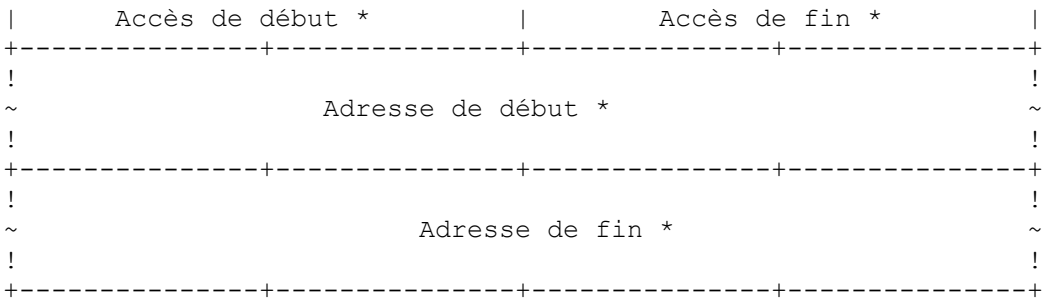


Figure 20 : Sélecteur de trafic

* Note : Tous les champs autres que Type de TS et Longueur de sélecteur dépendent du Type de TS. Le champ montré est pour les types de TS 7 et 8, deux seules valeurs actuellement définies.

- Type de TS (un octet) – Spécifie le type de sélecteur de trafic.
- ID de protocole IP (1 octet) - Valeur qui spécifie un ID de protocole IP associé (par exemple, UDP/TCP/ICMP). Une valeur de zéro signifie que l'ID de protocole n'est pas pertinent pour ce sélecteur de trafic -- la SA peut porter tous les protocoles.
- Longueur de sélecteur – Spécifie la longueur de cette sous structure de sélecteur de trafic y compris l'en-tête.
- Accès de début (2 octets) – Valeur qui spécifie le plus petit numéro de l'accès admis par ce sélecteur de trafic. Pour les protocoles pour lesquels l'accès est indéfini, ou si tous les accès sont permis, ce champ DOIT être zéro. Pour le protocole ICMP, les deux champs Type et Code d'un octet sont traités comme un seul numéro d'accès entier de 16 bits (avec Type dans les huit bits de plus fort poids et Code dans les huit bits de moindre poids) pour les besoins du filtrage fondé sur ce champ.
- Accès de fin (2 octets) - Valeur qui spécifie le plus grand numéro d'accès admis par ce sélecteur de trafic. Pour les protocoles pour lesquels l'accès est indéfini, ou si tous les accès sont permis, ce champ DOIT être 65 535. Pour le protocole ICMP, les deux champs Type et Code d'un octet sont traités comme un seul numéro d'accès entier de 16 bits (avec Type dans les huit bits de plus fort poids et Code dans les huit bits de moindre poids) pour les besoins du filtrage fondé sur ce champ.
- Adresse de début – La plus petite adresse incluse dans ce sélecteur de trafic (longueur déterminée par le type de TS).
- Adresse de fin – La plus grande adresse incluse dans ce sélecteur de trafic (longueur déterminée par le type de TS).

Les systèmes qui se conforment à la [RFC4301] et souhaitent indiquer "TOUT" accès DOIVENT régler l'accès de début à 0 et l'accès de fin à 65 535 ; noter que conformément à la [RFC4301], "TOUT" inclut "OPAQUE". Les systèmes qui fonctionnent avec la [RFC4301] et souhaitent indiquer un accès "OPAQUE", mais pas "TOUT" accès, DOIVENT régler l'accès de début à 65 535 et l'accès de fin à 0.

Le tableau suivant fait la liste des valeurs allouées pour le champ Type de sélecteur de trafic et les données de sélecteur d'adresse correspondantes.

Type de TS	**Valeur**
Réservé	0 à 6
TS_IPV4_ADDR_RANGE	7 Gamme d'adresses IPv4, représentées par deux valeurs de quatre octets. La première valeur est l'adresse IPv4 de début (incluse) et la seconde valeur est l'adresse IPv4 de fin (incluse). Toutes les adresses tombant entre les deux adresses spécifiées sont considérées comme étant à l'intérieur de la liste.
TS_IPV6_ADDR_RANGE	8 Gamme d'adresses IPv6, représentées par deux valeurs de quatre octets. La première valeur est l'adresse IPv4 de début (incluse) et la seconde valeur est l'adresse IPv6 de fin (incluse). Toutes les adresses tombant entre les deux adresses spécifiées sont considérées comme étant à l'intérieur de la liste.
Réservé à l'IANA	9 à 240
Usage privé	241 à 255

3.14 Charge utile chiffrée

La charge utile chiffrée, notée SK{...} ou E dans le présent mémoire, contient d'autres charges utiles en forme chiffrée. La charge utile Chiffrée, si elle est présente dans un message, DOIT être la dernière charge utile du message. Souvent, elle est la seule charge utile du message.

Les algorithmes de chiffrement et de protection d'intégrité sont négociés durant l'établissement de la IKE_SA, et les clés sont calculées comme spécifié aux paragraphes 2.14 et 2.18.

Les algorithmes de chiffrement et de protection d'intégrité sont modélisés d'après les algorithmes ESP décrits dans les [RFC2104], [RFC4303], et [RFC2451]. Le présent document spécifie entièrement le traitement cryptographique des données IKE, mais ces documents devraient être consultés par les concepteurs d'algorithmes. On exige un bloc de chiffrement d'une taille de bloc fixe et un algorithme de vérification d'intégrité qui calcule une somme de contrôle de longueur fixe sur un message de taille variable.

Le type de charge utile pour une charge utile Chiffrée est quarante six (46). La charge utile chiffrée comporte l'en-tête générique de charge utile IKE suivi par des champs individuels comme suit :

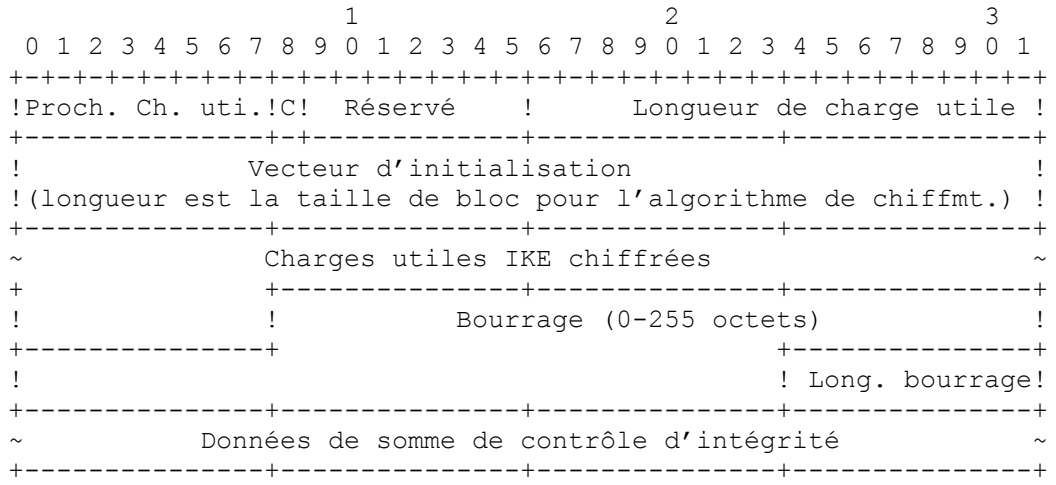


Figure 21 : Format de charge utile Chiffrée

- Prochaine charge utile - Type de charge utile de la première charge utile incorporée. Noter que c'est une exception dans le format d'en-tête standard, car la charge utile Chiffrée est la dernière charge utile du message et donc le champ Prochaine charge utile devrait normalement être à zéro. Mais comme le contenu de cette charge utile est fait de charges utiles incorporées et qu'il n'y a pas d'emplacement naturel où placer le type de la première, ce type est placé ici.

- Longueur de charge utile – Inclut les longueurs de l'en-tête, IV, charges utiles IKE Chiffrées, Bourrage, Longueur de bourrage, et données de somme de contrôle d'intégrité.

- Vecteur d'initialisation - Valeur choisie de façon aléatoire dont la longueur est égale à la longueur de bloc de l'algorithme de chiffrement sous-jacent. Les receveurs DOIVENT accepter toute valeur. Les envoyeurs DEVRAIENT soit prendre cette valeur pseudo-aléatoire et indépendante pour chaque message, soit utiliser le bloc de chiffrement final du message envoyé précédent. Les envoyeurs NE DOIVENT PAS utiliser la même valeur pour chaque message, ni utiliser une séquence de valeurs avec une faible distance de Hamming (par exemple, un numéro de séquence), ou utiliser du texte chiffré d'un message reçu.

- Les charges utiles IKE sont comme spécifié plus haut dans cette section. Ce champ est chiffré avec un chiffre négocié.

- Le bourrage PEUT contenir toute valeur choisie par l'envoyeur, et DOIT avoir une longueur qui combine les charges utiles, le bourrage, et la longueur de bourrage en un multiple de la taille du bloc de chiffrement. Ce champ est chiffré avec le chiffre négocié.

- Longueur de bourrage est la longueur du champ Bourrage. L'envoyeur DEVRAIT régler la longueur de bourrage à la valeur minimum qui fait de la combinaison des charges utiles, du bourrage, et de la longueur de bourrage un multiple de la taille de bloc, mais le receveur DOIT accepter toute longueur qui donne un alignement approprié. Ce champ est chiffré avec le chiffre négocié.

- Données de somme de contrôle d'intégrité est la somme de contrôle cryptographique sur le message entier, qui commence à l'en-tête IKE fixe jusqu'à la Longueur de bourrage. La somme de contrôle DOIT être calculée sur le message chiffré. Sa longueur est déterminée par l'algorithme d'intégrité négocié.

3.15 Charge utile de configuration

La charge utile Configuration, notée CP dans le présent document, est utilisée pour échanger des informations de configuration entre homologues IKE. L'échange sert à un IRAC à demander une adresse IP interne à un IRAS et à échanger d'autres informations de même sorte qu'il pourrait acquérir avec le protocole de configuration d'hôte dynamique (DHCP, *Dynamic Host Configuration Protocol*) si l'IRAC était directement connecté à un LAN.

Les charges utiles Configuration sont du type CFG_REQUEST/CFG_REPLY ou CFG_SET/CFG_ACK (voir le type CFG dans la description de charge utile ci-dessous). Les charges utiles CFG_REQUEST et CFG_SET peuvent facultativement être ajoutées à toute demande IKE. La réponse IKE DOIT inclure une charge utile CFG_REPLY ou CFG_ACK ou Notify correspondante, avec un type d'erreur qui indique pourquoi la demande n'a pas pu être honorée. Une exception est qu'une mise en œuvre minimale PEUT ignorer toutes les charges utiles CFG_REQUEST et CFG_SET, aussi un message de réponse sans CFG_REPLY ou CFG_ACK correspondante DOIT être accepté comme une indication que la demande n'a pas été acceptée.

"CFG_REQUEST/CFG_REPLY" permet à un point d'extrémité IKE de demander des informations à son homologue. Si un attribut dans la charge utile de configuration CFG_REQUEST n'est pas de longueur zéro, il est pris comme une suggestion pour cet attribut. La charge utile de configuration CFG_REPLY PEUT retourner cette valeur, ou une nouvelle. Elle PEUT aussi ajouter de nouveaux attributs et ne pas inclure certains de ceux demandés. Les demandeurs DOIVENT ignorer les attributs retournés qu'ils ne connaissent pas.

Certains attributs PEUVENT être multi-valeurs, auquel cas plusieurs stuctures d'attribut de configuration du même type sont envoyées et/ou retournées. Généralement, toutes les valeurs d'un attribut sont retournées lorsque l'attribut est demandé. Pour certains attributs (seules des adresses internes dans la présent version de la spécification) plusieurs demandes indiquent une demande à laquelle plusieurs valeurs sont allouées. Pour ces attributs, le nombre des valeurs retournées NE DEVRAIT PAS excéder le nombre demandé.

Si le type de données demandées dans une CFG_REQUEST n'est pas reconnu ou pas accepté, le répondant NE DOIT PAS retourner un type d'erreur mais DOIT plutôt, soit envoyer une CFG_REPLY qui PEUT être vide, soit une réponse qui ne contient pas de charge utile CFG_REPLY du tout. Les retours d'erreurs sont réservés aux cas où la demande est reconnue mais ne peut pas être traitée comme demandé ou où la demande est mal formatée.

"CFG_SET/CFG_ACK" permet à un point d'extrémité IKE de pousser les données de configuration jusqu'à son homologue. Dans ce cas, la charge utile de configuration CFG_SET contient des attributs que l'initiateur veut voir son homologue altérer. Le répondant DOIT retourner une charge utile de configuration s'il accepte une des données de configuration et elle DOIT contenir les attributs que le répondant a acceptés avec des données de longueur zéro. Ces attributs qu'il n'a pas acceptés NE DOIVENT PAS être dans la charge utile de configuration CFG_ACK. Si aucun attribut n'a été accepté, le répondant DOIT retourner une charge utile CFG_ACK ou un message de réponse sans charge utile CFG_ACK. Il n'y a actuellement pas d'utilisation définie de l'échange CFG_SET/CFG_ACK, bien qu'il puisse être utilisé en connexion avec des extensions fondées sur les identifiants de fabricant. Une mise en œuvre minimale de la présente spécification PEUT ignorer les charges utiles CFG_SET.

Des extensions via la charge utile CP NE DEVRAIENT PAS être utilisées pour de la gestion tout venant. Sa principale destination est de fournir un mécanisme d'amorçage pour des échanges d'informations au sein d'IPsec d'IRAS à IRAC. Alors qu'il PEUT être utile d'utiliser une telle méthode pour échanger des informations entre certaines passerelles de sécurité ou petits réseaux, les protocoles de gestion existants tels que [RFC2131], [RFC2865], SNMP, ou [RFC2251] devraient être préférés pour la gestion d'entreprise aussi bien que pour les échanges d'information ultérieurs.

La charge utile Configuration est définie comme suit :

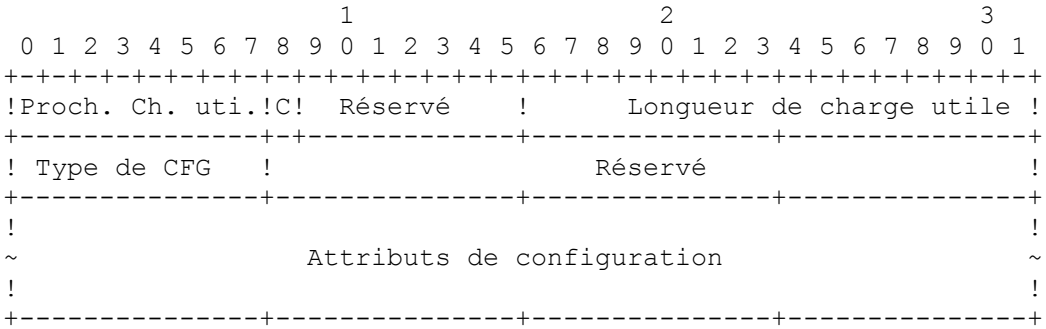


Figure 22 : Format de charge utile Configuration

Le type de charge utile pour la charge utile de configuration est quarante sept (47).

o Type de CFG (1 octet) - Type d'échange représenté par les attributs de configuration.

Type de CFG	**Valeur**
Réservé	0
CFG_REQUEST	1
CFG_REPLY	2
CFG_SET	3
CFG_ACK	4

Les valeurs 5 à 127 sont réservées à l'IANA. Les valeurs 128 à 255 sont pour usage privé entre des parties mutuellement consentantes.

o Réservé (3 octets) - DOIT être envoyé à zéro; DOIT être ignoré à réception.

o Attributs de configuration (longueur) – Ce sont des valeurs de longueur de type spécifiques de la charge utile de configuration et elles sont définies ci-dessous. Il peut y avoir zéro, un ou plusieurs attributs de configuration dans cette charge utile.

3.15.1 Attributs de configuration

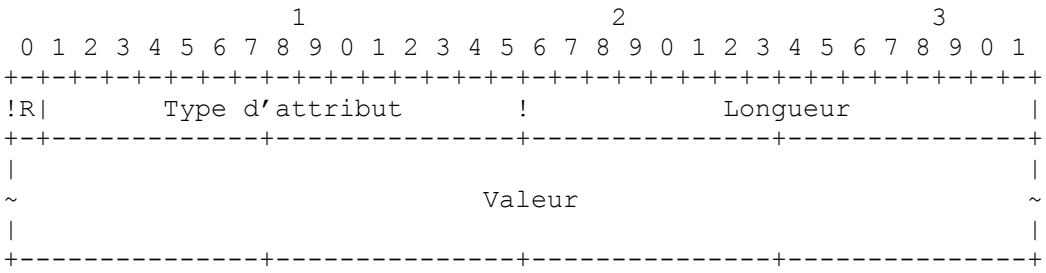


Figure 23 : Format d'attributs de configuration

o Réservé (1 bit) - Ce bit DOIT être mis à zéro et DOIT être ignoré à réception.

o Type d'attribut (15 bits) - Identifiant unique pour chaque type d'attribut de configuration.

o Longueur (2 octets) - Longueur en octets de la valeur.

o Valeur (0 octet ou plus) – Valeur de longueur variable de cet attribut de configuration.

Les types d'attribut suivants ont été définis :

Type d'attribut	**Valeur**	**Multi-valorisé**	**Longueur**
Réservé	0		
INTERNAL_IP4_ADDRESS	1	oui*	0 ou 4 octets
INTERNAL_IP4_NETMASK	2	non	0 ou 4 octets
INTERNAL_IP4_DNS	3	oui	0 ou 4 octets
INTERNAL_IP4_NBNS	4	oui	0 ou 4 octets
INTERNAL_ADDRESS_EXPIRY	5	non	0 ou 4 octets
INTERNAL_IP4_DHCP	6	oui	0 ou 4 octets
APPLICATION_VERSION	7	non	0 ou plus
INTERNAL_IP6_ADDRESS	8	oui*	0 ou 17 octets
Réservé	9		
INTERNAL_IP6_DNS	10	oui	0 ou 16 octets
INTERNAL_IP6_NBNS	11	oui	0 ou 16 octets
INTERNAL_IP6_DHCP	12	oui	0 ou 16 octets
INTERNAL_IP4_SUBNET	13	oui	0 ou 8 octets
SUPPORTED_ATTRIBUTES	14	non	multiple de 2
INTERNAL_IP6_SUBNET	15	oui	17 octets

* Ces attributs ne peuvent être multi-valorisés au retour que si des valeurs multiples étaient demandées.

Les types 16 à 16383 sont réservés à IANA. Les valeurs 16384-32767 sont pour usage privé entre parties mutuellement consentantes.

- INTERNAL_IP4_ADDRESS, INTERNAL_IP6_ADDRESS – Adresse sur le réseau interne, parfois appelée adresse de nœud rouge ou adresse privée et PEUT être une adresse privée sur l'Internet. Dans un message de demande, l'adresse spécifiée est une adresse demandée (ou zéro si aucune adresse spécifique n'est demandée). Si une adresse spécifique est demandée, elle indique vraisemblablement qu'une connexion antérieure existait avec cette adresse et que le demandeur aimerait réutiliser cette adresse. Avec IPv6, un demandeur PEUT fournir les octets d'adresse de faible poids qu'il veut utiliser. Plusieurs adresses internes PEUVENT être demandées en demandant plusieurs attributs d'adresse interne. Le répondant PEUT envoyer au maximum le nombre d'adresses demandées. INTERNAL_IP6_ADDRESS est composé de deux champs : le premier est une adresse IPv6 de seize octets et le second est une longueur de préfixe d'un octet comme défini dans la [RFC3513].

L'adresse demandée n'est valide que jusqu'à l'expiration du délai défini par l'attribut INTERNAL_ADDRESS EXPIRY ou bien il n'y a pas de IKE_SA entre les homologues.

- INTERNAL_IP4_NETMASK – Gabarit de réseau du réseau interne. Un seul gabarit de réseau est admis dans la demande et les messages de réponse (par exemple, 255.255.255.0), et il DOIT être utilisé uniquement avec un attribut INTERNAL_IP4_ADDRESS.

- INTERNAL_IP4_DNS, INTERNAL_IP6_DNS – Spécifie une adresse de serveur DNS au sein du réseau. Plusieurs serveurs DNS PEUVENT être requis. Le répondant PEUT répondre par zéro un ou plusieurs attributs de serveur DNS.

- INTERNAL_IP4_NBNS, INTERNAL_IP6_NBNS - Spécifie une adresse d'un serveur de nom NetBios (WINS) au sein du réseau. Plusieurs serveurs NBNS PEUVENT être requis. Le répondant PEUT répondre avec zéro un ou plusieurs attributs de serveur NBNS.

- INTERNAL_ADDRESS_EXPIRY – Spécifie le nombre de secondes pendant lesquelles l'hôte peut utiliser l'adresse IP interne. L'hôte DOIT renouveler l'adresse IP avant l'expiration de ce délai. Un seul de ces attributs PEUT être présent dans la réponse.

- INTERNAL_IP4_DHCP, INTERNAL_IP6_DHCP – Ordonne à l'hôte d'envoyer toute demande DHCP interne contenue au sein de cet attribut. Plusieurs serveurs DHCP PEUVENT être nécessaires. Le répondant PEUT répondre par zéro un ou plusieurs attributs de serveur DHCP.

- APPLICATION_VERSION – Informations de version ou d'application de l'hôte IPsec. C'est une chaîne de caractères ASCII imprimables qui N'EST PAS terminée par le caractère NUL.

- INTERNAL_IP4_SUBNET – Les sous-réseaux protégés par ce dispositif de bordure. Cet attribut est composé de deux champs : le premier est une adresse IP et le second un gabarit de réseau. Plusieurs sous-réseaux PEUVENT être demandés. Le répondant PEUT répondre avec zéro un ou plusieurs attributs de sous réseau.

- SUPPORTED_ATTRIBUTES – Lorsqu'il est utilisé au sein d'une demande, cet attribut DOIT être de longueur zéro et spécifie une interrogation à laquelle le répondant doit répondre avec tous les attributs qu'il accepte. La réponse contient un attribut qui contient un ensemble d'identifiants d'attribut ayant chacun 2 octets. La longueur divisée par 2 (octets) établit le nombre d'attributs acceptés contenus dans la réponse.

- INTERNAL_IP6_SUBNET - Les sous-réseaux protégés par ce dispositif de bordure. Cet attribut est composé de deux champs : le premier est une adresse IPv6 de seize octets et le second est une longueur de préfixe d'un octet comme défini dans la [RFC3513]. Plusieurs sous-réseaux PEUVENT être demandés. Le répondant PEUT répondre avec zéro, un, ou plusieurs attributs de sous-réseau.

Noter que le présent document ne fait aucune recommandation sur la façon dont une mise en œuvre affiche réellement les informations à envoyer dans une réponse. C'est à dire que nous ne recommandons aucune méthode spécifique selon laquelle un IRAS détermine quel serveur DNS devrait être retourné à un IRAC demandeur.

3.16 Charge utile de protocole d'authentification extensible (EAP)

La charge utile de protocole d'authentification extensible, notée EAP dans le présent mémoire, permet aux IKE_SA d'être authentifiées en utilisant le protocole défini dans la [RFC3748] et les extensions ultérieures à ce protocole. L'ensemble complet des valeurs acceptables pour la charge utile est défini ailleurs, mais un bref résumé de la RFC3748 est inclus ici

pour rendre le présent document autonome dans les cas les plus courants.

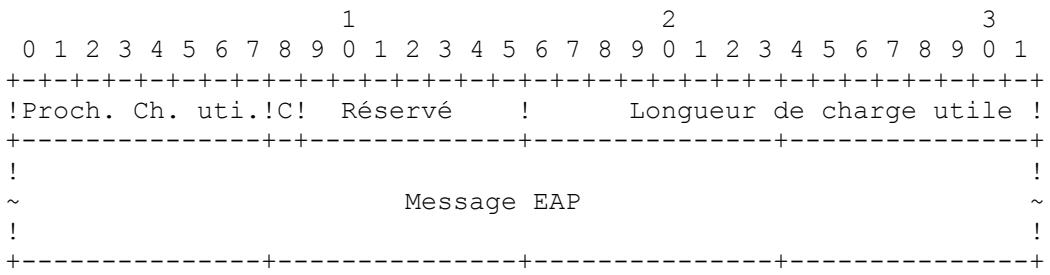


Figure 24 : Format de charge utile EAP

Le type de charge utile pour une charge utile EAP est quarante huit (48).

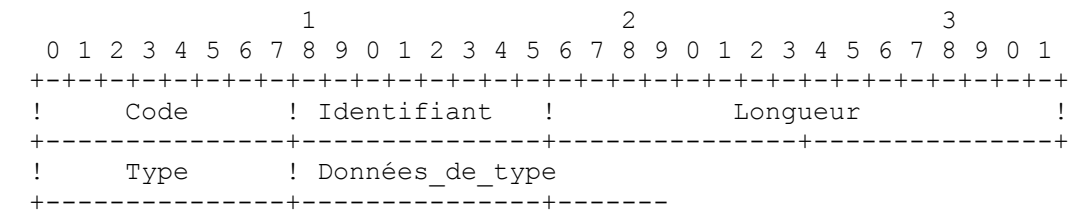


Figure 25 : Format de message EAP

- Code (1 octet) indique si ce message est une Demande (1), Réponse (2), Succès (3), ou Échec (4).

- Identifiant (1 octet) est utilisé dans PPP pour distinguer les messages répétés frauduleusement de ceux répétés normalement. Comme dans IKE, EAP fonctionne sur un protocole fiable, il n'a ici aucune fonction. Dans un message de réponse, cet octet DOIT être établi pour correspondre à l'identifiant dans la demande correspondante. Dans les autres messages, ce champ PEUT être réglé à n'importe quelle valeur.

- Longueur (2 octets) est la longueur du message EAP et DOIT être inférieur de quatre à la Longueur de charge utile de la charge utile qui l'incorpore.

- Type (1 octet) n'est présent que si le champ Code est Demande (1) ou Réponse (2). Pour les autres codes, la longueur de message EAP DOIT être de quatre octets et les champs Type et Données_de_type NE DOIVENT PAS être présents. Dans un message Demande (1), Type indique les données qui sont demandées. Dans un message Réponse (2), Type DOIT être sans accusé de réception ou correspondre au type des données demandées. Les types suivants sont définis dans la RFC3748 :
  1 Identité
  2 Notification
  3 Pas d'accusé de réception (uniquement en réponse)
  4 Défi MD5
  5 Mot de passe à utilisation unique (OTP, *One-Time Password*)
  6 Carte de jeton générique

- Données_de_type (Longueur variable) varie avec le type de demande et la réponse associée. Pour la documentation sur les méthodes d'EAP, voir [RFC3748].

Noter que dans la mesure où IKE passe une indication de l'identité de l'initiateur dans le message 3 du protocole, le répondant NE DEVRAIT PAS envoyer de demande d'identité EAP. L'initiateur DEVRAIT, cependant, répondre à de telles demandes si il en reçoit.

4 Exigences de conformité

Afin de garantir l'interopérabilité de toutes les mises en œuvre de IKEv2, il y a des exigences supplémentaires aux "DOIT prendre en charge" qui sont énumérés ailleurs. Bien sûr, IKEv2 est un protocole de sécurité, et une de ses fonctions majeures est de ne permettre qu'aux parties autorisées de réussir l'établissement complet des associations de sécurité. Ainsi une mise en œuvre particulière peut être configurée avec un nombre quelconque de restrictions concernant les algorithmes

et les autorités de confiance qui empêcheront une interopérabilité universelle.

IKEv2 est conçu pour permettre des mises en œuvre minimales qui peuvent interopérer avec les mises en œuvre conformes à la totalité de ses prescriptions. Il y a une série de dispositifs facultatifs qui peuvent aisément être ignorés par une mise en œuvre particulière si elle ne prend pas en charge ce dispositif. Ces dispositifs sont :
- la capacité à négocier des SA à travers un NAT et à tunneler la SA ESP résultante sur UDP.
- la capacité à demander (et à répondre à une demande) d'adresse IP temporaire à l'extrémité distante d'un tunnel.
- la capacité à prendre en charge divers types d'authentification traditionnels.
- la capacité à prendre en charge des tailles de fenêtre supérieures à un.
- la capacité à établir plusieurs SA ESP et/ou AH au sein d'une seule IKE_SA.
- la capacité à renouveler les clés des SA.

Pour assurer l'interopérabilité, toutes les mises en œuvre DOIVENT être capables d'analyser tous les types de charge utile (éventuellement pour les sauter) et d'ignorer les types de charge utile qu'elles ne prennent pas en charge sauf si le bit critique est établi dans l'en-tête de charge. Si le bit critique est établi dans un en-tête de charge utile non pris en charge, toutes les mises en œuvre DOIVENT rejeter les messages contenant ces charges utiles.

Toute mise en œuvre DOIT être capable d'effectuer les échanges des quatre messages IKE_SA_INIT et IKE_AUTH qui établissent deux SA (un pour IKE, un pour ESP et/ou AH). Les mises en œuvre PEUVENT être en initiation seule ou en réponse seule si c'est approprié pour leur plate-forme. Toute mise en œuvre DOIT être capable de répondre à un échange INFORMATIONAL, mais une mise en œuvre minimale PEUT répondre à tout message INFORMATIONAL par une réponse INFORMATIONAL vide (noter que dans le contexte d'une IKE_SA, un message "vide" consiste en un en-tête IKE suivi par une charge utile Chiffrée sans charge utile à l'intérieur). Une mise en œuvre minimale ne PEUT prendre en charge l'échange CREATE_CHILD_SA que pour autant qu'elle reconnaît les demandes et les rejette avec une charge utile Notify du type NO_ADDITIONAL_SAS Une mise en œuvre minimale n'a pas besoin d'être capable d'initier les échanges CREATE_CHILD_SA ou INFORMATIONAL. Lorsqu'une SA arrive à expiration (sur la base de valeurs configurées localement de durée de vie ou d'octets passés) une mise en œuvre PEUT essayer de la renouveler avec un échange CREATE_CHILD_SA ou elle PEUT supprimer (fermer) la vieille SA et en créer une nouvelle. Si le répondant rejette la demande CREATE_CHILD_SA avec une notification NO_ADDITIONAL_SAS, la mise en œuvre DOIT être capable de clore la vieille SA et d'en créer une nouvelle à la place. Les mises en œuvre ne sont pas obligées de prendre en charge la demande d'adresses IP temporaires ou de répondre à de telles demandes. Si une mise en œuvre n'accepte pas de produire de telles demandes, elle DOIT inclure une charge utile CP dans le message 3, contenant au moins un champ de type INTERNAL_IP4_ADDRESS ou INTERNAL_IP6_ADDRESS. Tous les autres champs sont facultatifs. Si une mise en œuvre accepte de répondre à de telles demandes, elle DOIT analyser la charge utile CP de type CFG_REQUEST dans le message 3 et reconnaître un champ de type INTERNAL_IP4_ADDRESS ou INTERNAL_IP6_ADDRESS. Si elle accepte de prêter une adresse du type approprié, elle DOIT retourner une charge utile CP de type CFG_REPLY contenant une adresse du type demandé. Le répondant DEVRAIT inclure tous les autres attributs qui s'y rapportent s'il en a.

Une mise en œuvre IPv4 répondante minimale ignorera le contenu de la charge utile CP sauf pour déterminer qu'elle comporte un attribut INTERNAL_IP4_ADDRESS et répondra avec l'adresse et les autres attributs qui s'y rapportent que l'initiateur les ait demandés ou non.

Une mise en œuvre IPv4 d'initiateur minimale génèrera une charge utile CP ne contenant qu'un attribut INTERNAL_IP4_ADDRESS et analysera la réponse en ignorant cet attribut si elle ne sait pas comment l'utiliser. Le seul attribut qu'elle DOIT être capable de traiter est INTERNAL_ADDRESS_EXPIRY, qu'elle doit utiliser pour limiter la durée de vie de la SA sauf si elle réussit à renouveler le bail avant son expiration. Les initiateurs minimaux n'ont pas besoin d'être capables de demander un renouvellement de bail et les répondants minimaux n'ont pas besoin d'y répondre.

Pour qu'une mise en œuvre puisse être dite conforme à la présente spécification, il DOIT être possible de la configurer pour qu'elle accepte ce qui suit :
- des certificats PKIX contenant, et signés par, des clés RSA d'une taille de 1024 ou 2048 bits, où l'identifiant passé est un ID_KEY_ID, ID_FQDN, ID_RFC822_ADDR, ou ID_DER_ASN1_DN.
- une authentification de clé partagée où l'identifiant passé est un ID_KEY_ID, ID_FQDN, ou une ID_RFC822_ADDR.
- une authentification où le répondant est authentifié en utilisant des certificats PKIX et où l'initiateur est authentifié en utilisant l'authentification de clé partagée.

5 Considérations sur la sécurité

Bien que ce protocole soit conçu pour minimiser la divulgation des informations de configuration aux homologues non authentifiés, certaines divulgations sont inévitables. Un homologue ou l'autre doit s'identifier le premier et prouver d'abord son identité. Pour éviter les investigations, l'initiateur d'un échange est obligé de s'identifier en premier, et

habituellement il est aussi obligé de s'authentifier en premier. L'initiateur peut cependant apprendre que le répondant prend en charge IKE ainsi que les protocoles cryptographiques qu'il accepte. Le répondant (ou quelqu'un qui se fait passer pour le répondant) peut sonder l'initiateur non seulement sur son identité, mais en utilisant des charges utiles CERTREQ, peut être capable de déterminer quels certificats l'initiateur veut utiliser.

L'utilisation de l'authentification EAP change quelque peu les possibilités d'investigation. Lorsque l'authentification EAP est utilisée, le répondant prouve son identité avant l'initiateur, ainsi un initiateur qui connaît le nom d'un initiateur valide pourrait sonder le répondant à la fois sur son nom et ses certificats.

Un renouvellement de clés répété en utilisant CREATE_CHILD_SA sans échange Diffie-Hellman supplémentaire laisse toutes les SA vulnérables à l'analyse cryptographique d'une seule clé ou à la subversion d'un des points d'extrémité. Les développeurs devraient noter le fait et mettre une limite aux échanges CREATE_CHILD_SA entre exponentiations. Le présent mémoire ne prescrit pas une telle limite.

La force d'une clé déduite d'un échange Diffie-Hellman utilisant un des groupes définis ici dépend de la force inhérente du groupe, de la taille des exposants utilisés, et de l'entropie fournie par le générateur de nombres aléatoires utilisé. Du fait de ces éléments, il est difficile de déterminer la force d'une clé pour un groupe défini. Le groupe Diffie-Hellman numéro deux, utilisé avec un fort générateur de nombres aléatoires et un exposant qui n'est pas inférieur à 200 bits, est d'usage courant avec 3DES. Le groupe cinq donne une plus grande sécurité que le groupe deux. Le groupe un n'a plus d'intérêt qu'historique et ne donne pas une force suffisante en dehors de son utilisation avec DES, qui est lui aussi complètement dépassé. Les mises en œuvre devraient noter ces évaluations pour établir des politiques et négocier des paramètres de sécurité.

Noter que ces limitations sont sur les groupes Diffie-Hellman eux-mêmes. Il n'y a rien dans IKE qui interdise d'utiliser des groupes plus forts, ni rien qui dilue la force obtenue de groupes plus forts (limitée par la force des autres algorithmes négociés, y compris la fonction pseudo aléatoire). En fait, le cadre extensible d'IKE encourage à la définition de plus de groupes ; l'utilisation de groupes à courbe elliptique peut accroître grandement la force tout en utilisant des nombres beaucoup plus petits.

On suppose que tous les exposants Diffie-Hellman sont effacés des mémoires après usage. En particulier, ces exposants NE DOIVENT PAS être déduits de secrets à longue durée de vie comme la racine d'un générateur pseudo-aléatoire qui n'est pas effacée après usage.

La force de toutes les clés est limitée par la taille du résultat de la fonction prf négociée. Pour cette raison, une fonction prf dont le résultat est inférieur à 128 bits (par exemple, 3DES-CBC) NE DOIT PAS être utilisée avec ce protocole.

La sécurité de ce protocole est très dépendante du caractère aléatoire des paramètres choisis de façon aléatoire. Ceux-ci devraient être générés par une source fortement aléatoire sous une source pseudo-aléatoire avec une graine appropriée (voir la RFC 4086). Les développeurs devraient veiller à s'assurer que l'utilisation de nombres aléatoires à la fois pour les clés et pour les noms occasionnels est agencée d'une façon qui ne mette pas en péril la sécurité des clés.

Pour des informations sur les motivations de beaucoup des choix de conceptions cryptographiques du présent protocole, voir [SIGMA] et [SKEME]. Bien que la sécurité des CHILD_SA négociées ne dépende pas de la force du chiffrement et de la protection d'intégrité négociées dans la IKE_SA, les mises en œuvre NE DOIVENT PAS négocier NONE comme algorithme IKE de protection d'intégrité ou ENCR_NULL comme algorithme IKE de chiffrement.

Lors de l'utilisation de clés pré-partagées, une considération critique est d'assurer le caractère aléatoire de ces secrets. La pratique la plus forte est de s'assurer qu'aucune clé pré-partagée ne contient autant d'aléa que la plus forte clé en négociation. Déduire un secret partagé d'un mot de passe, d'un nom, ou autre source à faible entropie n'est pas sûr. Ces sources sont susceptibles d'attaques de dictionnaire et d'ingénierie sociale, entre autres.

Les notifications NAT_DETECTION_*_IP contiennent un hachage des adresses et accès pour essayer de dissimuler les adresses IP internes qui sont derrière un NAT. Comme l'espace d'adresse IPv4 est seulement de 32 bits, et qu'il est habituellement très clairsemé, il serait possible à un attaquant de découvrir l'adresse interne utilisée derrière le NAT en essayant toutes les adresses IP possibles et en essayant de trouver le hachage correspondant. Les numéros d'accès sont normalement fixés à 500, et les SPI peuvent être extraits du paquet. Cela réduit le nombre de calculs de hachages à 2^32. Avec de bonnes hypothèses sur l'utilisation de l'espace d'adresses privées, le nombre de calculs de hachage est bien plus faible. Les concepteurs ne devraient donc pas supposer que l'utilisation de IKE ne va pas donner lieu à des fuites d'informations des adresses internes.

Lorsqu'on utilise une méthode d'authentification EAP qui ne génère pas une clé partagée pour protéger une charge utile AUTH ultérieure, certaines attaques par interposition et par travestissement du serveur sont possibles [EAPMITM]. Ces faiblesses surviennent lorsque EAP est aussi utilisé dans des protocoles qui ne sont pas protégés avec un tunnel sûr.

Comme EAP est un protocole d'authentification tout venant, qui est souvent utilisé pour fournir des facilités à une seule signature, le développement d'une solution IPsec s'appuyant sur une méthode d'authentification EAP qui ne génère pas une clé partagée (aussi appelée une méthode EAP à non génération de clé) peut être compromise du fait du développement d'une application entièrement non corrélée qui va aussi arriver à utiliser la même méthode EAP à non génération de clé, mais de façon non protégée. Noter que cette faiblesse n'est pas limitée seulement à EAP, mais peut survenir dans d'autres scénarios où une infrastructure d'authentification est réutilisée. Par exemple, si le mécanisme EAP utilisé par IKEv2 utilise un authentifiant à jetons, une attaque par interposition pourrait se faire passer pour le serveur de la Toile, intercepter l'échange de jetons d'authentification, et l'utiliser pour initier une connexion IKEv2. Pour cette raison, l'utilisation de méthodes EAP à non génération de clé DEVRAIT être évitée autant que possible. Lorsqu'elles sont utilisées, il est extrêmement important que toute usage de ces méthodes EAP DEVRAIT utiliser un tunnel protégé, où l'initiateur valide le certificat du répondant avant d'initier l'échange EAP. Les développeurs DEVRAIENT décrire les faiblesses de l'utilisation des méthodes EAP à non génération de clé dans la documentation de leurs mises en œuvre de façon que les administrateurs qui développent des solutions IPsec soient conscients de ses dangers.

Une mise en œuvre utilisant EAP DOIT aussi utiliser une authentification fondée sur la clé publique du serveur auprès du client avant le début de l'échange EAP, même si la méthode EAP offre l'authentification mutuelle. Cela évite d'avoir des variations de protocole IKEv2 supplémentaires et protège les données EAP contre les attaques actives.

Si les messages de IKEv2 sont assez longs pour que la fragmentation de niveau IP soit nécessaire, il est possible que des attaques puissent empêcher l'achèvement de l'échange en saturant les mémoires tampon de réassemblage. Les chances de réussite d'une telle attaque peuvent être minimisées en utilisant les codages de hachage et d'URL au lieu d'envoyer les certificats (voir au paragraphe 3.6). Des mesures supplémentaires sont exposées dans [KPS03].

6 Considérations relatives à l'IANA

Le présent document définit un certain nombre de nouveaux types et valeurs de champs pour lesquels des allocations futures seront gérées par l'IANA.

Les registres suivants ont été créés par l'IANA:

IKEv2 Exchange Types (paragraphe 3.1)
IKEv2 Payload Types (paragraphe 3.2)
IKEv2 Transform Types (paragraphe 3.3.2)
IKEv2 Transform Attribute Types (paragraphe 3.3.2)
IKEv2 Encryption Transform IDs (paragraphe 3.3.2)
IKEv2 Pseudo-random Function Transform IDs (paragraphe 3.3.2)
IKEv2 Integrity Algorithm Transform IDs (paragraphe 3.3.2)
IKEv2 Diffie-Hellman Transform IDs (paragraphe 3.3.2)
IKEv2 Identification Payload ID Types (paragraphe 3.5)
IKEv2 Certificate Encodings (paragraphe 3.6)
IKEv2 Authentication Method (paragraphen 3.8)
IKEv2 Notify Message Types (paragraphen 3.10.1)
IKEv2 Notification IPCOMP Transform IDs (paragraphe 3.10.1)
IKEv2 Security Protocol Identifiers (paragraphe 3.3.1)
IKEv2 Traffic Selector Types (paragraphe 3.13.1)
IKEv2 Configuration Payload CFG Types (paragraphe 3.15)
IKEv2 Configuration Payload Attribute Types (paragraphe 3.15.1)

Note : Un nouveau registre doit être créé lors de la création d'un nouveau type de transformation.

Les changements et ajouts à un de ces registres sont soumis à révision par experts.

7 Remerciements

Le présent document est le résultat de la collaboration de l'ensemble du groupe de travail IPsec. Si le nombre des auteurs qui peuvent figurer sur une RFC n'était pas limité, voici ceux qui y figureraient, par ordre alphabétique : Bill Aiello, Stephane Beaulieu, Steve Bellovin, Sara Bitan, Matt Blaze, Ran Canetti, Darren Dukes, Dan Harkins, Paul Hoffman, John Ioannidis, Charlie Kaufman, Steve Kent, Angelos Keromytis, Tero Kivinen, Hugo Krawczyk, Andrew Krywaniuk, Radia Perlman, Omer Reingold, et Michael Richardson. De nombreuses autres personnes ont contribué à sa conception. Il s'agit d'une évolution de IKEv1, d'ISAKMP, et du DOI IPsec, dont chacun a sa propre liste d'auteurs. Hugh Daniel a suggéré le

dispositif par lequel l'initiateur, dans le message 3, spécifie un nom pour le répondant, et a donné à ce dispositif le joli nom de "Toi Tarzan, moi Jane". David Faucher et Valery Smyzlov ont aidé à préciser le concept de négociation de sélecteur de trafic.

8 Références

8.1 Références normatives

[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 RFC 5226)*

[RFC2451] R. Pereira, R. Adams, "Algorithmes de chiffrement ESP en mode CBC", novembre 1998. *(P.S.)*

[RFC3168] K. Ramakrishnan et autres, "Ajout de la notification explicite d'encombrement (ECN) à IP", septembre 2001. *(P.S.)*

[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)*

[RFC3513] R. Hinden et S. Deering, "Architecture d'adressage du protocole Internet version 6 (IPv6)", avril 2003. (*Obs. voir* RFC4291)

[RFC3526] T. Kivinen et M. Kojo, "Groupes supplémentaires d'exponentiation modulaire (MODP) Diffie-Hellman pour l'échange de clés Internet (IKE)", mai 2003.

[RFC3748] B. Aboba et autres, "Protocole extensible d'authentification ", juin 2004. *(P.S., MàJ par* RFC5247*)*

[RFC3948] A. Huttunen et autres, "Encapsulation UDP de paquets ESP d'IPsec", janvier 2005. *(P.S.)*

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

8.2 Références informatives

[DES] ANSI X3.106, "American National Standard for Information Systems-Data Link Encryption", American National Standards Institute, 1983.

[DH] W. Diffie et M. Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory, V. IT-22, n. 6 juin 1977.

[DSS] NIST, "Digital Signature Standard", FIPS 186, National Institute of Standards et Technology, U.S. Department of Commerce, mai 1994.

[EAPMITM] N. Asokan, V. Nierni et K. Nyberg, "Man-in-the-Middle in Tunneled Authentication Protocols", http://eprint.iacr.org/2002/163, novembre 2002.

[IDEA] X. Lai, "On the Design et Security of Block Ciphers," ETH Series dans Information Processing, v. 1, Konstanz: Hartung-Gorre Verlag, 1992.

[KPS03] C. Kaufman, R. Perlman et B. Sommerfeld, "DoS protection for UDP-based protocols", Conférence ACM sur la sécurité informatique et des communications, octobre 2003.

[PK01] R. Perlman et C. Kaufman, "Analyse de la norme IPsec d'échange de clés", WET-ICE Security Conference, MIT,2001, http://sec.femto.org/wetice-2001/papers/radia-paper.pdf.

[RFC1321] R. Rivest, "Algorithme de résumé de message MD5", avril 1992. *(Information)*

[RFC1958] B. Carpenter, éd., "Principes de l'architecture de l'Internet", juin 1996. (*MàJ par* RFC3439) *(Information)*

[RFC2104] H. Krawczyk, M. Bellare et R. Canetti, "HMAC : Hachage de clés pour l'authentification de message", février 1997.

[RFC2131] R. Droms, "Protocole de configuration dynamique d'hôte", mars 1997. *(Mise à jour par les RFC 3396 et 4361)*

[RFC2251] M. Wahl, T. Howes et S. Kille, "Protocole léger d'accès à un répertoire (v3)", décembre 1997.

[RFC2367] D. McDonald, C. Metz, B. Phan, "API de gestion de clé PF_KEY, version 2", juillet 1998. *(Information)*

[RFC2401] S. Kent et R. Atkinson, "Architecture de sécurité pour le protocole Internet", novembre 1998. (*Obsolète, voir*

RFC4301)

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

[RFC2408] D. Maughan, M. Schertler, M. Schneider et J. Turner, "Association de sécurité Internet et protocole de 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)*

[RFC2412] H. Orman, "Protocole OAKLEY de détermination de clés", novembre 1998. *(Information)*

[RFC2474] K. Nichols, S. Blake, F. Baker et D. Black, "Définition du champ Services différenciés (DS) dans les en-têtes IPv4 et IPv6", décembre 1998. (*MàJ par* RFC3168, RFC3260) *(P.S.)*

[RFC2475] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang et W. Weiss, "Architecture pour services différenciés", décembre 1998. *(MàJ par RFC3260)*

[RFC2522] P. Karn, W. Simpson, "Photuris : Protocole de gestion de clé de session", mars 1999. *(Expérimentale)*

[RFC2775] B. Carpenter, "Transparence de l'Internet", février 2000. *(Information)*

[RFC2865] C. Rigney et autres, "Service d'authentification à distance de l'utilisateur appelant (RADIUS)", juin 2000. (*MàJ par* RFC2868, RFC3575, RFC5080) *(D.S.)*

[RFC2983] D. Black, "Services différenciés et tunnels", octobre 2000. *(Information)*

[RFC3173] A. Shacham et autres, "Protocole de compression de charge utile IP (IPComp)", septembre 2001. *(P.S.)*

[RFC3439] R. Bush, D. Meyer, "Lignes directrices et philosophie de l'architecture de l'Internet", décembre 2002. *(Information)*

[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.

[RFC3715] B. Aboba, W. Dixon, "Exigences de compatibilité entre IPsec et la traduction d'adresse réseau (NAT)", mars 2004. *(Info.)*

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

[RFC4302] S. Kent, "En-tête d'authentification IP", décembre 2005. *(P.S.)*

[RFC4303] S. Kent, "Encapsulation de charge utile de sécurité dans IP (ESP)", décembre 2005. (Remplace RFC2406) *(P.S.)*

[RSA] R. Rivest, A. Shamir et L. Adleman, "Méthode pour obtenir des systèmes de chiffrement de signatures numériques et de clés publiques", Communications de ACM, v. 21, n. 2, février 1978.

[SHA] NIST, "Norme de hachage sécurisé", FIPS 180-1, National Institute of Standards and Technology, U.S. Department of Commerce, mai 1994.

[SIGMA] H. Krawczyk, "SIGMA: the `SIGn-et-MAc' Approach to Authenticated Diffie-Hellman and its Use in the IKE Protocols", dans Advances in Cryptography - CRYPTO 2003 Proceedings, LNCS 2729, Springer, 2003. Disponible à : http://www.informatik.uni-trier.de/~ley/db/conf/crypto/crypto2003.html.

[SKEME] H. Krawczyk, "SKEME: A Versatile Secure Key Exchange Mechanism for Internet", Compte rendu du Symposium 1996 de l'IEEE sur la sécurité des systèmes en réseau et distribués.

[X.501] Recommandation UIT-T X.501 : Technologies de l'information - Interconnexion des systèmes ouverts – L'annuaire : Modèles, 1993.

[X.509] Recommandation UIT-T X.509 (1997 E) : Technologies de l'information - Interconnexion des systèmes ouverts – L'annuaire : Cadre d'authentification, juin 1997.

Appendice A : Résumé des changements depuis IKEv1

Les objectifs de cette révision de IKE sont :

1) de définir le protocole IKE tout entier dans un seul document, remplaçant les RFC 2407, 2408, et 2409 et incorporant les changements destinés à prendre en charge la traversée de NAT, l'authentification extensible, et l'acquisition d'adresse distante ;
2) de simplifier IKE en remplaçant les huit échanges initiaux différents par un seul échange de quatre messages (avec des

changements dans les mécanismes d'authentification qui affectent une seule charge utile AUTH plutôt que de restructurer tout l'échange) voir [PK01] ;
3) de supprimer les champs Domaine d'interprétation (DOI), Situation (SIT), et Identifiant de domaine étiqueté, et les bits Commit et Authentication only ;
4) de diminuer la latence de IKE dans le cas courant en ramenant l'échange initial à deux allers-retours (4 messages), et en permettant la capacité de portage de l'établissement de CHILD_SA sur cet échange ;
5) de remplacer la syntaxe cryptographique pour la protection des messages IKE eux-mêmes par une syntaxe fondée étroitement sur ESP pour simplifier la mise en œuvre et l'analyse de la sécurité ;
6) de réduire le nombre d'états d'erreur possibles en rendant le protocole fiable (en accusant réception de tous les messages et en les séquençant). Cela permet de raccourcir les échanges CREATE_CHILD_SA de 3 messages à 2 ;
7) d'accroître la robustesse en permettant au répondant de ne pas faire de traitement significatif jusqu'à ce qu'il reçoive un message prouvant que l'initiateur peut recevoir des messages à ce qu'il prétend être son adresse IP, et de ne pas affecter d'état à un échange jusqu'à ce que l'initiateur puisse être cryptographiquement authentifié ;
8) de régler les faiblesses cryptographiques, comme le problèmes des symétries dans les hachages utilisés pour l'authentification, documentées par Tero Kivinen ;
9) de spécifier les sélecteurs de trafic dans leur propre type de charge utile plutôt que de surcharge les charges utiles d'identifiant, et de rendre plus souple les sélecteurs de trafic qui peuvent être spécifiés ;
10) de spécifier le comportement requis sous certaines conditions d'erreur ou lors de la réception de données non comprises, pour faciliter des révisions futures sans entamer la rétro compatibilité ;
11) de simplifier et préciser la maintenance d'un état partagé en présence de défaillances de réseau et d'attaques de déni de service ; et
12) de maintenir dans la mesure du possible la syntaxe et les nombres magiques existants pour rendre probable la mise à niveau des mises en œuvre de IKEv1 afin qu'elles puissent prendre en charge IKEv2 avec le minimum d'efforts.

Appendice B : Groupes Diffie-Hellman

Deux groupes Diffie-Hellman sont définis ici pour usage dans IKE. Ces groupes ont été générés par Richard Schrœppel à l'Université de l'Arizona. Les propriétés de ces nombres premiers sont décrites dans la [RFC2412].

La force apportée par le groupe 1 peut n'être pas suffisante pour l'algorithme de chiffrement de mise en œuvre obligatoire et ne figure ici que pour des raisons historiques.

Des groupes Diffie-Hellman supplémentaires ont été définis dans [la RFC3526].

B.1 Groupe 1 – MODP à 768 bits

Ce groupe a reçu l'id 1 (un).

Le principal est : 2^768 - 2 ^704 - 1 + 2^64 * { [2^638 pi] + 149686 } Sa valeur hexadécimale valeur est :

FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1 29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245 E485B576 625E7EC6 F44C42E9 A63A3620 FFFFFFFF FFFFFFFF

Le générateur est 2.

B.2 Groupe 2 – MODP à 1024 bits

Ce groupe à reçu l'id 2 (deux).

Le principal est 2^1024 - 2^960 - 1 + 2^64 * { [2^894 pi] + 129093 }. Sa valeur hexadécimale est :

FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1 29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245 E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE65381 FFFFFFFF FFFFFFFF

Le générateur est 2.

Adresse de l'éditeur

Charlie Kaufman
Microsoft Corporation
1 Microsoft Way
Redmond, WA 98052
Phone: 1-425-707-3335
Email: charliek@microsoft.com

Déclaration de droits de reproduction

Propriété intellectuelle

Remerciement

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