Équipe d'ingénierie de l'Internet (IETF)
Request for Comments : 7296
STD 79
RFC rendue obsolète : 5996
Catégorie : Norme
ISSN : 2070-1721
Traduction Claude Brière de L'Isle

C. Kaufman, Microsoft
P. Hoffman, VPN ConsortiumC
Y. Nir, Check Point
P. Eronen, Independent
T. Kivinen, INSIDE Secure
octobre 2014

Protocole d'échange de clé Internet version 2 (IKEv2)

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). Le présent document rend obsolète la RFC 5996, et inclut tous ses errata. Il fait avancer IKEv2 au statut de norme de l'Internet.

Statut de ce mémoire
Le présent mémoire est une norme de l'Internet.

Le présent document a été produit par l'équipe d'ingénierie de l'Internet (IETF). Il représente le consensus de la communauté de l'IETF. Il a subi une révision publique et sa publication a été approuvée par le groupe de pilotage de l'ingénierie de l'Internet (IESG). Plus d'informations sur les normes de l'Internet sont disponibles à la Section 2 de la [RFC5741].

Les informations sur le statut actuel du présent document, tout errata, et comment fournir des réactions sur lui peuvent être obtenues à http://www.rfc-editor.org/info/rfc7296

Table des matières

1. Introduction....2
1.1 Scénarios d'utilisation....3
1.2 Les échanges initiaux....5
1.3 L'échange CREATE_CHILD_SA....7
1.4 L'échange INFORMATIONAL....9
1.5 Messages d'information en-dehors d'une IKE_SA....10
1.6. Terminologie des exigences....11
1.7 Différences significatives avec la RFC 4306 et la RFC 5996....11
1.8 Différences entre la RFC 5996 et le présent document....12
2 Détails du protocole IKE et variantes....13
2.1 Utilisation de temporisateurs de retransmission....13
2.2 Utilisation de numéros de séquence pour les identifiants de message....14
2.3 Taille de fenêtre pour demandes en chevauchement....15
2.4 Synchronisation d'état et fins de temporisation de connexion....15
2.5 Numéros de version et rétro compatibilité....16
2.6 SPI et mouchards de SA IKE....17

2.7 Négociation d'algorithme cryptographique....19
2.8 Changement de clés....20
2.9 Négociation de sélecteurs de trafic....23
2.10 Noms occasionnels....25
2.11 Agilité d'adresse et d'accès....25
2.12 Réutilisation des exponentielles Diffie-Hellman....25
2.13 Génération du matériel de chiffrement....26
2.14 Génération du matériel de chiffrement pour la SA IKE....26
2.15 Authentification de la SA IKE....27
2.16 Méthodes d'extension du protocole d'authentification....28
2.17 Génération des matériaux de chiffrement pour les SA filles....29
2.18 Changement de clé des SA IKE avec l'échange CREATE_CHILD_SA....30
2.19 Demande d'une adresse interne sur un réseau distant....30
2.20 Demande de la version de l'homologue....31
2.21 Traitement d'erreur....31
2.22 IPComp....33
2.23 Traversée de NAT....34
2.24 Notification explicite d'encombrement (ECN)....37
2.25. Collisions d'échange....38
3. Formats d'en-tête et de charge utile....39
3.1 En-tête IKE....39
3.2 En-tête générique de charge utile....40
3.3 Charge utile Association de sécurité....41
3.4 Charge utile Échange de clé....49
3.5 Charges utiles Identification....49
3.6 Charge utile Certificat....50
3.7 Charge utile Demande de certificat....52
3.8 Charge utile Authentification....53
3.9 Charge utile Nom occasionnel....54
3.10 Charge utile Notifie....54
3.11 Charge utile Supprime....56
3.12 Charge utile Identifiant de fabricant....57
3.13 Charge utile Sélecteur de trafic....58
3.14 Charge utile Chiffré....60
3.15 Charge utile Configuration....61
3.16 Charge utile du protocole d'authentification extensible (EAP)....66
4. Exigences de conformité....66
5. Considérations sur la sécurité....67
5.1 Autorisation de sélecteur de trafic....69
6. Considérations relatives à l'IANA....70
7. Références....70
7.1 Références normatives....70
7.2 Références pour information....71
Appendice A. Résumé des changements par rapport à IKEv1....73
Appendice B. Groupes Diffie-Hellman....73
B.1 Groupe 1 - MODP à 768 bits....74
B.2. Groupe 2 - MODP à 1024 bits....74
Appendice C. Échanges et charges utiles....74
C.1 Échange IKE_SA_INIT....74
C.2 Échange IKE_AUTH sans EAP....75
C.3 Échange IKE_AUTH avec EAP....75
C.4 Échange CREATE_CHILD_SA pour créer ou changer les clés des SA filles....76
C.5 Échange CREATE_CHILD_SA pour le changement de clé de SA IKE....76
C.6 Échange INFORMATIONAL....76
Remerciements....76
Adresse des auteurs....77

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*). La version 1 de IKE était définie dans les [RFC2407], [RFC2408], et [RFC2409]. IKEv2 a remplacé toutes ces RFC. IKEv2 était défini dans la [RFC4306] et a été précisé par la [RFC4718]. La [RFC5996] a remplacé et mis à jour les RFC 4306 et 4718. Le présent document remplace la RFC 5996. IKEv2 comme spécifié dans la RFC 4306 était un changement au protocole IKE qui n'était pas rétrocompatible. La RFC 5996 révisait la RFC 4306 pour fournir des précisions sur IKEv2, en faisant un minimum de changements au protocole IKEv2. Le présent document remplace la RFC 5996, en la révisant légèrement pour en faire une norme de l'Internet convenable. Une liste des différences significatives entre les RFC 4306 et 5996 figure au paragraphe 1.7, et les différences entre la RFC 5996 et le présent document figurent au paragraphe 1.8.

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", et est parfois appelée une "paire demande/réponse". Les deux premiers échanges de messages qui établissent une SA IKE sont appelés l'échange IKE_SA_INIT *(initier une SA IKE)* et l'échange IKE_AUTH *(authentification IKE)* ; les échanges IKE suivants sont appelés soit des échanges CREATE_CHILD_SA *(créer une SA fille)* soit des échanges 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 SA fille. Dans des cas exceptionnels, il peut y avoir plus que 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 SA fille est nécessaire entre les points d'extrémité IPsec, et donc, il n'y aura pas d'échange supplémentaire. Des échanges ultérieurs PEUVENT être utilisés pour établir des SA filles supplémentaires entre la même paire de points d'extrémité authentifiés et pour effectuer des fonctions 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 délai de temporisation, le demandeur doit retransmettre la demande (ou abandonner la connexion).

Le premier échange d'une session IKE, IKE_SA_INIT, négocie les paramètres de sécurité pour la SA IKE, envoie les noms occasionnels *(nom occasionnel)*, et envoie les valeurs Diffie-Hellman.
Le second échange, IKE_AUTH, transmet les identités, prouve sa connaissance des secrets correspondants aux deux identités, et établit une SA pour la première (et souvent seule) SA fille AH et/ou ESP (sauf si il y a échec d'établissement de la SA AH ou ESP fille, auquel cas la SA IKE est quand même établie sans la SA fille).

Les types d'échanges ultérieurs sont CREATE_CHILD_SA (qui crée une SA fille) 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 Encrypted *(chiffrée)* vide 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 utilisé 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

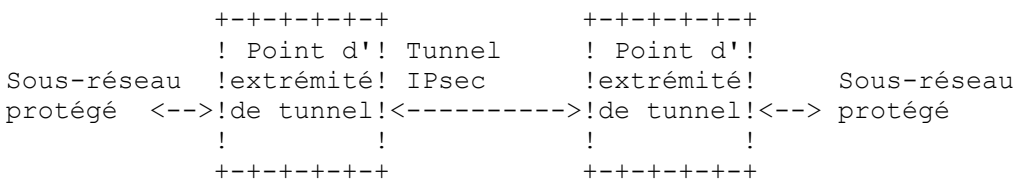


Figure 1 : Tunnel de passerelle de sécurité à passerelle de sécurité

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. 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'Internet IPv4, 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

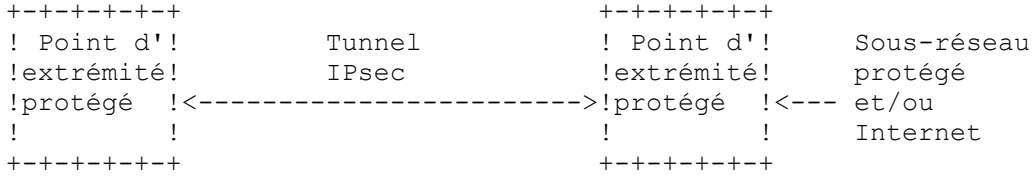


Figure 3 : Tunnel de point d'extrémité à passerelle de sécurité

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 peut n'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 (à savoir les charges utiles de configuration) 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. L'interaction avec les NAT est couverte en détail au paragraphe 2.23.

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 fournisseur d'accès Internet (FAI) 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 SA fille. 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. Voir au paragraphe 2.14 des informations sur la façon dont les clés de chiffrement sont générées. (Un attaquant par interposition qui ne peut pas réaliser l'échange IKE_AUTH complet peut néanmoins voir l'identité de l'initiateur.)

Tous les messages qui suivent l'échange initial sont protégés cryptographiquement en utilisant les algorithmes et clés de chiffrement négociés dans l'échange IKE_SA_INIT. Les messages suivants utilisent la syntaxe de la charge utile chiffrée décrite au paragraphe 3.14, chiffrés avec des clés qui sont déduites comme décrit au paragraphe 2.14. Tous les messages suivants incluent une charge utile chiffrée, même si il y est fait référence dans le texte comme "vides". Pour les échanges CREATE_CHILD_SA, IKE_AUTH, ou INFORMATIONAL, 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 SA IKE.

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

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
EAP	*(Extensible Authentication)* authentification extensible
HDR	*(IKE Header)* en-tête IKE
IDi	*(Identification – Initiator)* identification de l'initiateur
IDr	*(Identification – Responder)* identification du répondant
KE	*(Key Exchange)* échange de clés
Ni, Nr	*(Nonce)* nom occasionnel
N	*(Notify)* notifie

SA	*(Security Association)* association de sécurité
SK	*(Encrypted et Authenticated)* chiffrée et authentifiée
TSi	*(Traffic Selector – Initiator)* sélecteur de trafic de l'initiateur
TSr	*(Traffic Selector – Responder)* 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, le type d'échange, l'identifiant de message, 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 une quantité appelée SKEYSEED (voir au paragraphe 2.14), à partir de laquelle toutes les clés sont déduites pour cette SA IKE. Tous les messages qui suivent sont chiffrés et protégés en intégrité, à l'exception des en-têtes de message. 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) ; voir les paragraphes 2.13 et 2.14 pour les détails de la déduction de clé. 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 Diffie-Hellman pour la protection de la IKE_SA, une autre quantité SK_d est déduite et utilisée pour la dérivation des matériaux de chiffrement ultérieurs pour les SA filles. 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 ou des charges utiles CERT sont incluses, 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. Si l'IDr proposé par l'initiateur n'est pas acceptable au répondant, celui-ci peut utiliser d'autres IDr pour finir l'échange. Si l'initiateur n'accepte alors pas le fait que le répondant ait utilisé un IDr différent de celui qui était demandé, l'initiateur peut clore la SA après qu'il a remarqué le fait.

Les sélecteurs de trafic (TSi et TSr) sont discutés au paragraphe 2.9.

L'initiateur commence la négociation d'une SA fille en utilisant la charge utile SAi2. Les champs finaux (commençant par SAi2) sont décrits avec l'échange CREATE_CHILD_SA *(créer une SA fille)*.

← 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 de liste) authentifie son identité et protège l'intégrité du second message avec la charge utile AUTH, et achève la négociation d'une SA fille avec les champs supplémentaires décrits ci-dessous dans l'échange CREATE_CHILD_SA. Les deux parties à l'échange IKE_AUTH DOIVENT vérifier que toutes les signatures et les codes d'authentification de message (MAC, *Message Authentication Code*) sont calculés correctement. Si l'un des côtés utilise un secret partagé pour l'authentification, les noms dans la charge utile ID DOIVENT correspondre à la clé utilisée pour générer la charge utile AUTH.

Parce que l'initiateur envoie sa valeur Diffie-Hellman dans le IKE_SA_INIT *(initialisation d'association de sécurité IKE)* il doit deviner le groupe Diffie-Hellman que le répondant va choisir à partir de sa liste des groupes pris en charge. Si l'initiateur ne devine pas bien, le répondant va répondre par une charge utile Notify de type INVALID_KE_PAYLOAD *(charge utile d'échange de clé invalide)* 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 son ensemble complet de suites cryptographiques acceptables parce que le message de rejet n'était pas authentifié et autrement un attaquant actif pourrait tromper les points d'extrémité en négociant une suite plus faible qu'une plus forte que tous deux pourraient préférer.

Si la création de la SA fille durant l'échange IKE_AUTH échoue pour une raison quelconque, la SA IKE est quand même créée comme d'habitude. La liste des types de messages Notify dans l'échange IKE_AUTH qui n'empêchent pas une SA IKE d'être établie inclut au moins : NO_PROPOSAL_CHOSEN *(pas de proposition choisie)*, TS_UNACCEPTABLE *(sélecteur de trafic inacceptable)*, SINGLE_PAIR_REQUIRED *(une seule paire exigée)*, INTERNAL_ADDRESS_FAILURE *(échec d'adresse interne)*, et FAILED_CP_REQUIRED *(échec de charge utile de configuration exigé)*.

Si l'échec se rapporte à la création de la SA IKE (par exemple, un message d'erreur Notify AUTHENTICATION_FAILED *(échec d'authentification)* est retourné) la SA IKE n'est pas créée. Noter que bien que les messages IKE_AUTH soient chiffrés et protégés en intégrité, si l'homologue qui reçoit ce message d'erreur Notify n'a pas encore authentifié l'autre extrémité (ou si l'homologue échoue à authentifier l'autre extrémité pour une raison quelconque) les informations doivent être traitées avec précaution. Plus précisément, en supposant que le MAC se vérifie correctement, l'envoyeur du message d'erreur Notify est connu comme étant le répondant dans l'échange IKE_SA_INIT, mais l'identité de l'envoyeur ne peut pas être assurée.

Noter que les messages IKE_AUTH ne contiennent pas de charge utile KEi/KEr ou Ni/Nr. Donc, les charges utiles de SA dans l'échange IKE_AUTH ne peuvent pas contenir de transformation de type 4 (groupe Diffie-Hellman) avec une valeur autre que AUCUNE. Les mises en œuvre DEVRAIENT omettre toute la sous structure de transformation plutôt que d'envoyer la valeur AUCUNE.

1.3 L'échange CREATE_CHILD_SA

L'échange CREATE_CHILD_SA est utilisé pour créer de nouvelles SA filles et pour changer les clés des SA IKE et des SA filles. Cet échange consiste en une seule paire demande/réponse, et certaines de ses fonction étaient appelées un 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.

Les clés d'une SA sont changées en créant une nouvelle SA et en supprimant ensuite la vieille. Ce paragraphe décrit la première partie du changement de clés, la création des nouvelles SA ; le paragraphe 2.8 couvre les mécanismes du changement de clé, incluant de déplacer le trafic des anciennes SA aux nouvelles et la suppression des vieilles SA. Les deux paragraphes doivent être lus ensemble pour comprendre le processus de changement de clé entier.

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 mise en œuvre PEUT refuser toutes les demandes CREATE_CHILD_SA au sein d'une SA IKE.

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 SA fille. Le matériel de clés pour la SA fille 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).

Si un échange CREATE_CHILD_SA inclut une charge utile KEi, au moins une des offres de SA DOIT inclure le groupe Diffie-Hellman de la KEi. Le groupe Diffie-Hellman de la KEi DOIT être un élément du groupe que l'initiateur s'attend qu'accepte le répondant (des groupes Diffie-Hellman supplémentaires peuvent être proposés). Si le répondant choisit une proposition qui utilise un groupe Diffie-Hellman différent (autre que AUCUN) le répondant DOIT rejeter la demande et indiquer son groupe Diffie-Hellman préféré dans la charge utile Notify INVALID_KE_PAYLOAD. Il y a deux octets de données associés à cette notification : le numéro de groupe Diffie-Hellman accepté en ordre gros boutien. En cas d'un tel rejet, l'échange CREATE_CHILD_SA échoue, et l'initiateur va probablement réessayer l'échange avec une proposition Diffie-Hellman et une KEi dans le groupe que le répondant a donné dans la charge utile Notify INVALID_KE_PAYLOAD.

Le répondant envoie une notification NO_ADDITIONAL_SAS pour indiquer qu'une demande CREATE_CHILD_SA n'est pas acceptable parce que le répondant ne veut pas accepter d'autres SA filles sur cette SA IKE. Cette notification peut aussi être utilisée pour rejeter un changement de clé de SA IKE. Certaines mises en œuvre minimales peuvent n'accepter qu'un seul établissement de SA fille dans le contexte d'un échange IKE initial et rejeter toute tentative ultérieure d'en ajouter.

1.3.1 Création de nouvelles SA filles avec l'échange CREATE_CHILD_SA

Une SA fille peut être créée par l'envoi d'une demande CREATE_CHILD_SA. La demande CREATE_CHILD_SA pour changer les clés d'une SA fille est :

Initiateur **Répondant**
HDR, SK {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 pour la SA fille proposée dans les charges utiles TSi et TSr.

La réponse CREATE_CHILD_SA pour créer une nouvelle SA fille est :

← 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, un nom occasionnel dans la charge utile Nr, 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.

Les sélecteurs de trafic pour le trafic à envoyer sur cette SA sont spécifiés dans les charges utiles TS de la réponse, qui peuvent être un sous ensemble de ce que proposait l'initiateur de la SA fille.

La notification USE_TRANSPORT_MODE PEUT être incluse dans un message de demande qui inclut aussi une charge utile de SA demandant une SA fille. Elle demande que la SA fille 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 de type USE_TRANSPORT_MODE. Si le répondant décline la demande, la SA fille sera établie en mode tunnel. Si cela est inacceptable pour l'initiateur, celui-ci DOIT supprimer la SA. Noter qu'excepté quand on utilise cette option pour négocier le mode transport, toutes les SA filles vont utiliser le mode tunnel.

La notification ESP_TFC_PADDING_NOT_SUPPORTED affirme que le point d'extrémité envoyeur ne va pas accepter des paquets qui contiennent un bourrage de confidentialité de flux de trafic (TFC, *Traffic Flow Confidentiality*) sur la SA fille en négociation. Si aucun des deux points d'extrémité n'accepte le bourrage de TFC, cette notification est incluse dans la demande et dans la réponse. Si cette notification est incluse dans un seul des messages, le bourrage de TFC peut encore être envoyé dans l'autre direction.

La notification NON_FIRST_FRAGMENTS_ALSO est utilisée pour le contrôle de la fragmentation. Voir des explications plus complètes dans la [RFC4301]. Les deux parties doivent se mettre d'accord pour envoyer les fragments non premiers avant que l'une ou l'autre des parties le fasse. Elle n'est activée que si la notification NON_FIRST_FRAGMENTS_ALSO est incluse dans la demande proposant une SA et dans la réponse qui l'accepte. Si le répondant ne veut pas envoyer ou recevoir de fragments non premiers, il omet simplement la notification NON_FIRST_FRAGMENTS_ALSO de sa réponse, mais ne rejette pas toute la création de la SA fille.

Une notification IPCOMP_SUPPORTED, traitée au paragraphe 2.22, peut aussi être incluse dans l'échange.

Un échec de la tentative de créer une SA fille NE DEVRAIT PAS supprimer la SA IKE : il n'y a pas de raison de perdre le travail fait pour établir la SA IKE. Voir au paragraphe 2.21 une liste des messages d'erreur qui peuvent être produits lors de l'échec d'une création de SA fille.

1.3.2 Changer les clés de SA IKE avec l'échange CREATE_CHILD_SA

La demande CREATE_CHILD_SA pour changer les clés d'une SA fille est :

Initiateur **Répondant**
HDR, SK {SA, Ni, KEi} →

L'initiateur envoie une ou des offres de SA dans la charge utile SA, un nom occasionnel dans la charge utile Ni, et une valeur Diffie-Hellman dans la charge utile KEi. La charge utile KEi DOIT être incluse. Un nouvel SPI d'initiateur est fourni dans le champ SPI de la charge utile de SA. Une fois qu'un homologue a reçu une demande de changement de clés d'une SA IKE ou a envoyé une demande de changement de clé d'une SA IKE, il NE DEVRAIT PAS commencer de nouvel échange CREATE_CHILD_SA sur la SA IKE qui est en cours de changement de clés.

La réponse CREATE_CHILD_SA pour changer les clés d'une SA fille est :

← HDR, SK {SA, Nr, KEr}

Le répondant réplique (en utilisant le même identifiant de message pour répondre) avec l'offre acceptée dans une charge utile SA, un nom occasionnel dans la charge utile Nr, et une valeur Diffie-Hellman dans la charge utile KEr si la suite cryptographique choisie inclut ce groupe. Un nouveau SPI de répondant est fourni dans le champ SPI de la charge utile de la SA.

La nouvelle SA IKE a ses compteurs de messages réglés à 0, sans considération de ce qu'ils étaient dans la SA IKE antérieure. La première demande IKE des deux côtés sur la nouvelle SA IKE aura l'identifiant de message 0. La vieille SA IKE conserve sa numérotation, de sorte que toute demande ultérieure (par exemple, pour supprimer la SA IKE) aura une numérotation consécutive. La nouvelle SA IKE a sa taille de fenêtre remise à 1, et l'initiateur dans cet échange de changement de clés est le nouvel "initiateur d'origine" de la nouvelle SA IKE.

Le paragraphe 2.18 traite aussi le changement de clé de SA IKE en détails.

1.3.3 Changement de clé des SA filles avec l'échange CREATE_CHILD_SA

La demande CREATE_CHILD_SA pour changer les clés d'une SA fille est :

Initiateur **Répondant**
HDR, SK {N (REKEY_SA), 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 pour la SA fille proposée dans les charges utiles TSi et TSr.

Les notifications décrites au paragraphe 1.3.1 peuvent aussi être envoyées dans un échange de changement de clés. Généralement elles seront les mêmes notifications que celles utilisées dans l'échange original ; par exemple, lors d'un changement de clés de SA en mode transport, la notification USE_TRANSPORT_MODE sera utilisée.

La notification REKEY_SA DOIT être incluse dans un échange CREATE_CHILD_SA si l'objet de l'échange est de remplacer une SA ESP ou AH existante. La SA qui change de clés est identifiée par le champ SPI dans la charge utile Notify ; c'est le SPI que l'initiateur de l'échange attendrait dans des paquets ESP ou AH entrants. Il n'y a pas de données associées à ce type de message Notify. Le champ Identifiant de protocole de la notification REKEY_SA est réglé de façon à correspondre au protocole de la SA dont on change les clés, par exemple, 3 pour ESP et 2 pour AH.

La réponse CREATE_CHILD_SA pour changer les clés d'une SA fille est :

← 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, un nom occasionnel dans la charge utile Nr, 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.

Les sélecteurs de trafic pour le trafic à envoyer sur cette SA sont spécifiés dans les charges utiles TS dans la réponse, qui peut être un sous-ensemble de ce que proposait l'initiateur de la SA fille.

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. Noter que certains messages d'information, qui ne sont pas des échanges, peuvent être envoyés en dehors du contexte d'une SA IKE. Le paragraphe 2.21 couvre aussi très en détails les messages d'erreur.

Les messages de commande qui appartiennent à une IKE_SA DOIVENT être envoyés sous cette SA IKE. Les messages de commande qui appartiennent aux SA filles DOIVENT être envoyés sous la protection de la SA IKE qui les a générés (ou son successeur si la SA IKE a subi 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 vide. 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 l'autre point d'extrémité de vérifier qu'il est en vie.

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.

1.4.1 Suppression d'une SA avec des échanges INFORMATIONAL

Les SA ESP et AH existent toujours par paires, avec une SA dans chaque direction. Lorsque une SA est fermée, les deux membres de la paire DOIVENT être fermés (c'est-à-dire, supprimés). 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 ils devraient se trouver dans les en-têtes des paquets entrants) des SA à supprimer. Le receveur DOIT clore les SA désignées. Noter qu'on n'envoie jamais de charges utiles Delete pour les deux côtés d'une SA dans un seul message. Si il y a de nombreuses SA à supprimer en même temps, on inclut les charges utiles Delete pour la moitié entrante de chaque paire de SA dans l'échange INFORMATIONAL.

Normalement, la réponse dans 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, chacune 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.

Comme pour les SA ESP et AH, les SA IKE sont aussi supprimées par l'envoi d'un échange INFORMATIONAL. Supprimer une SA IKE clôt implicitement toutes les SA filles restantes négociées sous elle. La réponse à une demande qui supprime la SA IKE est une réponse INFORMATIONAL vide.

Un nœud DEVRAIT regarder les connexions ESP ou AH à moitié closes comme des anomalies et un nœud qui a des capacités d'audit devrait faire un audit de leur existence si elles persistent. Noter que la présente spécification ne spécifie pas 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 comme décrit ci-dessus. Il peut alors reconstruire les SA dont il a besoin sur une base saine sous une nouvelle IKE_SA.

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

Il y a certains cas où un nœud reçoit un paquet qu'il ne peut pas traiter, mais il peut vouloir notifier cette situation à l'envoyeur.

- Si un paquet ESP ou AH arrive avec un SPI non reconnu. Cela peut être dû à ce que le nœud receveur a eu une panne récente et a perdu l'état, ou parce qu'un autre système fonctionne mal ou l'attaque.

- Si un paquet de demande IKE chiffré arrive sur l'accès 500 ou 4500 avec un indice SPI IKE 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 un paquet de demande IKE arrive avec un numéro de version majeure supérieur à ce que la mise en œuvre prend en charge.

Dans le premier cas, si le nœud receveur a une SA IKE active pour l'adresse IP d'où est venu le paquet, il PEUT envoyer une notification INVALID_SPI du paquet capricieux sur cette SA IKE dans un échange INFORMATIONAL. Les données de notification contiennent le SPI du paquet invalide. Le receveur de cette notification ne peut pas dire si le SPI est pour AH ou ESP, mais ce n'est pas important parce que dans de nombreux cas, les SPI seront différents pour les deux. Si il n'existe aucune

SA IKE convenable, le nœud PEUT envoyer un message d'information sans protection cryptographique à l'adresse IP de source, en utilisant l'accès UDP de source comme accès de destination si le paquet était UDP (ESP ou AH encapsulé dans UDP). Dans ce cas, cela devrait seulement être utilisé par le receveur comme une indication que peut-être quelque chose irait de travers (parce cela peut facilement être falsifié). Ce message ne fait pas partie d'un échange INFORMATIONAL, et le nœud receveur NE DOIT PAS y répondre parce que cela pourrait causer une boucle de messages. Le message est construit comme suit : il n'y a pas de valeur de SPI IKE qui serait significative pour le receveur d'une telle notification ; utiliser des valeurs de zéro ou des valeurs aléatoires est acceptable, ceci étant l'exception à la règle du paragraphe 3.1 qui interdit les SPI d'initiateur IKE de zéro. Le fanion Initiateur est réglé à 1, le fanion Réponse est réglé à 0, et les fanions de version sont réglés de façon normale ; ces fanions sont décrits au paragraphe 3.1.

Dans le second cas et le troisième, le message est toujours envoyé sans protection cryptographique (en dehors d'une SA IKE) et inclut soit une notification INVALID_IKE_SPI, soit une notification INVALID_MAJOR_VERSION (sans données de notification). Le message est un message de réponse, et il est donc envoyé à l'adresse IP et à l'accès d'où il est venu avec les mêmes SPI IKE, identifiant de message et type d'échange que copié de la demande. Le fanion Réponse est réglé à 1, et les fanions Version sont réglés de façon normale.

1.6. Terminologie des exigences

La définition des termes de primitives dans ce document (comme Association de sécurité ou SA) se trouve dans la [RFC4301]. On notera que certaines parties de IKEv2 s'appuient sur les règles de traitement de la [RFC4301], comme décrit dans diverses sections de ce document.

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" en majuscules dans ce document sont à interpréter comme décrit dans le BCP 14, [RFC2119].

1.7 Différences significatives avec la RFC 4306 et la RFC 5996

Le présent document contient des éclaircissements et des développements à IKEv2 [RFC4306]. Beaucoup des éclaircissements se fondent sur la [RFC4718]. Les changements apportés par le présent document ont été discutés dans le groupe de travail IPsec et, après sa dissolution, sur la liste de diffusion IPsec. Le présent document contient des explications détaillées des zones qui n'étaient pas claires dans IKEv2, et il est donc utile pour les mises en œuvre de IKEv2.

Le protocole décrit dans le présent document conserve le même numéro de version majeur (2) et mineur (0) que dans la RFC 4306. C'est-à-dire que le numéro de version *n'est pas* changé par rapport à la RFC 4306. Le petit nombre de changements techniques apportés ici n'est pas supposé affecter les mises en œuvre de la RFC 4306 qui ont déjà été déployées au moment de la publication du présent document.

Le présent document rend les figures et les références un peu plus cohérentes qu'elles ne l'étaient dans la [RFC4306].

Les développeurs de IKEv2 ont noté que les exigences de niveau DEVRAIT dans la RFC 4306 sont souvent peu claires en ce qu'elles ne disent pas quand il n'y a pas d'obstacle au non respect des exigences. Ils ont aussi noté qu'il y a des exigences de niveau DOIT qui ne sont pas en rapport avec l'interopérabilité. Le présent document donne plus d'explications sur certaines de ces exigences. Toutes les utilisations en minuscules des mots DEVRAIT et DOIT ont maintenant leur sens français normal, et non le sens de l'interopérabilité de la [RFC2119].

Les développeurs de IKEv2 (et de IKEv1) ont noté qu'il y a une grande quantité de matériaux dans les tableaux de codes du paragraphe 3.10.1 de la RFC 4306. Cela fait que les développeurs n'ont pas toutes les informations nécessaires dans le corps principal du document. Beaucoup des matériaux provenant de ces tableaux ont été déplacés dans les parties associées du corps principal du document.

Le présent document supprime la discussion sur l'incorporation de AH et ESP. C'était une erreur dans la RFC 4306 causée par le délai entre l'achèvement de la RFC 4306 et celui de la RFC 4301. Fondamentalement, IKEv2 se fonde sur la RFC 4301, qui ne comporte pas de "faisceaux de SA" qui font partie de la RFC 2401. Alors qu'un seul paquet peut passer plusieurs fois par le traitement IPsec, chacun de ces passages utilise une SA séparée, et les passages sont coordonnés par les tableaux de transmissions. Dans IKEv2, chacune de ces SA doit être créée en utilisant un échange CREATE_CHILD_SA séparé.

Le présent document supprime la discussion de l'attribut de configuration INTERNAL_ADDRESS_EXPIRY parce que sa mise en œuvre est très problématique. Les mises en œuvre qui se conforment au présent document DOIVENT ignorer les propositions qui ont le type d'attribut de configuration 5, la vieille valeur pour INTERNAL_ADDRESS_EXPIRY. Le présent document supprime aussi INTERNAL_IP6_NBNS comme attribut de configuration.

Le présent document supprime la faculté de rejeter les messages dans lesquels les charges utiles ne sont pas dans le "bon" ordre ; maintenant, les mises en œuvre NE DOIVENT PAS les rejeter. Ceci est dû au manque de clarté quant à l'ordre dans lequel les charges utiles sont décrites.

Les listes d'éléments provenant de la RFC 4306 qui se retrouvaient dans le registre de l'IANA se réduisaient aux seuls éléments qui étaient effectivement définis dans la RFC 4306. Aussi, beaucoup de ces listes sont maintenant précédés par la très importante instruction aux développeurs qu'ils devraient regarder le registre de l'IANA au moment du développement parce que de nouveaux éléments ont été ajoutés depuis la RFC 4306.

Le présent document apporte des éclaircissements sur le moment où les notifications sont envoyées chiffrées et celui où elles ne le sont pas, selon l'état de la négociation à cet instant.

Le présent document discute plus en détails comment négocier les chiffrements en mode combiné.

Au paragraphe 1.3.2, "La charge utile KEi DEVRAIT être incluse" a été changé en "La charge utile KEi DOIT être incluse". Cela a aussi conduit à des changements au paragraphe 2.18.

Au paragraphe 2.1, il y a un nouveau texte qui couvre la façon dont les SPI de l'initiateur et/ou IP sont utilisés pour différencier si c'est une SA IKE "semi ouverte" ou une nouvelle demande.

Le présent document précise l'utilisation du fanion critique au paragraphe 2.5.

Au paragraphe 2.8, "Noter que, lors d'un changement de clé, la nouvelle SA fille PEUT avoir des sélecteurs de trafic et des algorithmes différents de ceux de la vieille" a été changé en "Noter que, lors d'un changement de clé, la nouvelle SA fille NE DEVRAIT PAS avoir des sélecteurs de trafic et des algorithmes différents de ceux de la vieille".

Le nouveau paragraphe 2.8.2 traite du changement simultané de clés de SA IKE.

Le présent document ajoute au paragraphe 2.13 la restriction que toutes les fonctions pseudo aléatoires (PRF) utilisées dans IKEv2 DOIVENT prendre des clés de taille variable. Cela ne devrait pas affecter les mises en œuvre parce qu'il n'y avait pas de PRF normalisée avec des clés de taille fixe.

Le paragraphe 2.18 exige de faire un échange Diffie-Hellman lors d'un changement de clé de la SA IKE. En théorie, la RFC 4306 permettait une politique où l'échange Diffie-Hellman était facultatif, mais ce n'était pas utile (ou approprié) lors du changement de clés de la SA IKE.

Le paragraphe 2.21 a été largement développé pour couvrir les différents cas où des réponses d'erreur sont nécessaires ainsi que leurs réponses appropriées.

Le paragraphe 2.23 précise que, dans une traversée de NAT, les paquets IPsec, aussi bien encapsulés dans UDP que non, doivent être bien compris à la réception.

Ajout du paragraphe 2.23.1 pour décrire la traversée de NAT lorsque le mode transport est demandé.

Ajout du paragraphe 2.25 pour expliquer comment agir quand il y a des collisions de programmation dans la suppression et/ou le changement de clés des SA, et deux nouvelles notifications d'erreur (TEMPORARY_FAILURE et CHILD_SA_NOT_FOUND) ont été définies.

Au paragraphe 3.6, "Les mises en œuvre DOIVENT prendre en charge le schéma "http:" pour les recherches d'URL et de hachage. Le comportement des autres schéma d'URL n'est actuellement pas spécifié, et de tels schémas NE DEVRAIENT PAS être utilisés en l'absence d'un document les spécifiant" a été ajouté.

Au paragraphe 3.15.3, un pointeur sur un nouveau document relatif à la configuration des adresses IPv6 a été ajouté.

L'Appendice C a été étendu et précisé.

1.8 Différences entre la RFC 5996 et le présent document

Il est précisé dans le résumé et l'introduction que le statut du document est celui de "Norme de l'Internet".

Le nouveau paragraphe 2.9.2 couvre les sélecteurs de trafic dans le changement de clés.

Ajout d'une référence à la RFC 6989 lors de la réutilisation des exponentielles Diffie-Hellman (paragraphe 2.12).

Ajout du nom "Last Substruc" pour l'en-tête Proposition de sous structure et Sous structure de transformation (paragraphes 3.3.1 et 3.3.2) pour le champ 0 (dernier) ou 2/3 (plus).

Ajout d'une référence à la RFC 6989 lors de l'utilisation de groupes qui ne sont pas des groupes d'exponentiation modulaire de Sophie Germain (MODP, *Sophie Germain Modular Exponentiation*) (paragraphe 3.3.2).

Ajout d'une référence à la RFC 4945 au paragraphe sur les charges utiles d'identification (paragraphe 3.5).

Au paragraphe 3.6, on déconseille les clés publiques RSA brutes. Il y a un nouveau travail en cours qui ajoute un format plus générique pour les clés publiques brutes.

Correction des paragraphes 3.6 et 3.10 comme spécifié dans les errata pour la RFC5996 (Identifiants d'errata 2707 et 3036).

Ajout d'une note aux Considérations relatives à l'IANA (Section 6) sur le fait de déconseiller la clé brute RSA, et suppression du vieux contenu (ce qui était déjà fait durant le traitement de la RFC5996). Ajout d'une note selon laquelle l'IANA devrait mettre à jour toutes les références à la RFC 5996 pour pointer sur le présent document.

2 Détails du protocole IKE et variantes

IKE écoute et envoie normalement 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 le paragraphe 2.23). Comme UDP est un protocole de datagrammes (non fiable) IKE inclut dans sa définition la récupération des erreurs de transmission, incluant les pertes de paquet, la répétition de paquet, et la falsification de paquet. IKE est conçu pour fonctionner tant que (1) au moins un d'une série de paquets retransmis atteint sa destination avant la fin de temporisation, et (2) le canal n'est pas si plein de paquets falsifiés et répétés que le réseau ou les capacités de CPU d'un des points d'extrémité en soient épuisés .Même en l'absence de ces exigences minimum de performances, IKE est conçu pour échouer proprement (même si le réseau était cassé).

Bien qu'il soit prévu que les messages IKEv2 soient courts, ils contiennent des structures sans limite supérieure ferme de taille (en particulier, les certificats numériques) et IKEv2 lui-même n'a pas de mécanisme pour fragmenter les grands messages. IP définit un mécanisme pour la fragmentation des messages UDP surdimensionnés, mais les mises en œuvre varient quant à la taille maximum de message prise en charge. De plus, l'utilisation de la fragmentation IP ouvre la porte à la mise en œuvre d'attaques de déni de service (DoS) [DOS]. Finalement, certaines mises en œuvre de NAT ou de 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 font jusqu'à 1280 octets, et elles DEVRAIENT être capables d'envoyer, recevoir, et traiter les messages qui font jusqu'à 3000 octets. Les mises en œuvre de IKEv2 doivent connaître la taille maximum de message UDP prise en charge et PEUVENT raccourcir les messages en écartant certains certificats ou propositions de suite de chiffrement si cela garde les messages en dessous du maximum. L'utilisation des formats "hachage et URL" plutôt que d'inclure les certificats dans les échanges lorsque possible peut éviter la plupart des problèmes. Les mises en œuvre et configurations doivent cependant se rappeler que si les recherches d'URL ne sont possibles qu'après l'établissement de la SA fille, des problèmes de récurrence pourraient empêcher cette technique de fonctionner.

La charge utile UDP de tous les paquets contenant des messages IKE envoyés sur l'accès 4500 DOIT commencer par le préfixe de quatre zéros ; autrement, le receveur ne saura pas comment les traiter.

2.1 Utilisation de temporisateurs de retransmission

Dans IKE, tous les messages existent par paire : une demande et une réponse. L'établissement d'une SA IKE consiste normalement en deux échanges. Une fois la SA IKE établie, l'une ou l'autre extrémité de l'association de sécurité peut initier des demandes à tout moment, et il peut y avoir de nombreuses demandes et réponses "en vol" à tout instant. Mais chaque message est étiqueté comme demande ou réponse, et pour chaque échange, 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 en cas de fin de temporisation. Le répondant NE DOIT jamais retransmettre une réponse à moins qu'il reçoive une retransmission de la demande. Dans ce cas, le répondant DOIT ignorer la demande retransmise sauf dans la mesure où elle cause 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 est supérieur ou égal au numéro de séquence dans la réponse plus sa taille de fenêtre (voir le paragraphe 2.3). Afin d'économiser la mémoire, il est permis aux répondants d'oublier la réponse après une temporisation de plusieurs minutes. Si le répondant reçoit une demande retransmise pour laquelle il a déjà oublié la réponse, il DOIT ignorer la demande (et non, par exemple, tenter de construire une nouvelle réponse).

IKE est un protocole fiable : l'initiateur DOIT retransmettre une demande jusqu'à ce que soit il reçoive une réponse correspondante, soit supposer que la SA IKE a une défaillance. Dans ce dernier cas, l'initiateur élimine tout l'état associé à la SA IKE et toutes les SA filles qui ont été négociées en utilisant cette SA IKE. Une retransmission par l'initiateur DOIT être identique au bit près à la demande d'origine. C'est-à-dire que, tout commençant à partir de l'en-tête IKE (à partir du SPI de l'initiateur de la SA IKE) doit être identique au bit près ; les éléments avant lui (comme les en-têtes IP et UDP) n'ont pas à être identiques.

Les retransmissions de la demande IKE_SA_INIT exigent un traitement un peu particulier. Quand un répondant reçoit une demande IKE_SA_INIT, il doit déterminer si le paquet est une retransmission appartenant à une SA IKE "semi ouverte" existante (dans ce cas, le répondant retransmet la même réponse) ou une nouvelle demande (auquel cas le répondant crée une nouvelle SA IKE et envoie une réponse fraîche) ou si elle appartient à une SA IKE existante dont la demande IKE_AUTH a déjà été reçue (auquel cas le répondant l'ignore).

Il n'est pas suffisant d'utiliser le SPI et/ou l'adresse IP de l'initiateur pour différencier entre ces trois cas parce que deux homologues différents derrière un seul NAT pourraient choisir le même SPI d'initiateur. À la place, un répondant robuste va faire la recherche de SA IKE en utilisant le paquet entier, son hachage, ou la charge utile Ni.

La politique de retransmission pour les messages unilatéraux est assez différente de celle des messages réguliers. Comme aucun accusé de réception n'est jamais envoyé, il n'y a pas de raison de retransmettre gratuitement des messages unilatéraux. Étant donné que tous ces messages sont des erreurs, il y a du sens à ne les envoyer qu'une fois par paquet "délictueux", et ne retransmettre que si plus de paquets délictueux sont reçus. Là encore, il est raisonnable de limiter les retransmissions de tels messages d'erreur.

2.2 Utilisation de numéros de séquence pour les identifiants de message

Chaque message IKE contient un identifiant de message au titre de son en-tête fixe. Cet identifiant de message est utilisé pour confronter les demandes et les réponses et pour identifier les retransmissions de messages. La retransmission d'un message DOIT utiliser le même identifiant de message que le message d'origine.

L'identifiant de message est une quantité de 32 bits, qui est zéro pour les messages IKE_SA_INIT (incluant les réessais du message dus à des réponses comme COOKIE et INVALID_KE_PAYLOAD) et est incrémentée pour chaque échange suivant. Donc, la première paire de messages IKE_AUTH aura un ID de 1, la secondr (quand EAP est utilisé) va être 2, et ainsi de suite. L'identifiant de message est remis à zéro dans la nouvelle SA IKE après que la SA IKE a changée de clés.

Chaque point d'extrémité dans l'association de sécurité IKE maintient deux identifiants de message "courants" : le prochain à être utilisé 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 des demandes sont générées et reçues. Les réponses contiennent toujours le même identifiant de message que la demande correspondante. Cela signifie que après l'échange initial, chaque entier n peut apparaître comme identifiant de message dans quatre messages distincts : la énième demande provenant de l'initiateur IKE d'origine, la réponse correspondante, la énième demande provenant du répondant IKE original, et la réponse correspondante. Si les deux extrémités font un nombre de demandes très différent, 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 fanions Initiateur et Réponse dans l'en-tête de message spécifient, pour un message particulier, lequel de ces quatre messages il est.

Tout au long du présent document, "initiateur" se réfère à la partie qui initie l'échange décrit. Le "initiateur original" se réfère toujours à la partie qui a initié l'échange qui a résulté en la SA IKE actuelle. En d'autres termes, si le "répondant original" commence le changement de clés de la SA IKE, cette partie devient le "initiateur original" de la nouvelle SA IKE.

Noter que les identifiants de messages sont protégés cryptographiquement et fournissent une protection contre les répétitions de message. Dans le cas peu probable où les identifiants de message deviendraient trop gros pour tenir dans 32 bits, la SA IKE DOIT être close ou changer de clés.

2.3 Taille de fenêtre pour demandes en chevauchement

La notification SET_WINDOW_SIZE affirme que le point d'extrémité envoyeur est capable de conserver 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 faire 4 octets et contenir la représentation en ordre gros boutien du nombre de messages que l'envoyeur promet de conserver. La taille de fenêtre est toujours de un jusqu'à ce que les échanges initiaux s'achèvent.

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

Après l'établissement d'une SA IKE, afin de maximiser le débit de IKE, un point d'extrémité IKE PEUT produire plusieurs demandes avant d'obtenir une réponse à l'une d'elles, jusqu'à la limite établie par le SET_WINDOW_SIZE de son homologue. Ces demandes peuvent se dépasser sur le réseau. Un point d'extrémité IKE DOIT être prêt à accepter et traiter une demande alors qu'il a une demande en instance afin d'éviter un blocage dans cette situation. Un point d'extrémité IKE peut aussi accepter et traiter plusieurs demandes alors qu'il a une demande en instance.

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

Un point d'extrémité IKE qui prend en charge une taille de fenêtre supérieure à un devrait être capable de traiter les demandes entrantes déclassées afin de maximiser les performances en cas de défaillances du réseau ou de réarrangement de l'ordre des paquets.

La taille de fenêtre est normalement une propriété (éventuellement configurable) d'une mise en œuvre particulière, et elle est sans rapport avec le contrôle d'encombrement (à la différence de la taille de fenêtre dans TCP, par exemple). En particulier, ce que le répondant devrait faire quand il reçoit une notification SET_WINDOW_SIZE contenant une valeur plus petite que ce qui est actuellement en vigueur n'est pas défini. Donc, il n'y a actuellement pas de moyen de réduire la taille de fenêtre d'une SA IKE existante ; on peut seulement l'augmenter. Lorsque on change les clés d'une SA IKE, la nouvelle SA IKE commence par la taille de fenêtre 1 jusqu'à ce qu'elle soit explicitement augmentée par l'envoi d'une nouvelle notification SET_WINDOW_SIZE.

La notification INVALID_MESSAGE_ID *(identifiant de message invalide)* est envoyée quand un identifiant de message IKE est reçu en dehors de la fenêtre prise en charge. Ce message Notify NE DOIT PAS être envoyé dans une réponse ; la demande invalide NE DOIT PAS recevoir d'accusé de réception. À la place, on informe l'autre côté en initiant un échange INFORMATIONAL avec des données de notification contenant l'identifiant de message invalide de quatre octets. Envoyer cette notification est FACULTATIF, et les notifications de ce type DOIVENT être limitées en débit.

2.4 Synchronisation d'état et fins de temporisation de connexion

Il est permis à un point d'extrémité IKE d'oublier tous ses états associés à une SA IKE et la collection des SA filles correspondantes à tout moment. C'est le comportement prévu dans le cas où un point d'extrémité a une défaillance puis redémarre. Il est important que quand un point d'extrémité a une défaillance ou réinitialise son état, l'autre point d'extrémité détecte ces conditions et ne continue pas à gaspiller la bande passante du réseau en envoyant des paquets sur des SA éliminées et les voir tomber dans un trou noir.

La notification INITIAL_CONTACT certifie que cette SA IKE est la seule SA IKE actuellement active entre les identités authentifiées. Elle PEUT être envoyée quand une SA IKE est établie après une panne, et le receveur PEUT utiliser cette information pour supprimer toutes les autres SA IKE qu'il a à 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'un utilisateur en itinérance où l'utilisateur est autorisé à se connecter au pare-feu d'entreprise à partir de deux systèmes distants en même temps). La notification INITIAL_CONTACT, si elle est envoyée, DOIT être dans la première demande ou réponse IKE_AUTH, et non comme un échange séparé après coup ; les parties receveuses PEUVENT l'ignorer dans les autres messages.

Comme IKE est conçu pour fonctionner en dépit des attaques de déni de service provenant du réseau, un point d'extrémité NE DOIT PAS conclure que l'autre point d'extrémité a une défaillance sur la base d'informations d'acheminement (par exemple, des messages ICMP) ou de messages IKE qui arrivent sans protection cryptographique (par exemple, des messages Notify qui se plaindraient de SPI inconnues). Un point d'extrémité DOIT conclure que l'autre point d'extrémité a eu une défaillance seulement quand des tentatives répétées de le contacter sont restées sans réponse pendant une période de temporisation ou quand une notification INITIAL_CONTACT cryptographiquement protégée est reçue sur une SA IKE différente à la même identité authentifiée. Un point d'extrémité devrait suspecter que l'autre point d'extrémité a eu une défaillance sur la base d'informations d'acheminement et initier une demande pour voir si l'autre point d'extrémité est en vie. Pour vérifier si l'autre côté est en vie, IKE spécifie une demande INFORMATIONAL vide qui (comme toutes les demandes IKE) exige un accusé de réception (noter que dans le contexte d'une SA IKE, un message "vide" consiste en un en-tête IKE suivi par une charge utile Chiffré *(Encrypted)* qui ne contient pas de charge utile). Si un message cryptographiquement protégé (frais, c'est-à-dire, non retransmis) a été reçu en provenance de l'autre côté récemment, les messages Notify non protégés PEUVENT être ignorés. Les mises en œuvre DOIVENT limiter le taux d'entreprise d'actions fondées sur des messages non protégés.

Le nombre d'essais et la longueur des temporisations ne sont pas traités dans la présente spécification parce qu'ils n'affectent pas l'interopérabilité. Il est suggéré que les messages soient retransmis au moins une douzaine de fois sur une période d'au moins plusieurs minutes avant d'abandonner une SA, mais des environnements différents peuvent exiger des règles différentes. Pour être un bon citoyen du réseau, les intervalles de retransmission DOIVENT augmenter exponentiellement pour éviter d'inonder le réseau et rendre encore pire une situation d'encombrement existante. Si il y a seulement eu du trafic sortant sur toutes les SA associées à une SA IKE, il est essentiel de confirmer la vivacité de l'autre point d'extrémité pour éviter des trous noirs. Si aucun message cryptographiquement protégé n'a été reçu sur une SA IKE ni sur une de ses SA filles récemment, le système a besoin d'effectuer une vérification de vivacité afin de prévenir l'envoi de messages à un homologue mort. (C'est parfois appelé la "détection de l'homologue mort" (DPD, *dead peer detection*) bien qu'en réalité ce sont les homologues vivants, non les morts, qui sont détectés.) La réception d'un message frais protégé cryptographiquement sur une SA IKE ou une de ses SA filles assure la vivacité de la SA IKE et de toutes ses SA filles. Noter que ceci fait peser des exigences sur les modes de défaillance d'un point d'extrémité IKE. Une mise en œuvre a besoin d'arrêter d'envoyer sur une SA si une défaillance l'empêche de recevoir sur toutes les SA associées. Si un système crée des SA filles qui peuvent défaillir indépendamment les unes des autres sans que la SA IKE associée soit capable d'envoyer un message de suppression, le système DOIT alors négocier de telles SA filles en utilisant des SA IKE séparées.

Un type d'attaque de DoS sur l'initiateur d'une SA IKE peut être évité si l'initiateur prend les soins appropriés : comme les deux premiers messages d'un établissement de SA ne sont pas cryptographiquement protégés, un attaquant pourrait répondre au message de l'initiateur avant le répondant authentique et empoisonner la tentative d'établissement de connexion. Pour empêcher cela, l'initiateur PEUT vouloir accepter plusieurs réponses à son premier message, traiter chaque réponse comme potentiellement légitime, répondre à chacune d'elles, et ensuite éliminer toutes les connexions semi ouvertes invalides quand il reçoit une réponse valide cryptographiquement protégée à une de ses demandes. Une fois qu'une réponse cryptographiquement valide est reçue, toutes les réponses suivantes 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 s'accorder sur la durée de vie d'une SA. Si IKE présume 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 SA filles établies par cette SA IKE sont alors supprimées.

Un point d'extrémité IKE peut à tout moment supprimer des SA filles inactives pour récupérer les ressources utilisées pour garder leur état. Si un point d'extrémité IKE choisit de supprimer des SA filles, il DOIT envoyer des charges utiles Supprime *(Delete)* à l'autre extrémité lui notifiant la suppression. Il PEUT de façon similaire laisser la SA IKE arriver en fin de temporisation. Clore implicitement la SA IKE clôt toutes les SA filles associées. Dans ce cas, un point d'extrémité IKE DEVRAIT envoyer une charge utile Supprime indiquant qu'il a clôt la SA IKE sauf si l'autre point d'extrémité ne répond plus.

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 2 et le numéro de version mineur est 0. Le présent document est un remplacement de la [RFC4306]. Il est probable que certaines mises en œuvre voudront prendre en charge la version 1.0 et la version 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 de façon si radicale qu'un nœud d'une version plus ancienne ne serait plus capable d’inter-opérer avec un nœud d'une version plus récente si il a simplement ignoré les champs qu'il ne comprend pas et a accompli les actions spécifiées dans la version plus ancienne de la spécification. Le numéro de version mineur indique de nouvelles capacités, et DOIT être ignoré par un nœud qui a un numéro de version mineur plus petit, mais utilisé pour des besoins d'information par le nœud qui a le plus grand numéro de version mineur. Par exemple, il peut indiquer la capacité de traiter un nouveau type de message Notify. Le nœud qui a le plus grand numéro de version mineur va simplement noter que son correspondant ne va pas être capable de comprendre ce message et donc, il ne va pas l'envoyer.

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 Notify non authentifié de type INVALID_MAJOR_VERSION contenant le plus fort (le plus proche) numéro de version qu'il supporte. Si un point d'extrémité prend en charge la version majeure n, et la version majeure m, il DOIT supporter toutes les versions entre n et m. Si il reçoit un message avec une version majeure qu'il supporte, il DOIT répondre avec ce numéro de version. Afin d'empêcher deux nœuds de se faire piéger à correspondre avec un numéro de version majeur inférieur au maximum qu'il prennent tous deux en charge, IKE a un fanion qui indique que le nœud est capable de comprendre un numéro de version majeur supérieur.

Donc, le numéro de version majeur dans l'en-tête IKE indique le numéro de version du message, et non le plus haut numéro de version que l'émetteur prend en charge. Si l'initiateur est capable de prendre en charge les versions n, n+1, et n+2, et si le répondant est capable de prendre en charge les versions n et n+1, ils vont alors négocier la n+1, et l'initiateur va établir un fanion indiquant sa capacité de prendre en charge une version supérieure. Si ils négocient par erreur (peut-être à cause d'un attaquant actif qui envoie des messages erronés) la version n, tous deux vont remarquer que l'autre côté peut prendre en charge un numéro de version supérieur, et ils DOIVENT rompre la connexion et se reconnecter en utilisant la version n+1.

Noter que IKEv1 ne suit pas ces règles, parce que il n'y a aucun moyen dans la v1 de noter qu'on est capable de prendre en charge un numéro de version supérieur. Ainsi un attaquant actif peut tromper deux nœuds à capacité v2 et les amener à utiliser la v1. Quand un nœud à capacité v2 négocie un repli sur la v1, il devrait noter ce fait dans ses journaux d'événements.

Aussi, pour la compatibilité future, tous les champs marqués RÉSERVÉ DOIVENT être réglés à zéro par une mise en œuvre fonctionnant avec la version 2.0, et leur contenu DOIT être ignoré par une mise en œuvre fonctionnant avec la version 2.0 ("soyez conservateur dans ce que vous envoyez et libéral dans ce que vous recevez" [RFC0791]). De cette façon, les futures versions du protocole pourront utiliser ces champs d'une façon dont il est garanti qu'elle sera ignorée par les mises en œuvre qui ne les comprennent pas. De façon similaire, les types de charge utile qui ne sont pas définis sont réservés pour une utilisation future ; les mises en œuvre d'une version où ils sont indéfinis DOIVENT sauter par dessus ces charges utiles et ignorer leur contenu.

IKEv2 ajoute un fanion "critique" à chaque en-tête de charge utile pour plus de souplesse en vue de la compatibilité future. Si le fanion critique est établi et si le type de charge utile n’est pas reconnu, le message DOIT être rejeté et la réponse à la demande IKE contenant cette charge utile DOIT inclure une charge utile Notify UNSUPPORTED_CRITICAL_PAYLOAD, indiquant qu'une charge utile critique non prise en charge était incluse. Dans cette charge utile Notify, les données de notification contiennent le type de charge utile de un octet. Si le fanion critique n'est pas établi et si le type de charge utile n'est pas pris en charge, cette charge utile DOIT être ignorée. Les charges utiles envoyées dans les messages de réponse IKE NE DOIVENT PAS avoir le fanion critique établi. Noter que le fanion critique s'applique seulement au type de charge utile, pas au contenu. Si le type de charge utile est reconnu, mais si la charge utile contient quelque chose qui ne l'est pas (comme une transformation inconnue à l'intérieur d'une charge utile de SA, ou un type de message Notify inconnu à l'intérieur de la charge utile Notify) le fanion critique est ignoré.

Bien que de nouveaux types de charge utile puissent être ajoutés à l'avenir et puissent apparaître entrelacés avec les champs définis dans la présente spécification, les mises en œuvre DEVRAIENT envoyer les charges utiles définies dans cette spécification dans l'ordre montré par les Figures des Sections 1 et 2 ; les mises en œuvre NE DOIVENT PAS rejeter comme invalide un message avec ces charges utiles dans un autre ordre.

2.6 SPI et mouchards de SA IKE

Les deux champs initiaux de huit octets dans l'en-tête, appelés les "SPI IKE", sont utilisés comme identifiant de connexion au début des paquets IKE. Chaque point d'extrémité choisit un des deux SPI et DOIT le choisir de façon qu'il soit un identifiant unique d'une SA IKE. Une valeur de SPI de zéro est spéciale : elle indique que la valeur de SPI distante n'est pas encore connue de l'envoyeur.

Les paquets IKE entrants sont transposés en une SA IKE en utilisant seulement le SPI du paquet, et non en utilisant (par exemple) l'adresse IP de source du paquet.

À la différence de ESP et AH où seules 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 SA IKE est toujours envoyé en premier, un point d'extrémité qui a de multiples SA IKE ouvertes qui veut trouver la SA IKE appropriée en utilisant le SPI qu'il a alloué doit regarder le fanion d'initiateur 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 IKE initial, l'initiateur ne va pas savoir la valeur du SPI du répondant et va donc régler ce champ à zéro. Quand l'échange IKE_SA_INIT ne résulte pas en la création d'une SA IKE à cause d'un

INVALID_KE_PAYLOAD, NO_PROPOSAL_CHOSEN, ou COOKIE, le SPI du répondant va être zéro aussi dans le message de réponse. Cependant, si le répondant envoie un SPI de répondant non zéro, l'initiateur devrait ne pas rejeter la réponse pour cette seule raison.

Deux attaques contre IKE auxquelles ont s'attend sont l'épuisement d'état et de CPU, où la cible est inondée de demandes d'initiation de session provenant d'adresses IP fallacieuses. Ces attaques peuvent être rendues moins efficaces si un répondant utilise une CPU minimale et n'engage pas d'état pour une SA avant de savoir si l'initiateur peut recevoir des paquets à l'adresse d'où il prétend être pour les envoyer.

Quand un répondant détecte un grand nombre de SA IKE semi ouvertes, il DEVRAIT répondre aux demandes IKE_SA_INIT avec une réponse contenant la notification COOKIE *(mouchard)*. Les données associées à cette notification DOIVENT faire entre 1 et 64 octets (inclusif) et sa génération est décrite plus loin dans cette section. Si la réponse IKE_SA_INIT comporte la notification COOKIE, l'initiateur DOIT alors réessayer la demande IKE_SA_INIT, et inclure la notification COOKIE contenant les données reçues comme première charge utile, et toutes les autres charges utiles inchangées. L'échange initial va alors être 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 de l'initiateur ni du répondant sauf pour la communication du mouchard. En particulier, les numéros de séquence de message dans les quatre premiers messages vont tous être zéro et les numéros de séquence de message dans les deux derniers messages vont être un. 'A' est le SPI alloué par l'initiateur, tandis que 'B' est le SPI alloué par le répondant.

Une mise en œuvre IKE peut appliquer sa génération de mouchard de répondant d'une façon telle qu'elle n'exige pas de sauvegarder d'état pour reconnaître son mouchard valide quand arrive le second message IKE_SA_INIT. Les algorithmes et la syntaxe exacts utilisés pour générer les mouchards n'affectent pas l'interopérabilité et ne sont donc pas spécifiés ici. Voici un exemple de la façon dont un point d'extrémité pourrait utiliser des mouchards pour mettre en œuvre une protection limitée contre le DoS. Une bonne façon de le faire est de régler le mouchard de répondant comme : Cookie = <VersionIDofSecret> | Hash(Ni | IPi | SPIi | <secret>) où <secret> est un secret généré de façon aléatoire connu seulement du répondant et changé périodiquement et | indique l'enchaînement. <VersionIDofSecret> devrait être changé chaque fois que <secret> est re généré. Le mouchard peut être recalculé quand le IKE_SA_INIT arrive la seconde fois, et comparé au mouchard dans le message reçu. Si il correspond, le répondant sait que le mouchard a été généré depuis le dernier changement en <secret> et que IPi doit être le même que l'adresse de source qu'il a vu la première fois. Incorporer le SPIi dans le calcul assure que si plusieurs SA IKE sont établies en parallèle elles vont toutes avoir des mouchards différents (en supposant que l'initiateur choisisse des SPIi uniques). Incorporer Ni dans le hachage assure qu'un attaquant qui voit seulement le message 2 ne peut pas réussir à forger un faux message 3. Aussi, incorporer le SPIi dans le hachage empêche un attaquant d'aller chercher un mouchard à l'autre extrémité, et ensuite d'initier de nombreux échanges IKE_SA_INIT tous avec des SPI d'initiateur (et peut-être des numéros d'accès) différents de sorte que le répondant pense qu'il y a une série de machines derrière un NAT qui essayent toutes de se connecter.

Si une nouvelle valeur pour <secret> est choisie alors qu'il y a des connexions en cours d'initialisation, un IKE_SA_INIT pourrait être retourné avec autre chose que le <VersionIDofSecret> courant. Le répondant dans ce cas PEUT rejeter le message en envoyant une autre réponse avec un nouveau mouchard ou il PEUT garder la vieille valeur de <secret> pendant un court instant et accepter les mouchards calculés de l'un ou l'autre. Le répondant ne devrait pas accepter des mouchards indéfiniment après que <secret> est changé, car cela réduirait à néant une partie de la protection contre le déni de service. Le répondant devrait changer fréquemment la valeur de <secret>, en particulier en cas d'attaque.

Quand une partie reçoit une demande IKE_SA_INIT contenant un mouchard dont le contenu ne correspond à la valeur attendue, cette partie DOIT ignorer le mouchard et traiter le message comme si aucun mouchard n'avait été inclus ; généralement cela signifie d'envoyer une réponse contenant un nouveau mouchard. L'initiateur devrait limiter le nombre d'échanges de mouchard qu'il essaye avant d'abandonner, éventuellement en utilisant le retard exponentiel. Un attaquant peut contrefaire plusieurs réponses de mouchard au message IKE_SA_INIT de l'initiateur, et chacune de ces réponses de mouchard contrefaites va causer l'envoi de deux paquets : un paquet de l'initiateur au répondant (qui va rejeter ces mouchards) et une réponse du répondant à l'initiateur qui inclut le mouchard correct.

Une note sur la terminologie : le terme de "cookie" a pour origine Karn et Simpson [RFC2522] dans Photuris, une proposition de gestion de clés avec IPsec, et il a perduré. L'en-tête de message fixe du protocole Internet d'association de sécurité et gestion de clés (ISAKMP, *Internet Security Association and Key Management Protocol*) [RFC2408] comporte deux champs de huit

octets appelés "mouchards", et cette syntaxe est utilisée par IKEv1 et IKEv2, bien que dans IKEv2 il y soit fait référence sous le nom de "SPI IKE" et qu'il y ait un nouveau champ séparé dans une charge utile Notify contenant le mouchard.

2.6.1 Interaction de COOKIE et de INVALID_KE_PAYLOAD

Il y a deux raisons courantes pour lesquelles l'initiateur peut avoir à réessayer l'échange IKE_SA_INIT : le répondant demande un mouchard ou veut un groupe Diffie-Hellman différent de celui qui était inclus dans la charge utile KEi. Si l'initiateur reçoit un mouchard provenant du répondant, l'initiateur a besoin de décider si il inclut ou non le mouchard dans la seule prochaine répétition de la demande IKE_SA_INIT, ou aussi dans tous les essais suivants.

Si l'initiateur n'inclut le mouchard que dans le prochain essai, un aller-retour supplémentaire peut être nécessaire dans certains cas. Un aller-retour supplémentaire est nécessaire aussi si l'initiateur inclut le mouchard dans tous les essais, mais le répondant ne prend pas cela en charge. Par exemple, si le répondant inclut les charges utiles KEi dans le calcul de mouchard, il va rejeter la demande en envoyant un nouveau mouchard.

Si les deux homologues prennent en charge l'inclusion du mouchard dans tous les essais, un échange légèrement plus court peut se produire.

Initiateur	**Répondant**
HDR(A,0), SAi1, KEi, Ni -->	
	<-- HDR(A,0), N(COOKIE)
HDR(A,0), N(COOKIE), SAi1, KEi, Ni -->	
	<-- HDR(A,0), N(INVALID_KE_PAYLOAD)
HDR(A,0), N(COOKIE), SAi1, KEi', Ni -->	
	<-- HDR(A,B), SAr1, KEr, Nr

Les mises en œuvre DEVRAIT prendre en charge ce plus court échange, mais NE DOIVENT PAS échouer si d'autres mises en œuvre ne prennent pas en charge ce plus court échange.

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, ou AH) pour la SA ainsi que d'algorithmes de chiffrement associés à chaque protocole.

Une charge utile SA consiste en une ou plusieurs propositions. Chaque proposition inclut un protocole. Chaque protocole contient une ou plusieurs transformations -- chacune spécifiant un algorithme de chiffrement. 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 de chiffrement).

Cette structure hiérarchique a été conçue pour coder efficacement les propositions de suites cryptographiques quand le nombre de suites acceptées est grand parce que de multiples valeurs sont acceptables pour de multiples transformations. Le répondant DOIT choisir une seule suite, qui peut être n'importe quel sous ensemble de la proposition de SA en suivant les règles ci après.

Chaque proposition contient un protocole. Si une proposition est acceptée, la réponse de SA DOIT contenir le même protocole. Le répondant DOIT accepter une seule proposition ou les rejeter toutes et retourner une erreur. L'erreur est donnée dans une notification de type NO_PROPOSAL_CHOSEN.

Chaque proposition de protocole IPsec contient une ou plusieurs transformations. Chaque transformation contient un type de transformation. La suite de chiffrement acceptée DOIT contenir exactement une transformation de chaque type inclus 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_. Donc, six combinaisons sont acceptables.

Si un initiateur propose à la fois des chiffrements normaux avec protection de l'intégrité ainsi que des chiffrements en mode combiné, deux propositions sont alors nécessaires. Une des propositions inclut les chiffrements normaux avec leurs algorithmes d'intégrité, et l'autre proposition inclut tous les chiffrements en mode combiné sans les algorithmes d'intégrité (parce que les chiffrements en mode combiné ne sont pas autorisés à avoir d'algorithme d'intégrité autre que "NONE").

2.8 Changement de clés

Les associations de sécurité IKE, ESP, et AH utilisent des clés secrètes qui devraient n'être en fonction que pour un temps limité et pour protéger une quantité limitée de données. Cela limite la durée de vie de l'association de sécurité entière. Quand la durée de vie d'une association de sécurité expire, l'association de sécurité NE DOIT PAS être utilisée. Si c'est demandé, une nouvelle association de sécurité PEUT être établie. Le re établissement des associations de sécurité pour prendre la place de celles qui sont arrivées à expiration est appelé un changement de clés *(rekeying)*.

Permettre à des mises en œuvre IPsec minimales la capacité de changer les clés des SA sans redémarrer la SA entière est facultatif. Une mise en œuvre PEUT refuser toutes les demandes CREATE_CHILD_SA *(créer une SA fille)* au sein d'une SA IKE. Si une SA est arrivée à expiration ou est sur le point de le faire et si les tentatives de changement de clés en utilisant les mécanismes décrits ici échouent, une mise en œuvre DOIT clore la SA IKE et toutes les SA filles associées et PEUT alors en commencer de nouvelles. Les mises en œuvre peuvent souhaiter prendre en charge le changement de clés en place des SA, car le faire offre de meilleures performances et est à même de réduire probablement le nombre de paquets perdus durant la transition.

Pour changer les clés d'une SA fille au sein d'une SA IKE existante, on crée une nouvelle SA équivalente (voir au paragraphe 2.17) et quand la nouvelle est établie, on supprime la vieille. Noter que, quand on change de clés, la nouvelle SA fille NE DEVRAIT PAS avoir des sélecteurs de trafic et algorithmes différents de ceux de la vieille.

Pour changer les clés d'une SA IKE, établir une nouvelle SA IKE équivalente (voir le paragraphe 2.18) avec l'homologue avec qui la vieille SA IKE est partagée en utilisant un CREATE_CHILD_SA dans le cadre de la SA IKE existante. Une SA IKE ainsi créée hérite de toutes les SA filles de la SA IKE d'origine, et la nouvelle SA IKE est utilisée pour tous les messages de contrôle nécessaires pour le maintient de ces SA filles. Après la création de la nouvelle SA IKE équivalente, l'initiateur supprime la vieille SA IKE, et la charge utile Supprime pour la détruire elle-même DOIT être la dernière demande envoyée sur la vieille SA IKE.

Les SA devraient être changées de clé de façon volontaire, c'est-à-dire, que la nouvelle SA devrait être établie avant que la vieille n'arrive à expiration et devienne inutilisable. Assez de temps devrait s'écouler entre le moment où la nouvelle SA est établie et celui où la vieille devient inutilisable afin que le trafic puisse être basculé sur la nouvelle SA.

Une différence entre IKEv1 et IKEv2 est que dans IKEv1 la durée de vie des SA était négociée. Dans IKEv2, chaque extrémité de la SA est responsable de l'application de sa propre politique de durée de vie à la SA et du changement de clé de la SA quand nécessaire. Si les deux extrémités ont des politiques de durée de vie différentes, l'extrémité avec la plus courte durée de vie va finir par se trouver être celle qui demande toujours le changement de clé. Si une SA a été inactive pendant longtemps et si un point d'extrémité ne veut pas initier la SA en l'absence de trafic, le point d'extrémité PEUT choisir de clore la SA au lieu d'en changer les clés lorsque la durée de vie arrive à expiration. Il peut aussi faire ainsi si il n'y a pas eu de trafic depuis le dernier changement de clé de la SA.

Noter que IKEv2 permet délibérément des SA parallèles avec les mêmes sélecteurs de trafic entre des points d'extrémité communs. Un de ses objets est pour prendre en charge les différences de qualité de service du trafic entre les SA (voir les [RFC2474], [RFC2475], et le paragraphe 4.1 de la [RFC2983]). 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, de sorte que l'heuristique de changement de clé de IKEv1 consistant à supprimer les SA sur la base de sélecteurs de trafic dupliqués NE DEVRAIT PAS être utilisée.

Il y a des fenêtres de temps -- en particulier en présence de paquets perdus -- où les points d'extrémité ne peuvent pas s'accorder sur l'état d'une SA. Le répondant à une CREATE_CHILD_SA DOIT être prêt à accepter des messages sur une SA avant d'envoyer sa réponse à la demande de création, de sorte qu'il n'y a pas d'ambiguïté pour l'initiateur. L'initiateur PEUT commencer à envoyer sur une SA aussitôt qu'il traite la réponse. L'initiateur, cependant, ne peut pas recevoir sur une SA nouvellement créée jusqu'à ce qu'il reçoive et traite la réponse à sa demande CREATE_CHILD_SA. Comment alors le répondant va t-il savoir quand il est à même d'envoyer 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, cependant, il peut en résulter que des paquets vont être éliminés sans nécessité, de sorte qu'une mise en œuvre PEUT différer de tels envois.

Le répondant peut être assuré que l'initiateur est prêt à recevoir des messages sur une SA si soit (1) il a reçu un message cryptographiquement valide sur l'autre moitié de la paire de SA, soit (2) la nouvelle SA change les clés d'une SA existante et il reçoit une demande IKE de clore la SA remplacée. Quand on change les clés d'une SA, le répondant continue d'envoyer du trafic sur la vieille SA jusqu'à ce qu'un de ces événements se produise. Quand on établit une nouvelle SA, le répondant PEUT différer d'envoyer des messages sur une nouvelle SA jusqu'à ce qu'il en reçoive un, ou qu'une fin de temporisation se soit produite. Si un initiateur reçoit un message sur une SA pour laquelle il n'a pas reçu une réponse à sa demande

CREATE_CHILD_SA, il interprète cela comme une perte probable de paquet et retransmet la demande CREATE_CHILD_SA. Un initiateur PEUT envoyer un message ESP factice sur une SA ESP nouvellement créée si il n'a pas de messages en file d'attente afin d'assurer au répondant que l'initiateur est prêt à recevoir des messages.

2.8.1 Changement simultané de clé de SA fille

Si les deux extrémités ont la même politique de durée de vie, il est possible que toutes deux initient un changement de clés au même moment (ce qui va résulter en des SA redondantes). Pour réduire la probabilité que cela se produise, la programmation des demandes de changement de clés DEVRAIT être soumis à une gigue (retardée d'une durée aléatoire après que le besoin de changement de clés est remarqué).

Cette forme de changement de clés peut temporairement résulter en plusieurs SA similaires entre les mêmes paires de nœuds. Quand il y a deux SA éligibles pour recevoir des paquets, un nœud DOIT accepter les paquets entrants sur l'une ou l'autre SA. Si des SA redondantes sont créées à travers une telle collision, la SA créée avec le plus bas des quatre noms occasionnels utilisés dans les deux échanges DEVRAIT être close par le point d'extrémité qui l'a créée. "Plus bas" signifie une comparaison octet par octet (au lieu de, par exemple, comparer les noms occasionnels comme de grands entiers). En d'autres termes, on commence par comparer le premier octet; si ils sont égaux, on passe à l'octet suivant, et ainsi de suite. Si on atteint la fin d'un nom occasionnel, il est le plus bas. Le nœud qui a initié la SA changée de clés survivante devrait supprimer la SA remplacée après l'établissement de la nouvelle.

Voici une explication de l'impact que cela a sur les mises en œuvre. Supposons que les deux hôtes A et B aient une paire de SA filles existantes avec les SPI (SPIa1, SPIb1) et que tous deux commencent le changement de clés en même temps :

Hôte A **Hôte B**

envoi req1 : N(REKEY_SA,SPIa1), SA(..,SPIa2,..),Ni1,.. -->

<-- envoi req2 : N(REKEY_SA,SPIb1), SA(..,SPIb2,..),Ni2

reçoit req2 <--

À ce point, A sait qu'un changement de clé simultané se produit. Cependant, il ne peut pas encore savoir lequel des échanges va avoir le plus bas nom occasionnel, de sorte qu'il va juste noter la situation et répondre comme d'habitude :

envoi rép2 : SA(..,SPIa3,..), Nr1,.. -->

--> reçoit req1

Maintenant B sait aussi qu'un changement de clés simultané est en cours. Il répond comme d'habitude :

<-- envoie rép1: SA(..,SPIb3,..), Nr2,..

reçoit rép1 <--

--> reçoit rép2

À ce point, il y a trois paires de SA filles entre A et B (la vieille et deux nouvelles). A et B peuvent maintenant comparer les noms occasionnels. Supposons que le plus bas était Nr1 dans le message rép2 ; dans ce cas, B (envoyeur de req2) supprime la nouvelle SA redondante, et A (nœud qui a initié la SA survivante) supprime la vieille.

envoi req3 : D(SPIa1) -->

<-- envoi req4 : D(SPIb2)

--> reçoit req3

<-- envoi rép3 : D(SPIb1)

reçoit req4 <--

envoi rép4 : D(SPIa3) -->

Le changement de clés est maintenant terminé.

Cependant, il y a une seconde séquence d'événements possible qui peut arriver si des paquets sont perdus dans le réseau, résultant en retransmissions. Le changement de clés commence comme d'habitude, mais le premier paquet de A (req1) est perdu.

Hôte A **Hôte B**

envoi req1 : N(REKEY_SA,SPIa1), SA(..,SPIa2,..), Ni1,.. --> (perdu)

<-- envoi req2 : N(REKEY_SA,SPIb1), SA(..,SPIb2,..),Ni2

reçoit req2 <--

envoi rép2 : SA(..,SPIa3,..), Nr1,.. -->

--> reçoit rép2

<-- envoi req3 : D(SPIb1)

reçoit req3 <--

envoi rép3 : D(SPIa1) -->

--> reçoit rép3

Du point de vue de B, le changement de clés est maintenant achevé, et comme il n'a pas encore reçu la req1 de A, il ne sait même pas qu'il y a eu un changement de clés simultané. Cependant, A va continuer de retransmettre le message, et finalement, il va atteindre B :

renvoi req1 -->

--> reçoit req1

Pour B, il semble que A essaye de changer les clés d'une SA qui n'existe plus ; donc, B répond à la demande avec quelque chose de non fatal comme CHILD_SA_NOT_FOUND :

<-- envoi rép1 : N(CHILD_SA_NOT_FOUND)

reçoit rép1 <--

Quand A reçoit cette erreur, il sait déjà qu'il y a eu un changement de clés simultané, de sorte qu'il peut ignorer le message.

2.8.2 Changement simultané de clé de SA IKE

Le cas probablement le plus complexe se produit quand les deux homologues essayent de changer les clés de la SA IKE au même moment. Fondamentalement, le texte du paragraphe 2.8 s'applique aussi à ce cas ; cependant, il est important de s'assurer que les SA filles sont héritées par la bonne SA IKE.

Le cas où les deux points d'extrémité remarquent le changement de clés simultané fonctionne de la même façon qu'avec les SA filles. Après les échanges CREATE_CHILD_SA, trois SA IKE existent entre A et B : la vieille SA IKE et deux nouvelles SA IKE. La nouvelle SA IKE qui contient le plus bas nom occasionnel DEVRAIT être supprimée par le nœud qui l'a créée, et l'autre nouvelle SA IKE survivante DOIT hériter de toutes les SA filles.

En plus des cas normaux de changement de clés simultanés, il est un cas particulier où un homologue finit son changement de clés avant même qu'il remarque que l'autre homologue est en train de faire un changement de clé. Si un seul homologue détecte un changement de clés simultané, les SA redondantes ne sont pas créées. Dans ce cas, quand l'homologue qui n'a pas remarqué le changement de clés simultané reçoit la demande de changement de clés de la SA IKE qu'il vient de réussir, il DEVRAIT retourner TEMPORARY_FAILURE *(défaillance temporaire)* parce que il est une SA IKE qui est actuellement en train d'essayer de clore (qu'il ait ou non déjà envoyé la notification de suppression pour la SA). Si l'homologue qui a remarqué le changement de clés simultané reçoit la demande de suppression de l'autre homologue pour la vieille SA IKE, il sait que l'autre homologue n'a pas détecté le changement de clés simultané, et le premier homologue peut oublier sa propre tentative de changement de clés.

Hôte A	**Hôte B**
envoi req1 : SA(..,SPIa1,..),Ni1,.. -->	
	<-- envoi req2 : SA(..,SPIb1,..),Ni2,..
	--> reçoit req1
	<-- envoi rép1 : SA(..,SPIb2,..),Nr2,..
reçoit rép1 <--	
envoi req3 : D() -->	
	--> reçoit req3

À ce point, l'hôte B voit une demande de clore la IKE_SA. Il n'y a rien de plus à faire que de répondre comme d'habitude. Cependant, à ce point l'hôte B devrait arrêter de retransmettre req2, car une fois que l'hôte A a reçu rép3, il va supprimer tous les états associés à la vieille IKE_SA et ne sera pas capable d'y répondre.

<-- envoi rép3 : ()

La notification TEMPORARY_FAILURE n'était pas incluse dans la RFC 4306, et la prise en charge de la notification TEMPORARY_FAILURE n'est pas négociée. Donc, les homologues plus anciens qui mettent en œuvre la RFC 4306 mais pas le présent document peuvent recevoir ces notifications. Dans ce cas, il vont la traiter de la même façon que toute autre notification d'erreur inconnue, et vont arrêter l'échange. Comme l'autre homologue a déjà changé les clés de l'échange, faire ainsi n'a aucun effet négatif.

2.8.3 Changement de clé de SA IKE contre ré authentification

Changer les clés de la SA IKE et ré authentification sont des concepts différents dans IKEv2. Le changement des clés de la SA IKE établit de nouvelles clés pour la SA IKE et remet à zéro les compteurs d'identifiant de message, mais il n'authentifie pas à nouveau les parties (aucune charge utile AUTH ou EAP n'est impliquée).

Bien que changer les clés de la SA IKE puisse être important dans certains environnements, la ré authentification (la vérification que les parties ont encore accès aux accréditifs à long terme) est souvent plus importante.

IKEv2 n'a aucun mécanisme spécial pour prendre en charge la ré authentification. Elle est faite en créant une nouvelle SA IKE à partir de rien (en utilisant les échanges IKE_SA_INIT/IKE_AUTH, sans aucune charge utile Notify REKEY_SA) créant de

nouvelles SA filles au sein de la nouvelle SA IKE (sans charge utile Notify REKEY_SA) et finalement en supprimant la vieille SA IKE (ce qui supprime aussi les vieilles SA filles).

Cela signifie que la ré authentification établit aussi de nouvelles clés pour la SA IKE et ses SA filles. Donc, alors que le changement de clés peut être effectué plus souvent que la ré authentification, la situation où la "durée de vie d'authentification" est plus courte que la "durée de vie de clé" n'a pas de sens.

Alors que la création d'une nouvelle SA IKE peut être initiée par l'une ou l'autre partie (initiateur ou répondant dans la SA IKE originale) l'utilisation de EAP et/ou des charges utiles Configuration signifie en pratique que la ré authentification doit être initiée par la même partie que la SA IKE originale. IKEv2 ne permet actuellement pas au répondant de demander la ré authentification dans ce cas ; cependant, il y a des extensions qui ajoutent cette fonctionnalité, comme dans la [RFC4478].

2.9 Négociation de sélecteurs de trafic

Quand un sous système IPsec conforme à la RFC4301 reçoit un paquet IP qui correspond à un sélecteur de "protection" dans sa base de données de politique de sécurité (SPD, *Security Policy Database*) le sous système protège ce paquet avec IPsec. Quand il n'existe pas encore de SA, c'est la tâche de IKE de la créer. La maintenance de la SPD d'un système sort du domaine d'application de IKE, 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 le paragraphe 1.1.3).

Les charges utiles Sélecteur de trafic (TS, *Trafic Selector*) permettent aux points d'extrémité de communiquer certaines des informations provenant de leur SPD à leurs homologues. Elles doivent être communiquées à IKE par la SPD (par exemple, l'API PF_KEY [RFC2367] utilise le message SADB_ACQUIRE). Les charges utiles TS spécifient les critères de sélection pour les paquets qui vont être transmis sur la SA nouvellement établie.

Cela peut servir de vérification de cohérence dans certains scénarios pour assurer que les SPD sont cohérents. Dans d'autres, cela guide la mise à jour dynamique de la SPD.

Deux charges utiles TS apparaissent dans chacun des messages de l'échange qui crée une paire de SA filles. Chaque charge utile 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.

La première des deux charges utiles TS est appelée TSi (sélecteur de trafic initiateur). 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 SA filles. TSr spécifie l'adresse de destination du trafic transmis au (ou l'adresse de source du trafic transmis du) répondant de la paire de SA filles. Par exemple, si l'initiateur original demande la création d'une paire de SA filles, et souhaite tunneler tout le trafic à partir du sous réseau 198.51.100.* sur le côté de l'initiateur au sous réseau 192.0.2.* sur le côté du répondant, l'initiateur va inclure un seul sélecteur de trafic dans chaque charge utile TS. TSi spécifierait la gamme d'adresses (198.51.100.0 - 198.51.100.255) et TSr spécifierait la gamme d"adresses (192.0.2.0 - 192.0.2.255). En supposant que la proposition était acceptable au répondant, il renverrait des charges utiles TS identiques.

IKEv2 permet au répondant de choisir un sous ensemble du trafic proposé par l'initiateur. Cela pourrait arriver quand les configurations des deux points d'extrémité sont en train d'être mises à jour mais que seulement une extrémité a reçu la nouvelle information. Comme les deux points d'extrémité peuvent être configurés par des personnes différentes, une incompatibilité peut persister pendant une assez longue durée même en l'absence d'erreur. Il 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 à jour.

Quand le répondant choisit un sous ensemble du trafic proposé par l'initiateur, il rétrécit les sélecteurs de trafic à un sous ensemble de la proposition de l'initiateur (pourvu que l'ensemble ne devienne pas nul). Si le type de sélecteur de trafic proposé est inconnu, le répondant ignore ce sélecteur de trafic, de sorte que le type inconnu n'est pas retourné dans l'ensemble rétréci.

Pour permettre au répondant de choisir la gamme approprié dans ce cas, si l'initiateur a demandé la SA à cause d'un paquet de données, l'initiateur DEVRAIT inclure le premier sélecteur de trafic dans chacun de TSi et TSr comme un sélecteur de trafic très spécifique incluant les adresses dans le paquet qui déclenche la demande. Dans l'exemple, l'initiateur inclurait dans TSi deux sélecteurs de trafic : le premier contenant la gamme d'adresses (198.51.100.43 - 198.51.100.43) l'accès de source et le protocole IP provenant du paquet et le second contenant (198.51.100.0 - 198.51.100.255) avec tous les accès et protocoles IP. L'initiateur inclurait de façon similaire deux sélecteurs de trafic dans TSr. Si l'initiateur crée la paire de SA filles non en réponse d'un paquet entrant, mais plutôt, disons, au démarrage, il ne peut alors pas y avoir d'adresses spécifiques que préfère l'initiateur pour le tunnel initial par rapport à tout autre. Dans ce cas, les premières valeurs dans TSi et TSr peuvent être des gammes plutôt que des valeurs spécifiques.

Le répondant effectue le rétrécissement comme suit :

- Si la politique du répondant ne lui permet d'accepter aucune partie des sélecteurs de trafic proposés, il répond avec un message Notify TS_UNACCEPTABLE.

- Si la politique du répondant permet l'ensemble entier du trafic couvert par TSi et TSr, aucun rétrécissement n'est nécessaire, et le répondant peut retourner les mêmes valeurs de TSi et TSr.

- Si la politique du répondant lui permet d'accepter le premier sélecteur de TSi et TSr, le répondant DOIT alors rétrécir les sélecteurs de trafic à un sous ensemble qui inclut les premiers choix de l'initiateur. Dans l'exemple ci dessus, le répondant pourrait répondre avec TSi à (198.51.100.43 - 198.51.100.43) avec tous les accès et protocoles IP.

- Si la politique du répondant ne lui permet pas d'accepter le premier sélecteur de TSi et TSr, le répondant rétrécit à un sous ensemble acceptable de TSi et TSr.

Lorsque le rétrécissement est fait, il peut y avoir plusieurs sous ensembles qui sont acceptables mais leur union ne l'est pas. Dans ce cas, le répondant en choisit arbitrairement un, et PEUT inclure une notification ADDITIONAL_TS_POSSIBLE dans la réponse. La notification ADDITIONAL_TS_POSSIBLE affirme que le répondant a rétréci les sélecteurs de trafic proposés mais que les autres sélecteurs de trafic auraient aussi été acceptables, bien que seulement dans une SA séparée. Il n'y a pas de données associées à ce type Notify. Ce cas ne va se produire que quand l'initiateur et le répondant ont des configurations différentes. Si l'initiateur et le répondant s'accordent sur la granularité des tunnels, l'initiateur ne va jamais demander un tunnel plus large que ce que le répondant va accepter.

Il est possible à la politique du répondant de contenir plusieurs plus petites gammes, toutes englobées par le sélecteur de trafic de l'initiateur, et la politique du répondant étant que chacune de ces gammes devrait être envoyée sur une SA différente. En continuant avec l'exemple ci-dessus, le répondant pourrait avoir une politique de vouloir tunneler ces adresses de et vers l'initiateur, mais pourrait exiger que chaque paire d'adresse soit sur une SA fille négociée séparément. Si l'initiateur n'a pas généré sa demande sur la base du paquet, mais (par exemple) au démarrage, il n'y aura pas les très spécifiques premiers sélecteurs de trafic pour aider le répondant à choisir la gamme correcte. Il n'y aurait alors aucun moyen pour le répondant de déterminer quelle paire d'adresses devrait être incluse dans ce tunnel, et il devrait deviner ou alors rejeter la demande avec un message Notify de SINGLE_PAIR_REQUIRED *(une seule paire exigée)*.

L'erreur SINGLE_PAIR_REQUIRED indique qu'une demande CREATE_CHILD_SA est inacceptable parce que son envoyeur veut seulement accepter les sélecteurs de trafic qui spécifient une seule paire d'adresses. Le demandeur est supposé répondre en demandant une SA pour le seul trafic spécifique qu'il essaye de transmettre.

Peu de mises en œuvre vont avoir des politiques qui exigent des SA séparées pour chaque paire d'adresse. À cause de cela, si seules des parties de TSi et TSr proposés par l'initiateur sont acceptables pour le répondant, les répondants DEVRAIENT rétrécir les sélecteurs à un sous ensemble acceptable plutôt que d'utiliser SINGLE_PAIR_REQUIRED.

2.9.1 Sélecteurs de trafic qui violent leur propre politique

Quand il crée une nouvelle SA, l'initiateur doit éviter de proposer des sélecteurs de trafic qui violent sa propre politique. Si cette règle n'est pas suivie, le trafic valide peut être abandonné. Si on utilise des politiques décorrélées de la [RFC4301], cette sorte de violations de politique ne peut pas se produire.

Ceci est mieux illustré par un exemple. Supposons que l'hôte A ait une politique dont l'effet est que le trafic pour 198.51.100.66 est envoyé via l'hôte B, chiffré avec AES, et le trafic pour tous les autres hôtes dans 198.51.100.0/24 est aussi envoyé via B, mais doit utiliser 3DES. Supposons aussi que l'hôte B accepte toutes les combinaisons de AES et 3DES.

Si l'hôte A propose une SA qui utilise 3DES, et inclut un TSr contenant (198.51.100.0 - 198.51.100.255) cela sera accepté par l'hôte B. Maintenant, l'hôte B peut aussi utiliser cette SA pour envoyer du trafic de 198.51.100.66, mais ces paquets vont être éliminés par A car il exige l'utilisation de AES pour ce trafic. Même si l'hôte A crée une nouvelle SA seulement pour 198.51.100.66 qui utilise AES, l'hôte B peut librement continuer d'utiliser la première SA pour le trafic. Dans cette situation, quand il propose la SA, l'hôte A devrait avoir suivi sa propre politique, et inclure un TSr contenant à la place ((198.51.100.0 - 198.51.100.65), (198.51.100.67 - 198.51.100.255)).

En général, si (1) l'initiateur fait une proposition "pour le trafic (TSi/TSr), faire une SA", et (2) pour un sous ensemble X' de X, l'initiateur n'accepte en fait pas le trafic X' avec SA, et (3) l'initiateur va vouloir accepter le trafic X' avec SA' (!=SA), le trafic valide peut être inutilement éliminé car le répondant peut appliquer soit SA, soit SA' au trafic X'.

2.9.2 Sélecteurs de trafic dans le changement de clés

Le changement de clés est utilisé pour remplacer une SA fille existante par une autre. Si la nouvelle SA est admise à avoir un ensemble de sélecteurs plus étroit que l'originale, le trafic qui était permis sur la vieille SA va être éliminé dans la nouvelle SA, violant donc l'idée de "remplacement". Donc, la nouvelle SA NE DOIT PAS avoir des sélecteurs plus étroits que l'originale. Si la nouvelle SA se trouve avoir besoin d'une portée plus étroite que celle de la SA actuellement utilisée, cela voudrait dire que la politique a été changée d'une façon telle que la SA actuellement utilisée est contraire à la politique. Dans ce cas, la SA devrait avoir été supprimée après que le changement de politique est entré en vigueur.

Quand l'initiateur tente de changer les clés de la SA fille, les sélecteurs de trafic proposés DEVRAIENT être les mêmes, ou un sur ensemble, que les sélecteurs de trafic utilisés dans la vieille SA fille. C'est-à-dire que ils devraient être les mêmes, ou un sur ensemble de la politique actuellement active (décorrélée). Le répondant NE DOIT PAS rétrécir la portée des sélecteurs de trafic au delà de ce qui est actuellement utilisé.

Parce que une SA dont on a changé les clés ne peut jamais avoir une portée plus étroite que celle actuellement utilisée, il n'est pas besoin de sélecteurs à partir du paquet, de sorte que ces sélecteurs NE DEVRAIENT PAS être envoyés.

2.10 Noms occasionnels

Chaque message IKE_SA_INIT contient 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 clé utilisée pour obtenir des clés pour la SA fille, et pour assurer la création des bits pseudo aléatoires forts de la clé Diffie-Hellman. Les noms occasionnels utilisés dans IKEv2 DOIVENT être choisis au hasard, ils DOIVENT faire au moins 128 bits, et DOIVENT être au moins de la moitié de la taille de clé de la fonction pseudo aléatoire (PRF, *pseudorandom function*) négociée. Cependant, l'initiateur choisit le nom occasionnel avant que le résultat de la négociation soit connu. À cause de cela, le nom occasionnel doit être assez long pour toutes les PRF proposées. Si la même source de nombre aléatoire est utilisée pour les clés et les noms occasionnels, il faut prendre soin de s'assurer que ce dernier usage ne compromet pas le premier.

2.11 Agilité d'adresse et d'accès

IKE fonctionne sur les accès UDP 500 et 4500, et établit implicitement des associations ESP et AH pour les adresses IP sur lesquelles il fonctionne. Les adresses et accès IP dans l'en-tête externe ne sont pas, cependant, eux-mêmes protégés cryptographiquement, et IKE est conçu pour fonctionner même à travers des boîtiers 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 accès d'où la demande a été reçue. Elle DOIT spécifier l'adresse et l'accès auxquels la demande a été reçue comme l'adresse et accès de source dans la réponse. IKE fonctionne de façon identique sur IPv4 ou IPv6.

2.12 Réutilisation des exponentielles Diffie-Hellman

IKE génère du matériel de chiffrement en utilisant un échange éphémère Diffie-Hellman afin d'obtenir la propriété de "secret parfait vers l'avant". Cela signifie qu'une fois une connexion close et ses clés correspondantes oubliées, même quelqu'un qui a enregistré toutes les données provenant de la connexion et a accès à toutes les clés à long terme des deux points d'extrémité ne peut pas reconstruire les clés utilisées pour protéger la conversation sans faire une recherche en force brute de l'espace de clés de session.

Réaliser le secret parfait vers l'avant exige que quand une connexion est close, chaque point d'extrémité DOIT oublier non seulement les clés utilisées par la connexion mais aussi toutes les informations qui pourraient être utilisées pour recalculer ces clés.

Parce que les exponentielles Diffie-Hellman sont coûteuses 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 ce faire. Un point d'extrémité pourrait ne choisir une nouvelle exponentielle que périodiquement bien qu'il puisse en résulter un secret vers l'avant moins que parfait si une connexion dure moins longtemps que la durée de vie de l'exponentielle. Ou il pourrait garder trace de l'exponentielle utilisée pour chaque connexion et ne supprimer les informations associées à l'exponentielle que quand une connexion correspondante est close. Cela permettrait que l'exponentielle soit réutilisée sans perdre le secret parfait vers l'avant au prix du maintien de plus d'état.

La décision de si et quand réutiliser les exponentielles Diffie-Hellman est privée au sens où elle n'affecte pas l'interopérabilité. Une mise en œuvre qui réutilise les exponentielles PEUT choisir de se souvenir de l'exponentielle utilisée par l'autre point

d'extrémité sur les échanges passés et si l'une est réutilisée, pour éviter la seconde moitié du calcul. Voir dans [REUSE] et la [RFC6989] une analyse de la sécurité de cette pratique et des considérations de sécurité supplémentaires quand on réutilise des clés éphémères Diffie-Hellman.

2.13 Génération du matériel de chiffrement

Dans le contexte de la SA IKE, quatre algorithmes cryptographiques sont négociés : un algorithme de chiffrement, un algorithme de protection de l'intégrité, un groupe Diffie-Hellman, et une fonction pseudo aléatoire (PRF). La PRF est utilisée pour la construction de matériaux de chiffrement pour tous les algorithmes cryptographiques utilisés dans la SA IKE comme dans les SA filles.

On suppose que chaque algorithme de chiffrement et de protection de l'intégrité utilise une clé de taille fixe et que toute valeur choisie au hasard de cette taille fixe peut servir de clé appropriée. Pour les algorithmes qui acceptent une clé de longueur variable, une taille de clé fixe DOIT être spécifiée au titre de la transformation cryptographique négociée (voir au paragraphe 3.3.5 la définition de l'attribut Transformation de longueur de clé). 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 de l'intégrité fondées sur un code d'authentification de message par hachage de clé (HMAC, *Hashed Message Authentication Code*) la taille de clé fixe est la taille du résultat de la fonction de hachage sous-jacente.

Il est supposé que les PRF acceptent des clés de toutes longueurs, mais ont une taille de clé préférée. La taille de clé préférée DOIT être utilisée comme la longueur de SK_d, SK_pi, et SK_pr (voir au paragraphe 2.14). Pour les PRF fondées sur une construction HMAC, la taille de clé préférée est égale à la longueur du résultat de la fonction de hachage sous-jacente. Les autres types de PRF DOIVENT spécifier leur taille de clé préférée.

Le matériel de chiffrement va toujours être déduit comme résultat de l'algorithme de PRF négocié. Comme la quantité de matériel de chiffrement nécessaire peut être supérieure à la taille du résultat de la PRF, celle-ci est utilisée itérativement Le terme "prf+" décrit une fonction qui sort un flux pseudo aléatoire sur la base des entrées à une fonction pseudo aléatoire appelée "prf".

Dans ce qui suit, | indique l'enchaînement. prf+ est défini comme : 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)
...

Cela se continue jusqu'à ce que tout le matériel nécessaire pour calculer toutes les clés requises aient été sorti de prf+. Les clés sont prises de la chaîne de résultat sans égard aux frontières (par exemple, si les clés requises sont une clé AES de 256 bits et une clé HMAC de 160 bits, et si la fonction prf génère 160 bits, la clé AES viendra de T1 et du commencement de T2, tandis que la clé HMAC va venir du reste de T2 et du commencement de T3).

La constante enchaînée à la fin de chaque fonction prf est d'un seul octet. La fonction prf+ n'est pas définie au delà de 255 fois la taille du résultat de la fonction prf.

2.14 Génération du matériel de chiffrement pour la SA IKE

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 partagé Diffie-Hellman établi durant cet échange. SKEYSEED est utilisé pour calculer sept autres secrets : SK_d utilisé pour déduire de nouvelles clés pour les SA filles établies avec cette SA IKE, SK_ai et SK_ar utilisés comme clés pour l'algorithme de protection de l'intégrité pour authentifier les messages composants des échanges suivants, SK_ei et SK_er utilisés pour chiffrer (et bien sûr déchiffrer) tous les échanges suivants, et SK_pi et SK_pr, qui sont utilisés quand on génère une charge utile AUTH. Les longueurs de SK_d, SK_pi, et SK_pr DOIVENT être de la longueur de clé préférée de la PRF qui a fait l'objet de l'accord.

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)

(indiquant que les quantités SK_d, SK_ai, SK_ar, SK_ei, SK_er, SK_pi, et SK_pr sont prises dans l'ordre à partir des bits générés de la prf+) g^ir est le secret partagé provenant de l'échange Diffie-Hellman éphémère. g^ir est représenté comme une chaîne d'octets en ordre gros boutien bourré de zéros si nécessaire pour le faire de la longueur du module. Ni et Nr sont les noms occasionnels, dépouillés de tout en-tête. Pour des raisons historiques de rétro compatibilité, il y a deux PRF qui ont un traitement spécial dans ce calcul. Si la PRF négociée est AES-XCBC-PRF-128 [RFC4434] ou AES-CMAC-PRF-128 [RFC4615], seuls les 64 premiers bits de Ni et les 64 premiers bits de Nr sont utilisés pour le calcul de SKEYSEED, mais tous les bits sont utilisés pour l'entrée à la fonction prf+.

Les deux directions du flux de trafic utilisent des clés différentes. Les clés utilisées pour protéger les messages provenant de l'initiateur original sont SK_ai et SK_ei. Les clés utilisées pour protéger les messages dans l'autre direction sont SK_ar et SK_er.

2.15 Authentification de la SA IKE

Quand on n'utilise pas l'authentification extensible (voir le paragraphe 2.16) les homologues sont authentifiés en ayant chacun signé (ou avec un MAC en utilisant un secret partagé avec bourrage comme clé, comme décrit plus loin dans ce paragraphe) un bloc de données. Dans ces calculs, IDi' et IDr' sont les charges utiles Identifiant entières en excluant l'en-tête fixe. Pour le répondant, les octets à signer commencent avec le premier octet du premier SPI dans l'en-tête du second message (réponse IKE_SA_INIT) et se terminent avec le dernier octet de la dernière charge utile dans le second message. Ajouté à cela (pour les besoins du calcul de la signature) sont le nom occasionnel Ni de l'initiateur (juste la valeur, pas la charge utile qui la contient) et la valeur prf(SK_pr, IDr'). Noter que ni le nom occasionnel Ni ni la valeur prf(SK_pr, IDr') ne sont transmis. De même, l'initiateur signe le premier message (demande IKE_SA_INIT) en commençant par le premier octet du premier SPI dans l'en-tête et se terminant par le dernier octet de la dernière charge utile. Ajouté à cela (pour les besoins du calcul de la signature) sont le nom occasionnel Nr du répondant, et la valeur prf(SK_pi, IDi'). Il est critique pour la sécurité de l'échange que chaque côté signe le nom occasionnel de l'autre côté.

Les octets signés de l'initiateur peuvent être décrits comme suit :

OctetsSignésInitiateur = MessageRéel1 | DonnéesNonceR | MACedIDPourI
En-TêteGénéralIKE = [quatre octets 0 si on utilise l'accès 4500] | En-TêteRéelIKE
En-TêteRéelIKE = SPIi | SPIr | . . . | Longueur
MessageRéel1 = En-TêteRéelIKE | ResteDuMessage1
ChargeUtileNonceR = En-TêteChargeUtile | DonnéesNonceR
ChargeUtileIDInitiateur = En-TêteChargeUtile | ResteDeChargeUtileIDInitiateur
ResteDeChargeUtileIDInitiateur = IDType | RÉSERVÉ | DonnéesIDInitiateur
MACedIDPourI = prf(SK_pi, ResteDeChargeUtileIDInitiateur)

Les octets signés du répondant peuvent être décrits comme suit :

OctetsSignésRépondant = MessageRéel2 | DonnéesNonceI | MACedIDPourR
En-TêteGénéralIKE = [quatre octets 0 si on utilise l'accès 4500] | En-TêteRéelIKE
En-TêteRéelIKE = SPIi | SPIr | . . . | Longueur
MessageRéel2 = En-TêteRéelIKE | ResteDuMessage2
ChargeUtileNonceR = En-TêteChargeUtile | DonnéesNonceI
ChargeUtileIDRépondant = En-TêteChargeUtile | ResteDeChargeUtileIDRépondant
ResteDeChargeUtileIDRépondant = IDType | RÉSERVÉ | DonnéesIDRépondant
MACedIDPourR = prf(SK_pr, ResteDeChargeUtileIDRépondant)

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

Facultativement, les messages 3 et 4 PEUVENT inclure un certificat, ou une chaîne de certificats fournissant la preuve que la clé utilisée pour calculer une signature numérique appartient au nom qui est dans la charge utile Identifiant. La signature ou le MAC vont être calculés en utilisant les algorithmes imposés par le type de clé utilisée par le signataire et spécifiée par le champ Méthode d'authentification dans la charge utile Authentification. Il n'est pas exigé que l'initiateur et le répondant signent avec le même algorithme de chiffrement. Le choix des algorithmes de chiffrement dépend du type de clé que chacun a. En particulier, l'initiateur peut utiliser une clé partagée alors que le répondant peut avoir une clé de signature publique et un certificat. Il va souvent se trouver (mais ce n'est pas exigé) que, si un secret partagé est utilisé pour l'authentification, la même clé est utilisée dans les deux directions.

Noter qu'il est de pratique courante, mais typiquement non sûre, d'avoir une clé partagée déduite seulement d'un mot de passe choisi par l'utilisateur sans incorporer d'autre source d'aléa. Ceci est typiquement non sûr parce que les mots de passe choisis par l'utilisateur ont très probablement une imprévisibilité insuffisante pour résister aux attaques de dictionnaire et ces attaques ne sont pas empêchées dans cette méthode d'authentification. (Les applications qui utilisent l'authentification fondée sur le mot de passe pour l'amorçage et une SA IKE 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 a besoin de contenir autant d'imprévisibilité que la plus forte clé qui est négociée. Dans le cas d'une clé pré partagée, la valeur AUTH est calculée par :

Pour l'initiateur :
AUTH = prf(prf(Secret partagé, "Bourrage de clé IKEv2"), <OctetsSignésInitiateur>)

Pour le répondant :
AUTH = prf(prf(Secret partagé, "Bourrage de clé IKEv2"), <OctetsSignésRépondant>)

où la chaîne "Bourrage de clé pour IKEv2" est 17 caractères ASCII sans terminaison nulle. Le secret partagé peut être de longueur variable. La chaîne de bourrage est ajoutée afin 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(Secret partagé, "Bourrage de clé IKEv2") qui ne pourrait pas être utilisée comme mot de passe équivalent pour des protocoles autres que IKEv2. Comme on l'a noté auparavant, déduire le secret partagé d'un mot de passe n'est pas sûr. Cette construction est utilisée parce que on prévoit que les gens le feront de toutes façons. L'interface de gestion par laquelle le secret partagé est fourni 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 hexadécimal du secret partagé. L'interface de gestion PEUT accepter d'autres codages si l'algorithme pour traduire le codage en chaîne binaire est spécifié.

Il y a deux types d'authentification EAP (décrits au paragraphe 2.16) et chaque type utilise des valeurs différentes dans le calcul de AUTH montré ci-dessus. Si la méthode EAP est génératrice de clés, substituer une clé de session maîtresse (MSK, *master session key*) au secret partagé dans le calcul. Pour les méthodes non génératrices de clés, substituer SK_pi et SK_pr, respectivement, au secret partagé dans les deux calculs de AUTH.

2.16 Méthodes d'extension du protocole d'authentification

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

Bien que le présent document fasse référence à la [RFC3748] avec l'intention que de nouvelles méthodes puissent être ajoutées à l'avenir sans mettre à jour la présente spécification, des variantes plus simples sont documentées ici. La [RFC3748] définit un protocole d'authentification qui exige un nombre variable de messages. L'authentification extensible est mise en œuvre dans IKE comme des échanges IKE_AUTH supplémentaires qui DOIVENT être réalisés afin d'initialiser la SA IKE.

Un initiateur indique son désir d'utiliser EAP en n'incluant pas la charge utile AUTH dans le premier message dans l'échange IKE_AUTH. (Noter que la charge utile AUTH est exigée pour l'authentification non EAP, et n'est donc pas marquée comme facultative dans la suite du présent document.) En incluant une charge utile IDi mais pas une charge utile AUTH, l'initiateur a déclaré une identité mais ne l'a pas prouvée. Si le répondant veut utiliser une méthode EAP, il va placer une charge utile Protocole d'authentification extensible (EAP) dans la réponse de l'échange IKE_AUTH et différer l'envoi de SAr2, TSi, et TSr jusqu'à ce que l'authentification de l'initiateur soit achevée dans un échange IKE_AUTH suivant. Dans le cas d'une méthode EAP minimale, l'établissement initial de la 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}

Comme décrit au paragraphe 2.2, quand EAP est utilisé, chaque paire de messages d'établissement initial de SA IKE va avoir ses numéros de message incrémentés ; la première paire de messages IKE_AUTH va avoir un identifiant de 1, la seconde aura 2, et ainsi de suite.

Pour les méthodes EAP qui créent une clé partagée comme effet collatéral de l'authentification, cette clé partagée DOIT être utilisée par l'initiateur et par le répondant pour générer des charges utiles AUTH dans les messages 7 et 8 en utilisant la syntaxe pour les secrets partagés spécifiée au paragraphe 2.15. La clé partagée provenant de EAP est le champ nommé MSK dans la spécification EAP. Cette clé partagée générée durant un échange IKE NE DOIT PAS être utilisée pour un autre objet.

Les méthodes EAP qui n'établissent pas de 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 n'utilisent pas un tunnel authentifié par le serveur. Voir plus de détails à la Section 5 "Considérations sur la sécurité". 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 SA IKE qui utilise EAP doit être capable d'étendre l'échange initial de protocole à au moins dix échanges IKE_AUTH pour le cas où le répondant enverrait des messages de notification et/ou des réessais de l'invite d'authentification. Une fois que l'échange de protocole défini par la méthode d'authentification EAP choisie s'est terminée avec succès, le répondant DOIT envoyer une charge utile EAP contenant le message Succès. De même, si la méthode d'authentification échoue, le répondant DOIT envoyer une charge utile EAP contenant le message Échec. Le répondant PEUT à tout moment terminer l'échange IKE en envoyant une charge utile EAP contenant le message Échec.

Suite à un tel échange étendu, les charges utiles EAP AUTH DOIVENT être incluses dans les deux messages qui suivent celui qui contient le message EAP Succès.

Quand l'authentification d'initiateur utilise EAP, il est possible que le contenu de la charge utile IDi soit utilisé seulement pour les besoins d'acheminement d'authentification autorisation et comptabilité (AAA, *Authentication, Authorization, and Accounting*) et choisir la méthode EAP à utiliser. Cette valeur peut être différente de l'identité authentifiée par la méthode EAP. Il est important que les recherches de politique et les décisions de contrôle d'accès utilisent l'identité authentifiée réelle. Le serveur EAP est souvent mis en œuvre dans un serveur AAA séparé qui communique avec le répondant IKEv2. Dans ce cas, l'identité authentifiée, si elle est différente de celle de la charge utile IDi, doit être envoyée du serveur AAA au répondant IKEv2.

2.17 Génération des matériaux de chiffrement pour les SA filles

Une seule SA fille est crée par l'échange IKE_AUTH, et des SA filles supplémentaires peuvent facultativement être créées dans un échange CREATE_CHILD_SA. Le matériel de chiffrement 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 SA fille 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 incluant un échange facultatif Diffie-Hellman, le matériel de chiffrement est défini par :

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

Où g^ir (nouveau) est le secret partagé provenant de l'échange éphémère Diffie-Hellman de cet échange CREATE_CHILD_SA (représenté comme une chaîne d'octets en ordre gros boutien bourré de zéros dans les bits de poids fort si nécessaire pour l'aligner sur la longueur du module).

Une seule négociation CREATE_CHILD_SA peut résulter en, plusieurs associations de sécurité. Les SA ESP et AH existent par paires (une dans chaque direction) de sorte que deux SA sont créées pour elles dans une seule négociation de SA fille. De plus, la négociation de SA fille peut inclure de futurs protocoles IPsec en plus de, ou à la place de, ESP ou AH (par exemple, ROHC_INTEG comme décrit dans la [RFC5857]). En tous cas, le matériel de chiffrement pour chaque SA fille DOIT être pris dans le KEYMAT étendu en utilisant les règles suivantes :

- Toutes les clés pour les SA portant des données de l'initiateur au répondant sont prises avant les SA qui vont du répondant à l'initiateur.
- Si plusieurs protocoles IPsec sont négociés, le matériel de chiffrement pour chaque SA fille est pris dans l'ordre dans lequel les en-têtes de protocole vont apparaître dans le paquet encapsulé.
- Si un protocole IPsec exige plusieurs clés, l'ordre dans lequel elles sont prises dans le matériel de chiffrement de la SA doit être décrit dans la spécification du protocole. Pour ESP et AH, la [RFC4301] définit l'ordre, à savoir la clé de chiffrement

(si il y en a une) DOIT être prise dans les premiers bits et la clé d'intégrité (si il y en a une) DOIT être prise dans les bits restants.

Chaque algorithme de chiffrement prend un nombre fixe de bits de matériel de chiffrement spécifié au titre de l'algorithme, ou négocié dans les charges utiles de SA (voir au paragraphe 2.13 la description des longueurs de clé, et au paragraphe 3.3.5 la définition de l'attribut Transformation de longueur de clé).

2.18 Changement de clé des SA IKE avec l'échange CREATE_CHILD_SA

L'échange CREATE_CHILD_SA peut être utilisé pour changer les clés d'une SA IKE existante (voir les paragraphes 1.3.2 et 2.8). Les nouveaux SPI d'initiateur et de répondant sont fournis dans les champs SPI des structures Proposal à l'intérieur des charges utiles de l'association de sécurité (pas des champs SPI dans l'en-tête IKE). Les charges utiles TS sont omises lors d'un changement de clés d'une SA IKE. Le SKEYSEED pour la nouvelle SA IKE est calculé en utilisant le SK_d provenant de la SA IKE existante comme suit : SKEYSEED = prf(SK_d (vieux), g^ir (nouveau) | Ni | Nr)

où g^ir (nouveau) 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 bourré avec des zéros si nécessaire pour faire de sa longueur celle du module) et Ni et Nr sont les deux noms occasionnels supprimés de tous les en-têtes.

La vieille SA IKE et la nouvelle peuvent avoir choisi une PRF différente. Comme l'échange de clés relève de la vieille SA IKE, c'est la PRF de la vieille SA IKE qui est utilisée pour générer la SKEYSEED.

La principale raison du changement de clé de la SA IKE est de s'assurer que la compromission du vieux matériel de clés ne fournit pas des informations sur les clés actuelles, ou vice versa. Donc, les mises en œuvre DOIVENT effectuer un nouvel échange Diffie-Hellman lors du changement de clés de la SA IKE. En d'autres termes, un initiateur NE DOIT PAS proposer la valeur "AUCUNE" pour la transformation Diffie-Hellman, et un répondant NE DOIT PAS accepter une telle proposition. Cela signifie qu'un échange de changement de clés réussi pour la SA IKE inclut toujours les charges utiles KEi/KEr.

La nouvelle SA IKE DOIT remettre ses compteurs de messages à 0.

SK_d, SK_ai, SK_ar, SK_ei, et SK_er sont calculés à partir de SKEYSEED comme spécifié au paragraphe 2.14, en utilisant SPIi, SPIr, Ni, et Nr provenant du nouvel échange, et en utilisant la nouvelle PRF de la SA IKE.

2.19 Demande d'une adresse interne sur un réseau distant

Dans le scénario de point d'extrémité à passerelle de sécurité, il se produit souvent qu'un point d'extrémité ait besoin d'une adresse IP dans le réseau protégé par la passerelle de sécurité et qu'il puisse avoir besoin que cette adresse soit allouée de façon dynamique. Une demande d'une telle adresse temporaire peut être incluse dans toute demande de création d'une SA fille (incluant la demande implicite dans le message 3) en incluant une charge utile CP. Noter cependant, qu'il est usuel de n'allouer qu'une seule adresse IP durant l'échange IKE_AUTH. Cette adresse persiste au moins jusqu'à la suppression de la SA IKE.

Cette fonction fournit l'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 SA IKE et une SA fille, l'IRAC DOIT demander l'adresse contrôlée 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 de n'importe quel nombre de sources telles qu'un serveur DHCP/BOOTP (protocole Bootstrap) ou 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 les variantes du protocole où il y a plusieurs échanges IKE_AUTH, les charges utiles CP DOIVENT être insérées dans les messages contenant les charges utiles SA.

CP(CFG_REQUEST) DOIT contenir au moins un attribut INTERNAL_ADDRESS (eIPv4 ou IPv6) mais PEUT contenir tout nombre d'attributs supplémentaires que l'initiateur veut retourner dans la réponse.

Exemple de message de l'initiateur au répondant :

CP(CFG_REQUEST) = INTERNAL_ADDRESS()
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'adresses).

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 que on ne veut pas que l'IRAS effectue une recherche de configuration inutile si l'IRAC ne peut pas traiter le 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 une CP(CFG_REQUEST), l'IRAS DOIT faire échouer la demande, et terminer la création de la SA fille avec une erreur FAILED_CP_REQUIRED. L'erreur FAILED_CP_REQUIRED n'est pas fatale pour la SA IKE ; elle cause simplement l'échec de la création de la SA fille. L'initiateur peut réparer cela en commençant plus tard une nouvelle demande de charge utile Configuration. Il n'y a pas de données associées dans l'erreur FAILED_CP_REQUIRED.

2.20 Demande de la version de l'homologue

Un homologue IKE qui souhaite enquêter sur les informations de version de logiciel IKE de l'autre homologue PEUT utiliser la méthode ci-dessous. C'est un exemple d'une demande de configuration au sein d'un échange INFORMATIONAL, après que la SA IKE et la première SA fille ont été créées.

Une mise en œuvre IKE PEUT refuser de divulguer ses informations de version avant l'authentification ou même après l'authentification dans le cas de certaines mises en œuvre connues pour avoir des faiblesses de 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 d'erreur

Un grand nombre de sortes d'erreurs peuvent survenir durant le traitement IKE. La règle générale est que si une demande est reçue mal formatée, ou inacceptable pour des raisons de politique (telles que pas d'algorithme de chiffrement correspondant) la réponse contient une charge utile Notify qui indique l'erreur. La décision d'envoyer ou non une telle réponse dépend de si il y a ou non une SA IKE authentifiée.

Si il y a une erreur d'analyse ou de traitement d'un paquet de réponse, la règle générale est de ne pas renvoyer de message d'erreur parce que les réponses ne devraient pas générer de nouvelles demandes (et une nouvelle demande serait le seul moyen de renvoyer un message d'erreur). De telles erreurs d'analyse ou de traitement des paquets de réponse devraient quand même causer le nettoyage de l'état IKE par le receveur (par exemple, en envoyant un Supprime pour une mauvaise SA).

Seuls les échecs d'authentification (AUTHENTICATION_FAILED et échec EAP) et les messages mal formés (INVALID_SYNTAX) conduisent à une suppression de la SA IKE sans exiger un échange INFORMATIONAL explicite portant la charge utile Supprime. Les autres conditions d'erreur PEUVENT exiger un tel échange si la politique impose sa nécessité. Si l'échange est terminé avec un échec EAP, une notification AUTHENTICATION_FAILED n'est pas envoyée.

2.21.1 Traitement d'erreur dans IKE_SA_INIT

Les erreurs qui surviennent avant qu'une SA IKE protégée cryptographiquement soit établie doivent être traitées avec beaucoup de soin. Il y a un compromis entre vouloir aider l'homologue à diagnostiquer un problème et donc répondre à l'erreur et vouloir éviter de faire partie d'une attaque de DoS fondée sur des messages contrefaits.

Dans un échange IKE_SA_INIT, toute notification d'erreur cause l'échec de l'échange. Noter que certaines notifications d'erreur telles que COOKIE, INVALID_KE_PAYLOAD ou INVALID_MAJOR_VERSION peuvent conduire à un échange réussi ultérieur. Comme toutes les notifications d'erreur sont complètement non authentifiées, le receveur devrait continuer d'essayer pendant quelques temps avant d'abandonner. Le receveur ne devrait pas agir immédiatement sur la base de la notification d'erreur sauf si des actions correctives sont définies dans la présente spécification, comme pour COOKIE, INVALID_KE_PAYLOAD, et INVALID_MAJOR_VERSION.

2.21.2 Traitement d'erreur dans IKE_AUTH

Toutes les erreurs qui se produisent dans un échange IKE_AUTH, causant l'échec de l'authentification pour quelque raison que ce soit (secret partagé invalide, identifiant invalide, producteur de certificat non fiable, certificat révoqué ou expiré, etc.) DEVRAIENT résulter en une notification AUTHENTICATION_FAILED. Si l'erreur se produit chez le répondant, la notification est retournée dans la réponse protégée, et est usuellement la seule charge utile de cette réponse. Bien que les messages IKE_AUTH soient chiffrés et protégés en intégrité, si l'homologue qui reçoit cette notification n'a pas encore authentifié l'autre extrémité, cet homologue doit traiter les informations avec prudence.

Si l'erreur se produit chez l'initiateur, la notification PEUT être retournée dans un échange INFORMATIONAL séparé, généralement sans autre charge utile. C'est une exception à la règle générale de ne pas commencer de nouveaux échanges sur la base d'erreurs dans les réponses.

Noter, cependant, que les messages de demande qui contiennent une charge utile critique non prise en charge, ou dans lesquelles le message entier est mal formé (plutôt que juste un mauvais contenu de charge utile) DOIVENT être rejetés dans leur totalité, et DOIVENT seulement conduire à une notification UNSUPPORTED_CRITICAL_PAYLOAD ou INVALID_SYNTAX envoyée comme réponse. Le receveur ne devrait dans ce cas pas vérifier les charges utiles relatives à l'authentification.

Si l'authentification a réussi dans l'échange IKE_AUTH, la SA IKE est établie ; cependant, établir la SA fille ou demander des informations de configuration peut encore échouer. Cet échec ne cause pas automatiquement la suppression de la SA IKE. Précisément, un répondant peut inclure toutes les charges utiles associées à l'authentification (IDr, CERT, et AUTH) tout en envoyant des notifications d'erreur pour les échanges portés (FAILED_CP_REQUIRED, NO_PROPOSAL_CHOSEN, et ainsi de suite) et l'initiateur NE DOIT PAS faire échouer l'authentification à cause de cela. L'initiateur PEUT, bien sûr, pour des raisons de politique, supprimer ultérieurement une telle SA IKE.

Dans un échange IKE_AUTH, ou dans l'échange INFORMATIONAL qui le suit immédiatement (en cas d'une erreur survenant dans le traitement d'une réponse à IKE_AUTH) les notifications UNSUPPORTED_CRITICAL_PAYLOAD, INVALID_SYNTAX, et AUTHENTICATION_FAILED sont les seules à causer la suppression de la SA IKE ou sa non création sans une charge utile Supprime. Des documents d'extension pourront définir des nouvelles notifications d'erreur avec cette sémantiques, mais NE DEVRONT PAS les utiliser sauf si l'homologue a démontré qu'il les comprend, comme en utilisant la charge utile Identifiant de fabricant.

2.21.3. Traitement d'erreur après l'authentification de la SA IKE

Après l'authentification de la SA IKE, toutes les demandes qui ont des erreurs DOIVENT résulter en une réponse qui notifie l'erreur à l'autre extrémité.

Dans des situations normales, il ne devrait pas y avoir de cas où une réponse valide provenant d'un homologue résulte en une situation d'erreur chez l'autre homologue, de sorte qu'il ne devrait y avoir aucune raison pour qu'un homologue envoie des messages d'erreur à l'autre extrémité sauf comme réponse. Parce que l'envoi d'un tel message d'erreur comme échange INFORMATIONAL pourrait conduire à plus d'erreurs qui pourraient cause des boucles, de telles erreurs NE DEVRAIENT PAS être envoyées. Si il y a des erreurs qui indiquent que les homologues n'ont pas le même état, il peut être bon de supprimer la SA IKE pour nettoyer l'état, et de recommencer.

Si un homologue qui analyse une demande remarque qu'elle est mal formatée (après qu'elle a réussi aux vérifications de code d'authentification de message et de fenêtre) et qu'il retourne une notification INVALID_SYNTAX, cette notification d'erreur

est alors considérée comme fatale chez les deux homologues, ce qui signifie que la SA IKE est supprimée sans qu'il soit besoin d'une charge utile Supprime explicite.

2.21.4 Traitement d'erreur en dehors d'une SA IKE

Un nœud doit limiter son taux d'envoi de messages en réponse à des messages non protégés.

Si un nœud reçoit un message sur l'accès UDP 500 ou 4500 en dehors du contexte d'une SA IKE connue de lui (et si le message n'est pas une demande de commencer une SA IKE) ce peut être le résultat d'une récente panne du nœud. Si le message est marqué comme réponse, le nœud peut examiner l'événement suspect mais NE DOIT PAS répondre. Si le message est marqué comme demande, le nœud peut examiner l'événement suspect et PEUT envoyer une réponse. Si une réponse est envoyée, elle DOIT l'être à l'adresse IP et l'accès d'où elle vient avec le même SPI IKE et l'identifiant de message copiés. La réponse NE DOIT PAS être protégée cryptographiquement et DOIT contenir une charge utile Notify INVALID_IKE_SPI. La notification INVALID_IKE_SPI indique qu'un message IKE a été reçu avec un SPI de destination non reconnu ; ceci indique généralement que le receveur a réamorcé et oublié l'existence d'une SA IKE.

Un homologue qui reçoit une telle charge utile Notify non protégée NE DOIT PAS répondre et NE DOIT PAS changer l'état des SA existantes. Le message pourrait être une contrefaçon ou pourrait être une réponse qu'un authentique correspondant a été amené par ruse à envoyer. Un nœud devrait traiter un tel message (et aussi un message réseau comme un ICMP Destination injoignable) comme une indication qu'il pourrait y avoir des problèmes avec les SA à cette adresse IP et devrait initier une vérification de vivacité pour toute SA IKE de cette sorte. Une mise en œuvre DEVRAIT limiter la fréquence de tels essais pour éviter d'être piégée à participer à une attaque de DoS.

Si une erreur se produit en dehors du contexte d'une demande IKE (par exemple, le nœud reçoit des messages ESP sur un SPI non existant) le nœud DEVRAIT initier un échange INFORMATIONAL avec une charge utile Notify décrivant le problème.

Un nœud qui reçoit un message suspect d'une adresse IP (et accès, si une traversée de NAT est utilisée) avec lequel il a une SA IKE DEVRAIT envoyer une charge utile Notify IKE dans un échange INFORMATIONAL sur cette SA. Le receveur NE DOIT PAS changer l'état d'une SA par suite de cela, mais peut souhaiter examiner l'événement pour aider à diagnostiquer des dysfonctionnements.

2.22 IPComp

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

La négociation de la compression IP est séparée de la négociation des paramètres cryptographiques associés à une SA fille. Un nœud qui demande une SA fille PEUT annoncer qu'il prend en charge un ou plusieurs algorithmes de compression au moyen d'une ou plusieurs charges utiles Notify de type IPCOMP_SUPPORTED. Ce message Notify ne peut être inclus que dans un message contenant une charge utile SA de négociation d'une SA fille et indique la volonté de son envoyeur d'utiliser IPComp sur cette SA. La réponse PEUT indiquer l'acceptation d'un seul algorithme de compression avec une charge utile Notify de type IPCOMP_SUPPORTED. Ces charges utiles NE DOIVENT PAS se produire dans des messages qui ne contiennent pas de charge utile SA.

Les données associées à ce message Notify incluent un CPI IPComp de deux octets suivi par un identifiant de transformation d'un octet facultativement suivi 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 la prise en charge de plusieurs algorithmes. Un message acceptant une SA ne peut en contenir plus d'un.

La liste des identifiants de transformation figure ci après. Les valeurs du tableau suivant ne sont actuelles qu'à la date de publication de la RFC 4306. D'autres valeurs peuvent avoir été ajoutées depuis ou seront ajoutées après la publication du présent document. Les lecteurs devraient se référer à [IKEV2IANA] pour les dernières valeurs.

Nom	Numéro	Défini dans
IPCOMP_OUI	1	(non spécifié)
IPCOMP_DEFLATE	2	RFC 2394
IPCOMP_LZS	3	RFC 2395
IPCOMP_LZJH	4	RFC 3051

Bien qu'il y ait eu des discussions sur la question de permettre que plusieurs algorithmes de compression soient acceptés et d'avoir différents algorithmes de compression disponibles pour les deux directions d'une SA fille, 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 avec un algorithme autre que ceux proposés et acceptés dans l'établissement de la SA fille.

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 de chiffrement et de proposer la compression IP avec certains d'entre eux mais pas les autres.

Dans certains cas, la compression d'en-tête robuste (ROHC, *Robust Header Compression*) peut être plus appropriée que la compression IP. La [RFC5857] définit l'utilisation de ROHC avec IKEv2 et IPsec.

2.23 Traversée de NAT

Les passerelles de traduction d'adresse réseau (NAT, *Network Address Translation*) sont un sujet controversé. Ce paragraphe décrit brièvement ce qu'elles sont et comment elles vont probablement agir sur le trafic IKE. De nombreuses personnes croient que les NAT sont diaboliques et qu'on ne devrait pas concevoir nos protocoles de façon à ce qu'elles fonctionnent mieux. IKEv2 spécifie bien des règles de traitement non intuitives 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 sein du réseau derrière le NAT mais qui sont probablement 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 qui ont des adresses uniques au monde, mais pas avec les nœuds derrière d'autres NAT. Il y a des exceptions à cette règle. Quand ces nœuds font des connexions avec des nœuds sur l'Internet réel, la passerelle de NAT "traduit" l'adresse IP de source en une adresse qui va être réacheminée sur la passerelle. Les messages à la passerelle provenant de l'Internet ont leurs adresses de destination "traduites" en l'adresse interne qui va acheminer le paquet au nœud d'extrémité correct.

Les NAT sont conçus pour être "transparents" aux nœuds d'extrémité. Ni le logiciel sur le nœud 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 comportent les adresses IP des points d'extrémité au sein des charges utiles du paquet vont échouer sauf si la passerelle de NAT comprend le protocole et modifie les références internes ainsi que celles dans les en-têtes. Cette connaissance est par nature non fiable, est une violation de la couche réseau, et résulte souvent en des problèmes subtils.

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

Il est de pratique courante des NAT de traduire les numéros d'accès TCP et UDP ainsi que les adresses et d'utiliser les numéros d'accès des paquets entrants pour décider quel nœud interne devrait recevoir un certain paquet. Pour cette raison, bien que les paquets IKE DOIVENT être envoyés aux et des accès UDP 500 ou 4500, ils DOIVENT être acceptées venant de tout accès et les réponses DOIVENT être envoyées à l'accès d'où ils sont venus. 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 celles-ci sont protégées cryptographiquement et ne pourraient pas être modifiées de façon transparente par les NAT.

L'accès 4500 est réservé à ESP et IKE encapsulé dans IKE . Un point d'extrémité IPsec qui découvre un NAT entre lui et son correspondant (comme décrit plus loin) DOIT envoyer tout le trafic qui suit à partir de l'accès 4500, que les NAT ne devraient pas traiter de façon spéciale (comme ils peuvent faire avec l'accès 500).

Un initiateur peut utiliser l'accès 4500 pour IKE et ESP, sans considération de si il y a ou non un NAT, même au début de IKE. Quand les deux côtés utilisent l'accès 4500, l'envoi de ESP avec encapsulation UDP n'est pas exigé, mais ce qui est exigé est de comprendre les paquets ESP reçus encapsulés dans UDP. L'encapsulation UDP NE DOIT PAS être faite sur l'accès 500. Si la traversée de traducteur d'adresse réseau (NAT-T, *Network Address Translation Traversal*) est prise en charge (c'est-à-dire, si des charges utiles NAT_DETECTION_*_IP ont été échangées durant IKE_SA_INIT) tous les appareils DOIVENT être capables de recevoir et traiter les paquets ESP aussi bien encapsulés UDP que non encapsulés UDP à tout moment. L'un et

l'autre côté peuvent décider d'utiliser ou non l'encapsulation UDP pour ESP sans considération du choix fait par l'autre côté. Cependant, si un NAT est détecté, les deux appareils DOIVENT utiliser l'encapsulation UDP pour ESP.

Les exigences spécifiques pour la prise en charge de la traversée de NAT [RFC3715] sont mentionnées ci-après. La prise en charge de la traversée de NAT est facultative. Dans ce seul paragraphe, les exigences marquées DOIT ne s'appliquent qu'aux mises en œuvre qui prennent en charge la traversée de NAT.

- L'initiateur et le répondant DOIVENT tous deux inclure dans leurs paquets IKE_SA_INIT des charges utiles Notify de 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 situation de ces charges utiles dans les paquets IKE_SA_INIT est juste après les charges utiles Ni et Nr (avant la charge utile facultative CERTREQ).

- Les données associées aux notifications NAT_DETECTION_SOURCE_IP sont un résumé SHA-1 des SPI (dans l'ordre où ils apparaissent dans l'en-tête) adresse IP, et accès à partir desquels ce paquet a été envoyé. Il PEUT y avoir plusieurs charges utiles NAT_DETECTION_SOURCE_IP dans un message si l'envoyeur ne sait pas lequel des rattachements réseau va être utilisé pour envoyer le paquet.

- Les données associées à la notification NAT_DETECTION_DESTINATION_IP sont un résumé SHA-1 des SPI (dans l'ordre d'apparition dans l'en-tête) de l'adresse IP, et de l'accès auquel ce paquet a été envoyé.

- Le receveur de l'une ou l'autre des notifications NAT_DETECTION_SOURCE_IP ou NAT_DETECTION_DESTINATION_IP PEUT comparer la valeur fournie à un hachage SHA-1 des SPI, respectivement, adresse IP de source ou de receveur, et accès, et si elles ne correspondent pas, il DEVRAIT permettre la traversée de NAT. Dans le cas d'une discordance du hachage de NAT_DETECTION_SOURCE_IP avec toutes les charges utiles NAT_DETECTION_SOURCE_IP reçues, le receveur PEUT rejeter la tentative de connexion si la traversée de NAT n'est pas acceptée. Dans le cas de discordance du hachage de NAT_DETECTION_DESTINATION_IP cela signifie que le système qui reçoit la charge utile NAT_DETECTION_DESTINATION_IP est derrière un NAT et que le système DEVRAIT commencer à envoyer des paquets Garder en vie comme défini dans la [RFC3948] ; autrement, il PEUT rejeter la tentative de connexion si la traversée de NAT n'est pas supportée.

- Si aucune des charges utiles NAT_DETECTION_SOURCE_IP reçues ne correspond à la valeur attendue des sources et accès IP trouvés dans l'en-tête IP du paquet contenant la charge utile, cela signifie que le système qui envoie ces charges utiles est derrière un NAT (c'est-à-dire, quelqu'un le long du chemin a changé l'adresse de source du paquet d'origine pour correspondre à l'adresse du boîtier de NAT). Dans ce cas, le système qui reçoit les charges utiles devrait permettre des mises à jour dynamiques de l'adresse IP de l'autre système, comme décrit plus loin.

- L'initiateur IKE DOIT vérifier si les charges utiles NAT_DETECTION_SOURCE_IP ou NAT_DETECTION_DESTINATION_IP 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 SA IKE 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 les SPI, et que les SPI ne peuvent pas validement être zéro, il est toujours possible de distinguer les messages ESP et IKE.

- Les mises en œuvre DOIVENT traiter les paquets ESP reçus encapsulés dans UDP même quand aucun NAT n'est détecté.

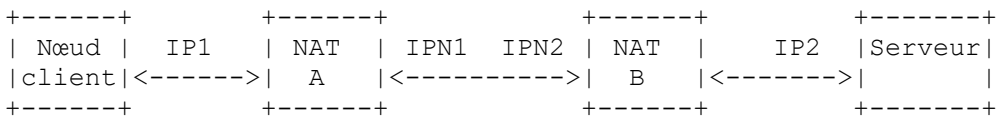
- Les adresses IP originales de source et de destination exigées pour le calcul de la somme de contrôle de mode transport de paquet TCP et UDP (voir la [RFC3948]) sont obtenues des sélecteurs de trafic associés à l'échange. Dans le cas de traversée de NAT en mode transport, les sélecteurs de trafic DOIVENT contenir exactement une adresse IP, qui est alors utilisée comme adresse IP originale. Ceci est couvert plus en détail au paragraphe 2.23.1.

- Il y a des cas où un boîtier de NAT décide de supprimer les transpositions qui sont encore actives (par exemple, l'intervalle de garder en vie est trop long, ou le boîtier de NAT est réamorcé). Ceci va être apparent à un hôte si il reçoit un paquet dont la protection d'intégrité le valide, mais qui a un accès une adresse, ou les deux, différents de celui qui était associé à la SA dans le paquet validé. Quand un tel paquet validé est trouvé, un hôte qui ne prend pas en charge d'autres méthodes de récupération comme la mobilité et multi rattachement IKEv2 (MOBIKE) [RFC4555], et qui n'est pas derrière un NAT, DEVRAIT envoyer tous les paquets (incluant les paquets en retransmission) à l'adresse et accès IP qui figurent dans le paquet validé, et DEVRAIT mémoriser cela comme la nouvelle combinaison d'adresse et d'accès pour la SA (c'est-à-dire, il DEVRAIT mettre à jour l'adresse de façon dynamique). Un hôte derrière un NAT NE DEVRAIT PAS faire ce type de mise à jour dynamique d'adresse si un paquet validé a des valeurs différentes d'accès et/ou d'adresse parce que cela ouvre la porte à une possible attaque de DoS (comme de permettre à un attaquant de casser la connexion avec un seul paquet). Aussi, la mise à jour dynamique d'adresse ne devrait être faite qu'en réponse à un nouveau paquet ; autrement, un attaquant peut

inverser les adresses avec de vieux paquets répétés. À cause de cela, les mises à jour dynamiques ne peuvent être faites en toute sécurité que si la protection contre la répétition est activée. Quand IKEv2 est utilisé avec MOBIKE, mettre à jour dynamiquement les adresses comme décrit ci-dessus interfère avec la façon dont MOBIKE récupère de la même situation. Voir au paragraphe 3.8 de la [RFC4555] des informations supplémentaires.

2.23.1 Traversée de NAT en mode de transport

Le mode transport utilisé avec la traversée de NAT requiert un traitement particulier des sélecteurs de trafic utilisés dans IKEv2. Le scénario complet ressemble à :



(Les autres scénarios sont des simplifications de ce cas complexe, de sorte que la discussion utilise le scénario complet.)

Dans ce scénario, il y a deux NAT qui traduisent l'adresse : le NAT A et le NAT B. Le NAT A est un NAT dynamique qui transpose l'adresse de source IP1 du client en IPN1. Le NAT B est un NAT statique configuré de façon à ce que les connexions venant à l'adresse IPN2 soient transposées en l'adresse IP2 de la passerelle, c'est-à-dire que l'adresse de destination IPN2 est transposée en IP2. Cela permet au client de se connecter à un serveur en se connectant à IPN2. Le NAT B n'a pas nécessairement besoin d'être un NAT statique, mais le client a besoin de savoir comment se connecter au serveur, et il ne peut le faire que si il connaît d'une façon ou d'une autre l'adresse externe du NAT B, c'est-à-dire, l'adresse IPN2. Si le NAT B est un NAT statique, son adresse peut alors être configurée à la configuration du client. Une autre option serait de la trouver en utilisant un autre protocole (comme le DNS) mais cela sort du domaine d'application de IKEv2.

Dans ce scénario, le client et le serveur sont tous deux configurés à utiliser le mode transport pour le trafic originaire du nœud client et destiné au serveur.

Quand le client commence à créer la SA IKEv2 et la SA fille pour l'envoi du trafic au serveur, il peut avoir un paquet déclencheur avec une adresse IP de source de IP1, et une adresse IP de destination de IPN2. Sa base de données d'autorisation d'homologues (PAD, *Peer Authorization Database*) et sa SPD ont besoin d'avoir une configuration correspondant à ces adresses (ou des entrées à caractères génériques qui les couvrent). Parce que ceci est du mode transport, il utilise exactement les mêmes adresses que les sélecteurs de trafic et l'adresse IP externe des paquets IKE. Pour le mode transport, il DOIT utiliser exactement une adresse IP dans les charges utiles TSi et TSr. Il peut avoir plusieurs sélecteurs de trafic si il a, par exemple, plusieurs gammes d'accès qu'il veut négocier, mais toutes les entrées TSi doivent utiliser la gamme IP1-IP1 comme adresses IP, et toutes les entrées TSr doivent avoir la gamme IPN2-IPN2 comme adresses IP. Le premier sélecteur de trafic de TSi et TSr DEVRAIT avoir des sélecteurs de trafic très spécifiques incluant des numéros de protocole et d'accès, comme ceux provenant du paquet qui déclenche la demande.

Le NAT A va alors remplacer l'adresse de source du paquet IKE de IP1 en IPN1, et le NAT B va remplacer l'adresse de destination du paquet IKE de IPN2 en IP2, de sorte que quand le paquet arrive au serveur il va encore avoir exactement les mêmes sélecteurs de trafic qui ont été envoyés par le client, mais l'adresse IP du paquet IKE a été remplacée par IPN1 et IP2.

Quand le serveur reçoit ce paquet, il regarde normalement dans la PAD décrite dans la [RFC4301] sur la base de l'identifiant et ensuite recherche dans la SPD sur base des sélecteurs de trafic. Comme IP1 n'a en fait rien à voir avec le serveur (il est l'adresse que le client a derrière le NAT) il est inutile de faire une recherche sur la base de cela si le mode transport est utilisé. Par ailleurs, le serveur ne peut pas savoir si le mode transport est permis par sa politique avant qu'il trouve l'entrée de SPD correspondante.

Dans ce cas, le serveur devrait d'abord vérifier que l'initiateur a demandé le mode transport, et ensuite faire la substitution d'adresse sur les sélecteurs de trafic. Il a besoin de mémoriser d'abord les adresses IP du vieux sélecteur de trafic pour les utiliser ultérieurement pour le calcul de la somme de contrôle incrémentaire (l'adresse IP dans le TSi peut être mémorisée comme adresse de source originelle et l'adresse IP dans le TSr peut être mémorisée comme adresse originelle de destination). Après cela, si l'autre extrémité a été détectée comme étant derrière un NAT, le serveur remplace l'adresse IP dans les charges utiles TSi par l'adresse IP obtenue de l'adresse de source du paquet IKE reçu (c'est-à-dire, il remplace IP1 dans TSi par IPN1). Si l'extrémité serveur a été détectée comme étant derrière un NAT, il remplace l'adresse IP dans les charges utiles TSr par l'adresse IP obtenue de l'adresse de destination du paquet IKE reçu (c'est-à-dire, il remplace IPN2 dans TSr par IP2).

Après cette substitution d'adresse, les sélecteurs de trafic et les adresses IKE UDP de source/destination paraissent semblables, et le serveur fait une recherche dans la SPD sur la base de ces nouveaux sélecteurs de trafic. Si il trouve une entrée et si elle permet le mode transport, cette entrée est alors utilisée. Si une entrée est trouvé mais qu'elle ne permet pas le mode transport, le serveur PEUT alors défaire la substitution d'adresse et refaire la recherche dans la SPD en utilisant les sélecteurs de trafic

d'origine. Si la seconde recherche réussit, la serveur va créer une SA en mode tunnel en utilisant les sélecteurs de trafic réels envoyés par l'autre extrémité.

Cette substitution d'adresse en mode transport est nécessaire parce que la SPD est parcourue en utilisant les adresses qui vont être vues par l'hôte local. Cela va aussi assurer que les entrées de la base de données d'association de sécurité (SAD) pour les vérifications de sortie de tunnel et les paquets de retour sont ajoutées en utilisant les adresses telles que vues par la pile du système d'exploitation local.

Le cas le plus courant est que la SPD du serveur va contenir des entrées de caractères génériques correspondant à toutes les adresses, mais cela permet aussi de faire des entrées de SPD différentes, par exemple, pour les différentes adresses externes de NAT connues.

Après la recherche dans la SPD, le serveur va faire le rétrécissement de sélecteur de trafic sur la base de l'entrée de SPD qu'il a trouvée. Il va encore utiliser les sélecteurs de trafic déjà substitués, et il va donc renvoyer les sélecteurs de trafic qui ont IPN1 et IP2 comme adresses IP ; il peut quand même rétrécir le numéro de protocole ou les gammes d'accès utilisés par les sélecteurs de trafic. L'entrée de SAD créée pour la SA fille aura les adresses telles que vues par le serveur, à savoir IPN1 et IP2.

Quand le client reçoit la réponse du serveur à la SA fille, il va faire un traitement similaire. Si la SA en mode transport a été créée, le client peut mémoriser les sélecteurs de trafic originaux retournés comme adresses originales de source et destination. Il va remplacer les adresses IP dans les sélecteurs de trafic par celles provenant de l'en-tête IP du paquet IKE : il va remplacer IPN1 par IP1 et IP2 par IPN2. Ensuite, il va utiliser ces sélecteurs de trafic quand il vérifiera la SA par rapport aux sélecteurs de trafic envoyés, et quand il installera l'entrée de SAD.

Un résumé des règles pour la traversée de NAT en mode transport est :
Pour le client qui propose le mode transport :
- Les entrées TSi DOIVENT avoir exactement une adresse IP, et elle DOIT correspondre à l'adresse de source de la SA IKE.
- Les entrées TSr DOIVENT avoir exactement une adresse IP, et elle DOIT correspondre à l'adresse de destination de la SA IKE.
- Les premiers sélecteurs de trafic TSi et TSr DEVRAIENT avoir des sélecteurs de trafic très spécifiques incluant les numéros de protocole et d'accès, comme dans le paquet qui a déclenché la demande.
- Il PEUT y avoir plusieurs entrées TSi et TSr.
- Si le mode transport a été choisi pour la SA (c'est-à-dire, si le serveur incluait une notification USE_TRANSPORT_MODE dans sa réponse) :
  - Mémoriser les sélecteurs de trafic originaux comme étant les adresses de source et de destination reçues.
  - Si le serveur est derrière un NAT, substituer à l'adresse IP dans les entrées TSr l'adresse distante de la SA IKE.
  - Si le client est derrière un NAT, substituer à l'adresse IP dans les entrées TSi l'adresse locale de la SA IKE.
  - Faire la substitution d'adresse avant d'utiliser ces sélecteurs de trafic pour tout ce qu n'est pas la mémorisation de leur contenu original. Cela inclut la vérification que les sélecteurs de trafic ont été rétrécis correctement par l'autre extrémité, la création de l'entrée de SAD, et ainsi de suite.

Pour le répondant, quand le mode transport est proposé par le client :
- Mémoriser les adresses IP du sélecteur de trafic original comme adresses de source et de destination reçues, pour le cas où il serait nécessaire de défaire la substitution d'adresses, pour les utiliser comme "adresse réelle de source et de destination" spécifiée par la [RFC3948], et pour le calcul de la somme de contrôle TCP/UDP.
- Si le client est derrière un NAT, substituer à l'adresse IP dans les entrées TSi l'adresse distante de la SA IKE.
- Si le serveur est derrière un NAT, substituer à l'adresse IP dans les entrées TSr l'adresse locale de la SA IKE.
- Faire une recherche dans la PAD et la SPD en utilisant l'identifiant et les sélecteurs de trafic substitués.
- Si aucune entrée de SPD n'est trouvée, ou (si trouvée) l'entrée de SPD ne permet pas le mode transport, défaire les substitutions de sélecteur de trafic. Faire à nouveau une recherche dans la PAD et la SPD en utilisant l'identifiant et les sélecteurs de trafic originaux, mais aussi en recherchant une entrée de mode tunnel dans la SPD (c'est-à-dire, revenir au mode tunnel).
- Cependant, si une entrée de SPD en mode transport a été trouvée, faire le rétrécissement normal de sélection de trafic sur la base des sélecteurs de trafic substitués et de l'entrée de SPD. Utiliser les sélecteurs de trafic résultants lors de la création des entrées de SAD, et lors du renvoi des sélecteurs de trafic au client.

2.24 Notification explicite d'encombrement (ECN)

Quand les tunnels IPsec se comportent comme originellement spécifié dans la [RFC2401], l'usage de 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 de ECN pour les tunnels IPsec pour IPsec fondé sur IKEv1 exige plusieurs modes opératoires 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 filles en mode tunnel créées par IKEv2. Spécifiquement, les encapsulateurs et désencapsulateurs de tunnel pour toutes les SA en mode tunnel créées par IKEv2 DOIVENT prendre en charge l'option de pleine fonctionnalité ECN pour les tunnels spécifiés dans la [RFC3168] et DOIVENT mettre en œuvre le traitement d'encapsulation et désencapsulation de tunnel spécifié dans la [RFC4301] pour empêcher d'éliminer les indications d'encombrement de ECN.

2.25. Collisions d'échange

Parce que les échanges IKEv2 peuvent être initiés par l'un et l'autre homologue, il est possible que deux échanges affectant la même SA se chevauchent partiellement. Cela peut conduire à une situation où les informations d'état de la SA sont temporairement non synchronisées, et un homologue peut recevoir une demande qu'il ne peut pas traiter d'une façon normale.

Évidemment, l'utilisation d'une taille de fenêtre supérieure à 1 conduit à des situations plus complexes, en particulier si les demandes sont traitées dans le désordre. Ce paragraphe se consacre aux problèmes qui peuvent survenir même avec une taille de fenêtre de 1, et recommande des solutions.

Une notification TEMPORARY_FAILURE DEVRAIT être envoyée quand un homologue reçoit une demande qui ne peut pas être achevée à cause d'une condition temporaire telle qu'une opération de changement de clé. Quand un homologue reçoit une notification TEMPORARY_FAILURE, il NE DOIT PAS réessayer immédiatement l'opération; il DOIT attendre que l'envoyeur puisse achever l'opération quelle qu'elle soit qui a causé la condition temporaire. Le receveur PEUT réessayer la demande une ou plusieurs fois sur une période de plusieurs minutes. Si un homologue continue de recevoir des TEMPORARY_FAILURE sur la même SA IKE après plusieurs minutes, il DEVRAIT en conclure que les informations d'état sont désynchronisées et clore la SA IKE.

Une notification CHILD_SA_NOT_FOUND DEVRAIT être envoyée quand un homologue reçoit une demande de changement de clés d'une SA fille qui n'existe pas. La SA dont l'initiateur a tenté de changer les clés est indiquée par le champ SPI dans la charge utile Notify, qui est copié du champ SPI dans la notification REKEY_SA. Un homologue qui reçoit une notification CHILD_SA_NOT_FOUND DEVRAIT éliminer en silence la SA fille (si elle existe toujours) et envoyer une demande de création d'une nouvelle SA fille à partir de rien (si la SA fille n'existe pas encore).

2.25.1 Collisions lors de changement de clé ou de clôture de SA filles

Si un homologue reçoit une demande de changer les clés d'une SA fille qu'il est actuellement en train d'essayer de fermer, il DEVRAIT répondre par une TEMPORARY_FAILURE. Si un homologue reçoit une demande de changer les clés d'une SA fille et qu'il est actuellement en train de le faire, il DEVRAIT répondre comme d'habitude, et DEVRAIT se préparer à clore ultérieurement les SA redondantes sur la base des noms occasionnels (voir au paragraphe 2.8.1). Si un homologue reçoit une demande de changement de clés d'une SA fille qui n'existe pas, il DEVRAIT répondre par CHILD_SA_NOT_FOUND.

Si un homologue reçoit une demande de clore une SA fille qu'il est actuellement en train d'essayer de clore, il DEVRAIT répondre sans charge utile Supprime (voir au paragraphe 1.4.1). Si un homologue reçoit une demande de clore une SA fille dont il est actuellement en train de changer les clés, il DEVRAIT répondre comme d'habitude, avec une charge utile Supprime. Si un homologue reçoit une demande de clore une SA fille qui n'existe pas, il DEVRAIT répondre sans charge utile Supprime.

Si un homologue reçoit une demande de changement des clés de la SA IKE, et si il est actuellement en train de créer, changer les clés, ou clore une SA fille de cette SA IKE, il DEVRAIT répondre par TEMPORARY_FAILURE.

2.25.2 Collisions lors de changement de clé ou de clôture de SA IKE

Si un homologue reçoit une demande de changer les clés d'une SA IKE et qu'il est actuellement en train de le faire, il DEVRAIT répondre comme d'habitude, et DEVRAIT se préparer à clore ultérieurement les SA redondantes et déplacer les SA filles héritées sur la base des noms occasionnels (voir au paragraphe 2.8.2). Si un homologue reçoit une demande de changement des clés d'une SA IKE qu'il est actuellement en train de clore, il DEVRAIT répondre par TEMPORARY_FAILURE.

Si un homologue reçoit une demande de clore une SA IKE dont il est actuellement en train de changer les clés, il DEVRAIT répondre comme d'habitude, et oublier sa propre demande de changement de clés. Si un homologue reçoit une demande de clore une SA IKE qu'il est en train d'essayer de fermer, il DEVRAIT répondre comme d'habitude, et oublier sa propre demande de clôture.

Si un homologue reçoit une demande de création ou de changement de clés d'une SA fille quand il est actuellement en train de changer les clés de la SA IKE, il DEVRAIT répondre par TEMPORARY_FAILURE. Si un homologue reçoit une demande de

suppression d'une SA fille quand il est actuellement en train de changer les clés de la SA IKE, il DEVRAIT répondre comme d'habitude, avec une charge utile Supprime.

3. Formats d'en-tête et de charge utile

Dans les tableaux de cette Section, des primitives cryptographiques et des attributs de configuration sont marqués "NON SPÉCIFIÉ". Ce sont des éléments pour lesquels il n'y a pas de spécification connue et donc l'interopérabilité est actuellement impossible. Une future spécification pourrait décrire leur utilisation, mais jusqu'à ce qu'une telle spécification soit faite, les mises en œuvre NE DEVRAIENT PAS tenter d'utiliser les éléments marqués "NON SPÉCIFIÉ" quand elles souhaitent être interopérables.

3.1 En-tête IKE

Les messages IKE utilisent l'accès UDP 500 et/ou 4500, avec un message IKE par datagramme UDP. Les informations provenant du début du paquet jusqu'à l'en-tête UDP sont largement ignorées sauf que les adresses IP et accès UDP provenant des en-têtes sont inversés et utilisés pour les paquets de retour. Lorsque ils sont envoyés sur l'accès UDP 500, les messages IKE commencent immédiatement après l'en-tête UDP. Lorsque envoyés sur l'accès UDP 4500, les messages IKE ont ajoutés en tête quatre octets de zéros. Ces quatre octets de zéros ne font partie du message IKE et ne sont pas inclus dans les champs de longueur ou les sommes de contrôle définies par IKE. Chaque message IKE commence par l'en-tête IKE, noté HDR dans le présent document. À la suite de l'en-tête sont une ou plusieurs charges utiles IKE dont chacune est identifiée par un champ Prochaine charge utile qui précède la charge utile. Les charges utiles sont identifiées dans l'ordre dans lequel elles apparaissent dans un message IKE en regardant le champ Prochaine charge utile dans l'en-tête IKE, et ensuite, selon le 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 "Chiffré" est trouvée, cette charge utile est déchiffrée et son contenu analysé comme des charges utiles supplémentaires. Une charge utile Chiffré DOIT être la dernière charge utile d'un paquet et une charge utile Chiffré NE DOIT PAS contenir une autre charge utile Chiffré.

Les SPI du répondant dans l'en-tête identifient une instance d'une association de sécurité IKE. Il est donc possible à une seule instance de IKE de multiplexer des sessions distinctes avec plusieurs homologues, incluant plusieurs sessions par homologue.

Tous les champs multi octets représentant des entiers sont posés en ordre gros boutien (aussi appelé "octet de poids fort en premier", ou "ordre des octets du réseau").

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

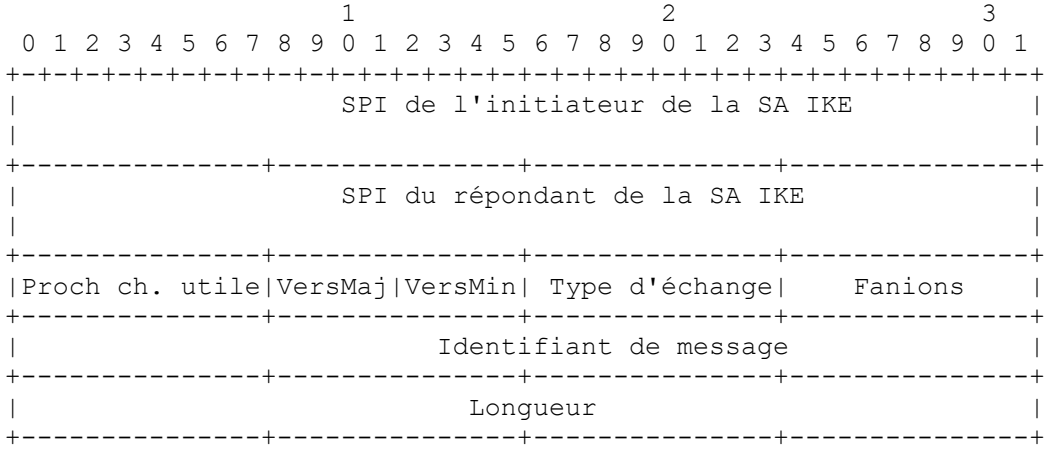


Figure 4 : Format d'en-tête IKE

- SPI d'initiateur (8 octets) - valeur choisie par l'initiateur pour identifier une unique association de sécurité IKE. Cette valeur NE DOIT PAS être zéro.
- SPI du répondant (8 octets) - valeur choisie par le répondant pour identifier une unique association de sécurité IKE. Cette valeur DOIT être zéro dans le premier message d'un échange initial IKE (incluant les répétitions de ce message incluant un cookie).
- 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 plus loin.

- Version majeure (4 bits) - indique la version majeure du protocole IKE utilisé. Les mises en œuvre qui se fondent sur la présente version de IKE DOIVENT régler la version majeure à 2. Les mises en œuvre fondées sur les versions antérieures de IKE et ISAKMP DOIVENT régler la version majeure à 1. Les mises en œuvre qui se fondent sur la version du présent document (version 2) de IKE DOIVENT rejeter ou ignorer les messages contenant un numéro de version supérieur à 2 avec un message de notification INVALID_MAJOR_VERSION comme décrit au paragraphe 2.5.
- Version mineure (4 bits) - indique la version mineure de protocole IKE utilisée. Les mises en œuvre qui se fondent sur la présente version de IKE DOIVENT régler version mineure à 0. Elles DOIVENT ignorer les numéros de version mineure des messages reçus.
- Type d'échange (1 octet) - indique le type de l'échange utilisé. Cela contraint les charges utiles envoyées dans chaque message de l'échange. Les valeurs du tableau qui suit sont seulement celles en cours au moment de la publication de la RFC 4306. D'autres valeurs peuvent avoir été ajoutées depuis ou le seront après la publication du présent document. Le lecteur devrait se référer à [IKEV2IANA] pour les dernières valeurs.

Type d'échange	**Valeur**
IKE_SA_INIT	34
IKE_AUTH	35
CREATE_CHILD_SA	36
INFORMATIONAL	37

- Fanions (1 octet) - indique des options spécifiques qui sont établies pour le message. La présence d'options est indiquée par le réglage à un du bit approprié dans le champ Fanion. Ces bits sont :

```
+-+-+-+-+-+-+-+-+
|X|X|R|V|I|X|X|X|
+-+-+-+-+-+-+-+-+
```

Dans la description qui suit, un bit "établi" signifie que sa valeur est "1", tandis que "non établi" signifie que sa valeur est "0". Les bits "X" DOIVENT être non établis à l'envoi et DOIVENT être ignorés à réception.

* R (Réponse) - ce bit indique que le message est une réponse à un message contenant le même identifiant de message. Ce bit DOIT être non établi dans tous les messages de demande et DOIT être établi dans toutes les réponses. Un point d'extrémité IKE NE DOIT PAS générer une réponse à un message qui est marqué comme étant une réponse (avec une exception au paragraphe 2.21.2).

* V (Version) - ce bit indique que l'émetteur est capable de prendre en charge un numéro de version majeure du protocole supérieur à celui indiqué dans le champ Numéro de version majeure. Les mises en œuvre de IKEv2 DOIVENT avoir ce bit à zéro à l'envoi et DOIVENT l'ignorer dans les messages entrants.

* I (Initiateur) - ce bit DOIT être établi dans les messages envoyés par l'initiateur original de la SA IKE et DOIT être non établi dans les messages envoyés par le répondant original. Il est utilisé par le receveur pour déterminer lesquels des huit octets des SPI ont été générés par le receveur. Ce bit change pour refléter qui a initié le dernier changement de clés de la SA IKE.

- Identifiant de message (4 octets, entier non signé) - identifiant de message utilisé pour contrôler la retransmission de paquets perdus et confronter les demandes et les réponses. Il est essentiel pour 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.

- Longueur (4 octets, entier non signé) - longueur du message total (en-tête + charges utiles) en octets.

3.2 En-tête générique de charge utile

Chaque charge utile IKE définie 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 vont inclure l'en-tête générique de charge utile, mais pour faire court, la description de chaque champ va être omise.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Proch. ch. ut. |C|  RÉSERVÉ    |    Longueur de charge utile   |
+---------------+-+-------------+---------------+---------------+
```

Figure 5 : En-tête générique de charge utile

Les champs En-tête générique de charge utile sont définis comme suit :

- Prochaine charge utile (1 octet) - identifiant du type de charge utile de la prochaine charge utile du message. Si la charge utile en cours est la dernière du message, ce champ sera alors à 0. Ce champ donne une capacité de "chaînage" par lequel des charges utiles supplémentaires peuvent être ajoutées à un message en ajoutant chacune à la fin du message et en établissant le champ Prochaine charge utile de la charge utile précédente 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 des structures de données dans le format des charges utiles supplémentaires. 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) ; à l'inverse, le champ Prochaine charge utile de la dernière charge utile contenue est réglé à zéro. La liste des valeurs de type de charge utile figure ci-dessous. Les valeurs du tableau suivant ne sont que celles en vigueur à la date de publication de la RFC 4306. D'autres valeurs peuvent avoir été ajoutées depuis ou le seront après la publication du présent document. Le lecteur devrait se référer à [IKEV2IANA] pour les dernières valeurs.

Type de prochaine charge utile	**Notation**	**Valeur**
Pas de prochaine charge utile	0	
Association de sécurité	SA	33
Échange de clés	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
Notifie	N	41
Supprime	D	42
Identifiant de fabricant	V	43
Sélecteur de trafic - initiateur	TSi	44
Sélecteur de trafic - répondant	TSr	45
Chiffré et authentifié	SK	46
Configuration	CP	47
Authentification extensible	EAP	48

(Les valeurs de type de charge utile de 1 à 32 ne devraient pas être allouées à l'avenir afin qu'il n'y ait pas de chevauchement avec les code alloués pour IKEv1.)

- Critique (1 bit) - DOIT être réglé 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. Il DOIT être réglé à un si l'envoyeur veut que le receveur rejette ce message entier si il ne comprend pas le type de charge utile. Il DOIT être ignoré par le receveur si il comprend le code de type de charge utile. Il DOIT être réglé à zéro pour les types de charge utile définis dans le présent 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 l'idée de ne pas établir le bit critique pour les charges utiles définies dans le présent document est que toutes les mises en œuvre DOIVENT comprendre tous les types de charge utile définis dans le présent document et donc doivent ignorer la valeur du bit critique. Les charges utiles sautées sont supposées avoir des champs Prochaine charge utile et Longueur de charge utile valides. Voir au paragraphe 2.5 plus d'informations sur ce bit.

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

- Longueur de charge utile (2 octets, entier non signé) - longueur en octets de la charge utile en cours, incluant l'en-tête générique de charge utile.

De nombreuses charges utiles contiennent des champs marqués "RÉSERVÉ". Certaines charges utiles dans IKEv2 (et historiquement dans IKEv1) ne sont pas alignées sur des limites de 4 octets.

3.3 Charge utile Association de sécurité

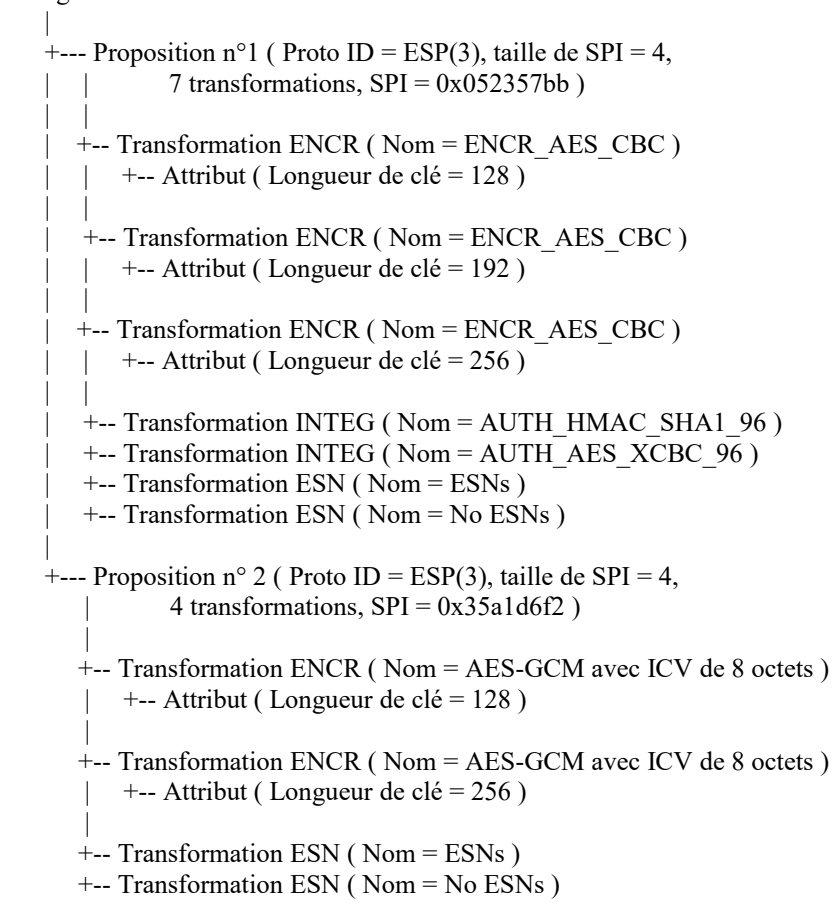
La charge utile Association de sécurité, notée SA dans le présent document, est utilisée pour négocier les attributs d'une association de sécurité. L'assemblage des charges utiles Association de sécurité exige une grande sérénité. Une charge utile SA PEUT contenir plusieurs propositions. Si il y en a plus d'une, elles DOIVENT être ordonnées de la préférée à la moins souhaitée. Chaque proposition contient un seul protocole IPsec (où un protocole est 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 totale de charge utile est cohérente avec les longueurs internes et les comptes de la charge utile. Les propositions, les transformations, et les attributs ont chacun leur propre codage de longueur variable. Ils sont incorporés de telle sorte que le champ Longueur de charge utile d'une SA inclue le contenu combiné des informations de la SA, de la proposition, des transformations, et des attributs. La longueur d'une proposition inclut les longueurs de toutes les transformations et des 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, des propositions, des transformations, et des attributs se fonde sur ISAKMP ; cependant, leur sémantique est quelque peu différente. La raison de la complexité et de la hiérarchie est de permettre que plusieurs combinaisons possibles des algorithmes soient codées dans une seule SA. Parfois, il y a un choix entre plusieurs algorithmes, tandis que d'autres fois, il y a une combinaison des algorithmes. Par exemple, un initiateur peut vouloir proposer d'utiliser ESP avec soit (3DES et HMAC_MD5) soit (AES et HMAC_SHA1).

Une des raisons du changement de la sémantique de la charge utile de SA depuis ISAKMP et IKEv1 est de rendre les codages plus compacts dans le cas courant.

La structure de proposition contient un numéro de proposition et un identifiant de protocole IPsec. Chaque structure DOIT avoir une proposition numéro un (1) supérieure à la structure précédente. La première proposition dans la charge utile SA de l'initiateur DOIT avoir un numéro de proposition de un (1). Une raison d'utiliser plusieurs propositions est de proposer à la fois des chiffrements standard et des chiffrement en mode combiné. Les chiffrements en mode combiné incluent à la fois l'intégrité et le chiffrement dans un seul algorithme de chiffrement, et DOIVENT soit ne pas offrir d'algorithme d'intégrité, soit un seul algorithme d'intégrité de "AUCUN", la méthode RECOMMANDÉE étant pas d'algorithme d'intégrité. Si un initiateur veut proposer les deux chiffrements en mode combiné et en mode normal, il doit inclure deux propositions : une aura tous les chiffrements en mode combiné, et l'autre aura les chiffrements normaux avec les algorithmes d'intégrité. Par exemple, une telle proposition aurait deux structures de proposition. La proposition 1 est ESP avec AES-128, AES-192, et AES-256 en mode de chaînage de bloc de chiffrement (CBC, *Cipher Block Chaining*) avec soit HMAC-SHA1-96, soit XCBC-96 comme algorithme d'intégrité ; la proposition 2 est AES-128 ou AES-256 en mode GCM avec une valeur de vérification d'intégrité (ICV, *Integrity Check Value*) de 8 octets. Les deux propositions permettent sans l'exiger l'utilisation des numéros de séquence étendus (ESN, *Extended Sequence Number*). Ceci peut être illustré comme suit :



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. AH a généralement deux transformations : les numéros de séquence étendus (ESN, *Extended Sequence Numbers*) et un algorithme de vérification d'intégrité. ESP en a généralement trois : ESN, un algorithme de chiffrement, et un algorithme de vérification d'intégrité. IKE a généralement quatre transformations : un groupe Diffie-Hellman, un algorithme de vérification d'intégrité, un algorithme de PRF, et un algorithme de chiffrement. Pour chaque protocole, l'ensemble de transformations permises reçoit des numéros d'identifiant de transformation, qui apparaissent dans l'en-tête de chaque transformation.

Si il y a plusieurs transformations avec le même type de transformation, la proposition est un OU de ces transformations. Si il y a plusieurs transformations avec des types de transformation différents, la proposition est un ET des différents groupes. Par exemple, pour proposer ESP avec (3DES ou AES-CBC) et (HMAC_MD5 ou HMAC_SHA), la proposition ESP contiendrait deux candidats de transformation de type 1 (une pour 3DES et une pour AEC-CBC) et deux candidats de transformation de type 3 (une pour HMAC_MD5 et une pour HMAC_SHA). Cela propose effectivement quatre combinaisons d'algorithmes. Si l'initiateur voulait en proposer seulement un sous ensemble , 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. L'initiateur devrait plutôt construire deux propositions différentes, chacune avec deux transformations.

Une certaine transformation 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 va spécifier l'algorithme et l'attribut va spécifier 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 taille de clés pour l'algorithme de chiffrement AES) une mise en œuvre DOIT inclure plusieurs transformations avec le même type de transformation, chacune 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 avec un porté dans la transformation et les autres portés dans les attributs.

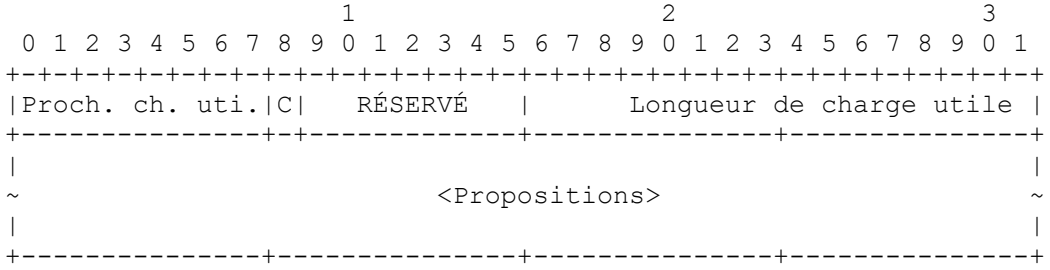


Figure 6 : charge utile Association de sécurité

o Propositions (longueur variable) - une ou plusieurs sous structures de proposition.

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

3.3.1 Sous structure Proposition

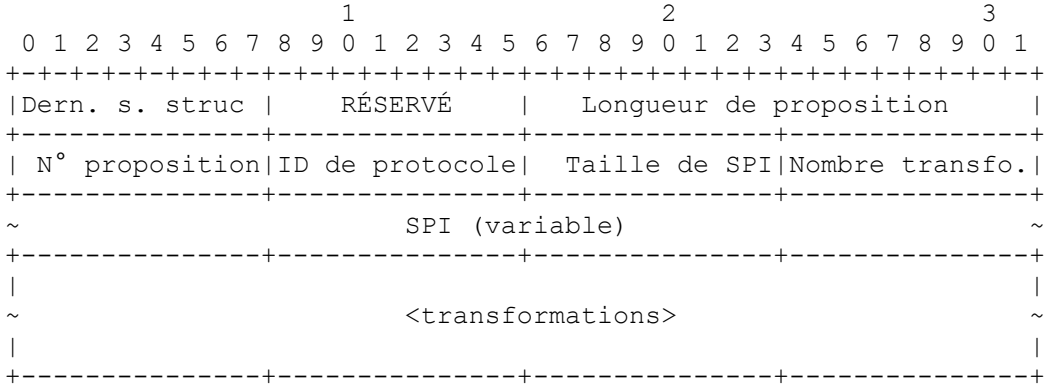


Figure 7 : sous structure Propositions

- Dernière sous structure (1 octet) - spécifie si c'est ou non la dernière sous structure de proposition dans la SA. Ce champ a une valeur de 0 si c'est la dernière sous structure de proposition, et une valeur de 2 si il y a plus de sous structures de propositions. Cette syntaxe est héritée de ISAKMP, mais est inutile parce que la dernière proposition peut être identifiée par la longueur de la SA. La valeur (2) correspond à un type de charge utile 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, entier non signé) - longueur de cette proposition, incluant toutes les transformations et attributs qui suivent.

- Numéro de propositions (1 octet) - quand une proposition est faite, la première proposition dans une charge utile SA DOIT être 1, et les propositions suivantes DOIVENT être un de plus que la proposition précédente (indiquant un OU de deux propositions). Quand une proposition est acceptée, le numéro de proposition dans la charge utile SA DOIT correspondre au numéro de la proposition envoyée qui a été acceptée.

- Identifiant de protocole (1 octet) - spécifie l'identifiant de protocole IPsec pour la négociation en cours. Les valeurs du tableau qui suit ne sont que celles en cours au moment de la publication de la RFC 4306. D'autres valeurs peuvent avoir été ajoutées depuis ou seront ajoutées après la publication du présent document. Le lecteur devrait se référer à [IKEV2IANA] pour les dernières valeurs.

Protocole	**Identifiant de protocole**
IKE	1
AH	2
ESP	3

- Taille de SPI (1 octet) - Pour une négociation initiale de SA IKE, ce champ DOIT être zéro ; les SPI sont obtenus de l'en-tête externe. Durant les négociations ultérieures, il est égal à la taille, en octets, des SPI du protocole correspondant (8 pour IKE, 4 pour ESP et AH).

- Nombre de transformations (1 octet) - spécifie le nombre de transformations dans cette proposition.

- SPI (variable) - SPI de l'entité envoyeuse. Même si la taille de SPI n'est pas un multiple de 4 octets, aucun bourrage n'est appliqué à la charge utile. Quand 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 transformation.

3.3.2 Sous structure de transformation

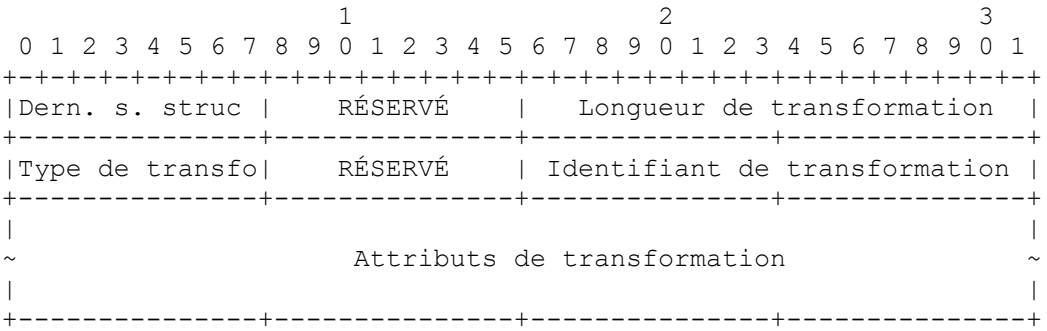


Figure 8 : Sous structure de transformation

- Dernière sous structure (1 octet) - Spécifie si c'est ou non la dernière sous structure de transformation dans la proposition. Ce champ a une valeur de 0 si c'était la dernière sous structure de transformation, et une valeur de 3 si il y a encore des sous structures de transformation. Cette syntaxe est héritée de ISAKMP, mais est inutile parce que la dernière transformation peut être identifiée à partir de la longueur de la proposition. La valeur (3) correspond à un type de charge utile de transformation dans IKEv1, et les quatre premiers octets de la structure de transformation sont conçus pour ressembler un peu à l'en-tête d'une charge utile.

- RÉSERVÉ - DOIT être envoyé à zéro ; DOIT être ignoré à réception.

- Longueur de transformation - longueur (en octets) de la sous structure de transformation incluant l'en-tête et les attributs.

- Type de transformation (1 octet) - type de transformation spécifié dans cette transformation. Des protocoles différents prennent en charge des types de transformation différents. 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 de ce type n'est incluse dans la proposition. Si l'initiateur souhaite rendre l'usage de la transformation facultatif pour le répondant, il inclut une sous structure de transformation avec un identifiant de transformation = 0 comme option.

- Identifiant de transformation (2 octets) - instance spécifique du type de transformation proposée.

La liste des valeurs de type de transformation est donnée ci-dessous. Les valeurs du tableau suivant ne sont que celles qui étaient en cours au moment de la publication de la RFC 4306. D'autres valeurs peuvent avoir été ajoutées depuis ou le seront après la publication du présent document. Le lecteur devrait se référer à [IKEV2IANA] pour avoir les dernières valeurs.

Description	Type de transformation	Utilisée dans
Algorithme de chiffrement (ENCR)	1	IKE et ESP
Fonction pseudo aléatoire (PRF)	2	IKE
Algorithme d'intégrité (INTEG)	3	IKE*, AH, facultatif dans ESP
Groupe Diffie-Hellman (D-H)	4	IKE, facultatif dans AH & ESP
Numéros de séquence étendus (ESN)	5	AH et ESP

(*) La négociation d'un algorithme d'intégrité est obligatoire pour le format de charge utile Chiffré spécifié dans le présent document. Par exemple, la [RFC5282] spécifie des formats supplémentaires sur la base d'un chiffrement authentifié, dans lequel un algorithme séparé d'intégrité n'est pas négocié.

Pour le type de transformation 1 (algorithme de chiffrement) la liste des identifiants de transformation est donnée ci-dessous. Les valeurs du tableau qui suit sont seulement celles qui étaient en cours à la date de publication de la RFC 4306. D'autres valeurs peuvent avoir été ajoutées depuis ou le seront après la publication du présent document. Le lecteur devrait se référer à [IKEV2IANA] pour avoir les dernières valeurs.

Nom	Numéro	Défini dans
ENCR_DES_IV64	1	non spécifié
ENCR_DES	2	[RFC2405], [DES]
ENCR_3DES	3	[RFC2451]
ENCR_RC5	4	[RFC2451]
ENCR_IDEA	5	[RFC2451], [IDEA]
ENCR_CAST	6	[RFC2451]
ENCR_BLOWFISH	7	[RFC2451]
ENCR_3IDEA	8	non spécifié
ENCR_DES_IV32	9	non spécifié
ENCR_NULL	11	[RFC2410]
ENCR_AES_CBC	12	[RFC3602]
ENCR_AES_CTR	13	[RFC3686]

Pour le type de transformation 2 (fonction pseudo aléatoire) la liste des identifiants de transformation figure ci-dessous. Les valeurs du tableau qui suit ne sont que celles en cours au moment de la publication de la RFC 4306. D'autres valeurs peuvent avoir été ajoutées depuis ou le seront après la publication du présent document. Le lecteur devrait se référer à [IKEV2IANA] pour avoir les dernières valeurs.

Nom	Numéro	Défini dans
PRF_HMAC_MD5	1	[RFC2104], [RFC1321]
PRF_HMAC_SHA1	2	[RFC2104], [FIPS.180-4.2012]
PRF_HMAC_TIGER	3	non spécifié

Pour le type de transformation 3 (algorithme d'intégrité) la liste des identifiants de transformation figure ci-dessous. Les valeurs du tableau qui suit ne sont que celles en cours au moment de la publication de la RFC 4306. D'autres valeurs peuvent avoir été ajoutées depuis ou le seront après la publication du présent document. Le lecteur devrait se référer à [IKEV2IANA] pour avoir les dernières valeurs.

Nom	Numéro	Défini dans
aucun	0	
AUTH_HMAC_MD5_96	1	[RFC2403]

AUTH_HMAC_SHA1_96	2	[RFC2404]
AUTH_DES_MAC	3	non spécifié
AUTH_KPDK_MD5	4	non spécifié
AUTH_AES_XCBC_96	5	[RFC3566]

Pour le type de transformation 4 (groupe Diffie-Hellman) la liste des identifiants de transformation figure ci-dessous. Les valeurs du tableau qui suit ne sont que celles en cours à la date de publication de RFC 4306. D'autres valeurs peuvent avoir été ajoutées depuis ou le seront après la publication du présent document. Le lecteur devrait se référer à [IKEV2IANA] pour avoir les dernières valeurs.

Nom	**Numéro**	**Défini dans**
Aucun	0	
Groupe MODP à 768 bits	1	Appendice B
Groupe MODP à 1024 bits	2	Appendice B
Groupe MODP à 1536 bits	5	[RFC3526]
Groupe MODP à 2048 bits	14	[RFC3526]
Groupe MODP à 3072 bits	15	[RFC3526]
Groupe MODP à 4096 bits	16	[RFC3526]
Groupe MODP à 6144 bits	17	[RFC3526]
Groupe MODP à 8192 bits	18	[RFC3526]

Bien que ESP et AH n'incluent pas directement l'échange Diffie-Hellman, un groupe Diffie-Hellman PEUT être négocié pour la SA fille. Cela permet que les homologues emploient Diffie-Hellman dans l'échange CREATE_CHILD_SA, fournissant le secret parfait vers l'avant pour les clés de SA fille générées.

Noter que les groupes MODP Diffie-Hellman dont la liste figure ci-dessus n'ont pas besoin d'essais de validité particuliers pour être effectués, mais d'autres types de groupes (groupes de courbes elliptiques, et groupes MODP avec de petits sous groupes) ont besoin d'avoir des essais supplémentaires effectués sur eux pour qu'ils soient utilisés en toute sécurité. Voir plus d'informations dans la [RFC6989] "Essais Diffie-Hellman supplémentaires pour IKEv2".

Pour le type de transformation 5 (Numéros de séquence étendus) la liste des identifiants de transformation définis figure ci-dessous. Les valeurs du tableau qui suit ne sont que celles en cours à la date de publication de RFC 4306. D'autres valeurs peuvent avoir été ajoutées depuis ou le seront après la publication du présent document. Le lecteur devrait se référer à [IKEV2IANA] pour avoir les dernières valeurs.

Nom	**Numéro**
Pas de numéro de séquence étendu	0
Numéro de séquence étendu	1

Noter qu'un initiateur qui prend en charge les ESN va généralement inclure deux transformations ESN, avec les valeurs "0" et "1", dans ses propositions. Une proposition contenant une seule transformation ESN avec la valeur "1" signifie que l'utilisation des numéros de séquence normaux (non étendus) n'est pas acceptable.

De nombreux types de transformation supplémentaires ont été définis depuis la publication de la RFC 4306. Prière de se référer pour les détails au registre de l'IANA "Paramètres de l'échange de clés Internet version 2 (IKEv2)".

3.3.3 Types de transformations valides par protocole

Le nombre et le type des transformations qui accompagnent une charge utile SA dépendent du protocole dans la SA elle-même. Une charge utile SA qui propose l'établissement d'une SA a les types de transformation obligatoires et facultatifs suivants. Une mise en œuvre conforme DOIT comprendre tous les types obligatoires et facultatifs pour chaque protocole qu'elle prend en charge bien qu'il ne soit pas nécessaire qu'elle accepte les propositions avec des suites inacceptables). Une proposition PEUT omettre les types facultatifs si la seule valeur qu'elle va accepter pour eux est AUCUNE.

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

(*) La négociation d'un algorithme d'intégrité est obligatoire pour le format de charge utile Chiffré spécifié dans le présent document. Par exemple, [la RFC5282] spécifie des formats supplémentaires sur la base du chiffrement authentifié, dans lesquels un algorithme d'intégrité séparé n'est pas négocié.

3.3.4 Identifiants de transformation obligatoire

La spécification de suites qui DOIVENT et DEVRAIENT être prises en charge pour l'interopérabilité a été supprimée du présent document parce qu'il est probable qu'elles vont changer plus rapidement que l'évolution du présent document. Au moment de la publication du présent document, la [RFC4307] spécifie ces suites, mais on notera qu'elles pourront être mises à jour à l'avenir, et que d'autres RFC pourront spécifier des ensembles de suites différents.

Une importante leçon apprise de IKEv1 est qu'aucun système ne devrait mettre en œuvre seulement les algorithmes obligatoires et s'attendre à ce qu'ils soient le meilleur choix pour tous les consommateurs.

Il est probable que l'IANA va ajouter des transformations supplémentaires à l'avenir, et que certains usagers pourront 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. Pour la prise en charge de cet objectif, toutes les mises en œuvre de IKEv2 DEVRAIENT inclure une facilité de gestion qui permette la spécification (par un usager ou un administrateur du système) des paramètres Diffie-Hellman (le générateur, les modules, et les longueurs et valeurs d'exposant) pour de nouveaux groupes Diffie-Hellman. Les mises en œuvre DEVRAIENT fournir une interface de gestion à travers laquelle ces paramètres et les identifiants de transformation associés puissent être entrés (par un usager ou un administrateur du système) pour permettre la négociation de tels groupes.

Toutes les mises en œuvre de IKEv2 DOIVENT inclure une facilité de gestion qui permette à un usager ou administrateur de système de spécifier les suites qui lui sont acceptables avec l'utilisation de IKE. À réception d'une charge utile avec un ensemble d'identifiants de transformation, la mise en œuvre DOIT comparer les identifiants de transformation transmis à ceux configurés en local via les commandes de gestion, pour vérifier que la suite proposée est acceptable sur la base de la politique locale. La mise en œuvre DOIT rejeter les propositions de SA qui ne sont pas autorisées par ces contrôles de suites IKE. Noter que les suites cryptographiques qui DOIVENT être mises en œuvre n'ont pas besoin d'être configurées comme acceptables à une politique locale.

3.3.5 Attributs de transformation

Chaque transformation dans une charge utile Association de sécurité peut inclure des attributs qui modifient ou complètent la spécification de la transformation. L'ensemble des attributs valides dépend de la transformation. Actuellement, un seul type d'attribut est défini, l'attribut Longueur de clé est utilisé par certaines transformations de chiffrement avec des clés de longueur variable (voir les détails ci-dessous).

Les attributs sont des paires type/valeur et sont définis ci-dessous. 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.

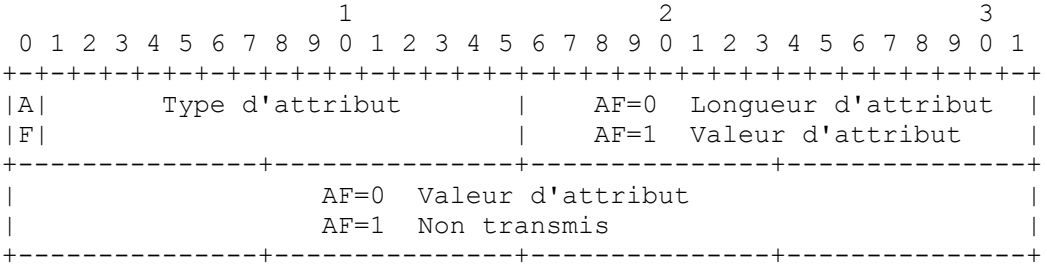


Figure 9 : Attributs de données

- Format d'attribut (AF) (1 bit) - indique si l'attribut de données suit le format Type/Longueur/Valeur (TLV) ou un format raccourci Type/Valeur (TV). Si le bit AF est zéro (0), l'attribut utilise alors le format de TLV ; si le bit AF est un (1), le format TV (avec une valeur de deux octets) est utilisé.

- Type d'attribut (15 bits) - identifiant unique pour chaque type d'attribut (voir ci-dessous).

- Valeur d'attribut (longueur variable) - valeur de l'attribut associé au type d'attribut. Si le bit AF bit est à zéro (0), ce champ a une longueur variable définie par le champ Longueur d'attribut. Si le bit AF est à un (1), la valeur d'attribut a une longueur de 2 octets.

Le seul type d'attribut (Longueur de clé) actuellement défini est de longueur fixe ; la spécification du codage de longueur variable n'est incluse que pour de futures extensions. Les attributs décrits comme de longueur fixe NE DOIVENT PAS être codés en utilisant le codage de longueur variable sauf si cette longueur excède deux octets. Les attributs de longueur variable NE DOIVENT PAS être codés comme de longueur fixe même si leur valeur peut tenir en deux octets Noter que ceci est un

changement par rapport à IKEv1, où une souplesse accrue peut avoir simplifié le composeur de messages mais certainement compliqué l'analyseur.

Les valeurs du tableau suivant ne sont que celles en vigueur lors de la publication de la RFC 4306. D'autres valeurs peuvent avoir été ajoutées depuis ou le seront après la publication du présent document. Le lecteur devrait se référer à [IKEV2IANA] pour avoir les dernières valeurs.

Type d'attribut	**Valeur**	**Format d'attribut**
Longueur de clé (en bits)	14	TV

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.

L'attribut Longueur de clé spécifie pour certaines transformations la longueur de clé en bits (DOIT utiliser l'ordre des octets du réseau) comme suit :

- L'attribut Longueur de clé NE DOIT PAS être utilisé avec des transformations qui utilisent une clé de longueur fixe. Par exemple, cela inclut ENCR_DES, ENCR_IDEA, et toutes les transformations de type 2 (fonction pseudo aléatoire) et de type 3 (algorithme d'intégrité) spécifiées dans le présent document. Il est recommandé que les futures transformations de type 2 ou 3 n'utilisent pas cet attribut.
- Certaines transformations spécifient que l'attribut Longueur de clé DOIT être toujours inclus (omettre l'attribut n'est pas permis et les propositions qui ne le contiennent pas DOIVENT être rejetées). Par exemple, cela inclut ENCR_AES_CBC et ENCR_AES_CTR.
- Certaines transformations permettent des clés de longueur variable, mais spécifient aussi une longueur de clé par défaut si l'attribut n'est pas inclus. Par exemple, ces transformations incluent ENCR_RC5 et ENCR_BLOWFISH.

Note de mise en œuvre : pour plus d'interopérabilité et pour prendre en charge la mise à niveau indépendante des points d'extrémité, les mises en œuvre de ce protocole DEVRAIENT accepter des valeurs qu'elles estiment fournir plus de sécurité. Par exemple, si un homologue est configuré à accepter un chiffrement de longueur variable avec une longueur de clé de X bits et si ce chiffrement est offert avec une longueur de clé supérieure, la mise en œuvre DEVRAIT accepter l'offre si elle prend en charge l'utilisation de plus longues clés.

La prise en charge de cette capacité permet à un répondant 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". Cependant, comme l'attribut est toujours retourné inchangé (voir au paragraphe suivant) un initiateur qui veut accepter plusieurs longueurs de clé doit inclure plusieurs transformations avec le même type de transformation, chacune avec un attribut Longueur de clé différent.

3.3.6 Négociation d'attribut

Durant la négociation d'association de sécurité, les initiateurs présentent des offres aux répondants. Les répondants DOIVENT choisir un seul ensemble complet de paramètres dans l'offre (ou rejeter toutes les offres si aucune n'est acceptable). Si il y a plusieurs propositions, le répondant DOIT choisir une seule proposition. Si la proposition choisie a plusieurs transformations du même type, le répondant DOIT en choisir une seule. Tous les attributs d'une transformation choisie DOIVENT être retournés non modifiés. L'initiateur d'un échange DOIT vérifier que l'offre acceptée est cohérente avec une de ses propositions, et sinon il DOIT terminer l'échange.

Si le répondant reçoit une proposition qui contient un type de transformation qu'il ne comprend pas, ou une proposition qui n'a pas un type de transformation obligatoire, il DOIT considérer cette proposition comme inacceptable ; cependant, les autres propositions dans la même charge utile SA sont traitées comme d'habitude. De même, si le répondant reçoit une transformation qu'il ne comprend pas, ou qui contient un attribut de transformation qu'il ne comprend pas, il DOIT considérer que cette transformation est inacceptable ; les autres transformations avec le même type de transformation sont traitées comme d'habitude. Cela permet que de nouveaux types de transformation et attributs de transformation soient définis à l'avenir.

Négocier des groupes Diffie-Hellman présente des défis particuliers. L'offre de SA inclut des propositions d'attributs 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 probablement accepter et inclure un KE correspondant à ce groupe. Si le répondant choisit une proposition utilisant un groupe Diffie-Hellman différent (autre que AUCUN) le répondant va indiquer le groupe correct dans la réponse et l'initiateur DEVRAIT prendre un élément de ce groupe pour sa valeur de KE quand il va réessayer le premier message. Il DEVRAIT, cependant, continuer de proposer son ensemble complet de groupes pris en charge afin d'empêcher une attaque en dégradation par interposition. Si une des propositions offertes est pour le groupe Diffie-Hellman AUCUN, et si le répondant choisit ce groupe Diffie-Hellman, il DOIT alors ignorer la charge utile KE de l'initiateur et omettre la charge utile KE dans sa réponse.

3.4 Charge utile Échange de clé

La charge utile Échange de clé, notée KE dans le présent document, est utilisée pour échanger les numéros publics Diffie-Hellman au titre d'un échange de clé Diffie-Hellman. La charge utile Échange de clé consiste en l'en-tête générique de charge utile IKE suivi par la valeur publique Diffie-Hellman elle même.

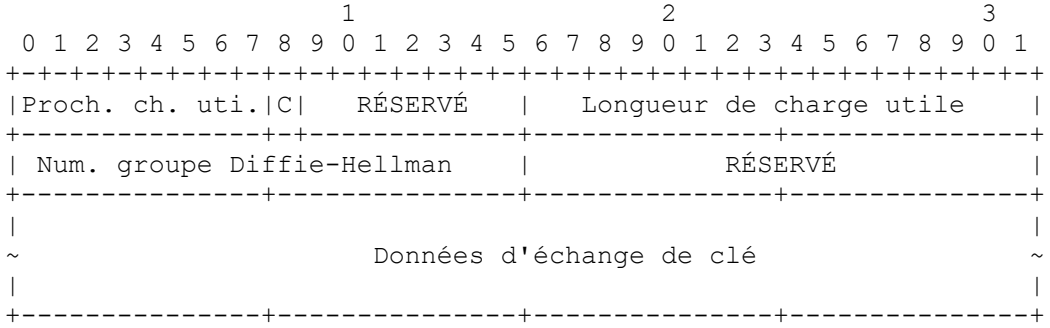


Figure 10 : Format de charge utile Échange de clé

La charge utile Échange de clé 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 pour les groupes MODP DOIT être égale à la longueur du module premier sur lequel l'exponentiation a été effectuée, en ajoutant des bits zéro devant la valeur si nécessaire.

Le numéro de groupe Diffie-Hellman identifie le groupe Diffie-Hellman dans lequel les données d'échange de clé ont été calculées (voir au paragraphe 3.3.2). Ce numéro de groupe Diffie-Hellman DOIT correspondre au groupe Diffie-Hellman spécifié dans une proposition dans la charge utile SA envoyée dans le même message, et DEVRAIT correspondre au groupe Diffie-Hellman du premier groupe de la première proposition, si elle existe. Si aucune des propositions de cette charge utile SA ne spécifie de groupe Diffie-Hellman, la charge utile KE NE DOIT PAS être présente. Si la proposition choisie utilise un groupe Diffie-Hellman différent (autre que AUCUN) le message DOIT être rejeté avec une charge utile Notify de type INVALID_KE_PAYLOAD. Voir aussi les paragraphes 1.2 et 2.7.

Le type de charge utile pour la charge utile Échange de clé est trente quatre (34).

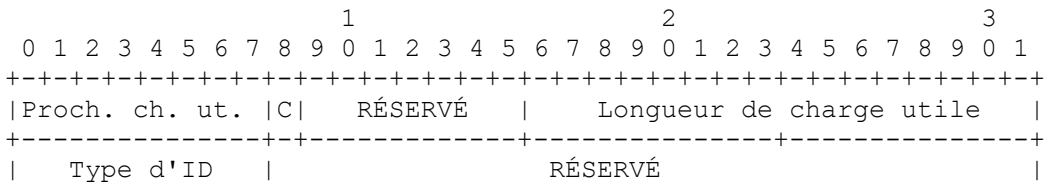
3.5 Charges utiles Identification

Les charges utiles Identification, notées IDsi et IDr dans le présent document, permettent aux homologues d'affirmer mutuellement leur identité. Cette identité peut être utilisée pour une recherche de politique, mais n'a pas nécessairement à correspondre à quelque chose qui serait dans la charge utile CERT ; les deux champs peuvent être utilisés par une mise en œuvre pour prendre des décisions de contrôle d'accès. Quand on utilise les types d'identité ID_IPV4_ADDR/ID_IPV6_ADDR dans les charges utiles IDi/IDr, IKEv2 n'exige pas que cette adresse corresponde à l'adresse dans l'en-tête IP des paquets IKEv2, ou à quelque chose des charges utiles TSi/TSr. Le contenu de IDi/IDr est utilisé simplement pour aller chercher les données de politique et d'authentification relatives à l'autre partie.

Note : dans IKEv1, deux charges utiles Identification étaient utilisées dans chaque direction pour contenir les informations de sélecteur de trafic (TS) sur les données qui traversent la SA. Dans IKEv2, ces informations sont portées dans les charges utiles TS (voir au paragraphe 3.13).

La base de données d'authentification d'homologue (PAD, *Peer Authorization Database*) décrite dans la [RFC4301] décrit l'utilisation de la charge utile ID dans IKEv2 et fournit un modèle formel pour la liaison d'une identité à une politique, en plus des services qui traitent plus spécifiquement des détails de la mise en application de la politique. La PAD est destinée à fournir un lien entre la SPD et la gestion de l'association de sécurité IKE. Voir les détails au paragraphe 4.4.3 de la [RFC4301].

La charge utile Identification consiste en l'en-tête générique de charge utile IKE suivi par les champs d'identification comme suit :



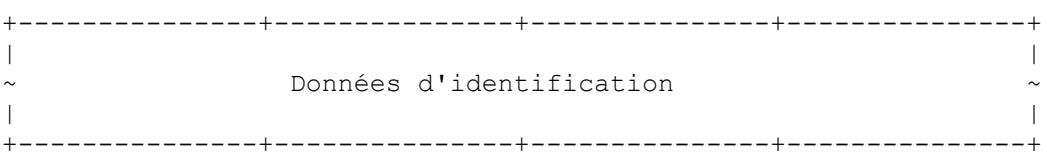


Figure 11 : Format de charge utile Identification

- Type d'ID (1 octet) - spécifie le type d'identification utilisé.
- 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 charge utile d'identification.

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

Le tableau qui suit donne la listes des sémantiques allouées pour le champ Type d'identification. Les valeurs du tableau qui suit ne sont que celles en cours à la date de publication de RFC 4306. D'autres valeurs peuvent avoir été ajoutées depuis ou le seront après la publication du présent document. Le lecteur devrait se référer à [IKEV2IANA] pour avoir les dernières valeurs.

Type d'identification	Valeur	Commentaire
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 ID_FQDN est "example.com". La chaîne NE DOIT PAS contenir de terminateur (par exemple, NUL, CR, etc.). Tous les caractères de ID_FQDN sont ASCII ; pour un nom de domaine internationalisé, la syntaxe est définie dans la [RFC5890], par exemple "xn--tmonesimerkki-bfbb.exemple.net".
ID_RFC822_ADDR	3	Chaîne d'adresse pleinement qualifiée de messagerie électronique de la RFC 822. Un exemple de ID_RFC822_ADDR est "jsmith@exemple.com". La chaîne NE DOIT PAS contenir de terminateur. À cause de la [RFC6532], il serait bon que les mises en œuvre traitent ce champ comme codé en UTF-8 plutôt que pur ASCII.
ID_IPV6_ADDR	5	Une seule adresse IPv6 de seize (16) octets.
ID_DER_ASN1_DN	9	Codage binaire selon les règles de codage distinctif (DER) d'un nom distinctif X.500 en ASN.1 [RFC5280].
ID_DER_ASN1_GN	10	Codage binaire DER d'un nom général X.509 [RFC5280] en ASN.1.
ID_KEY_ID	11	Flux d'octets opaque qui peut être utilisé pour passer des informations spécifiques du fabricant nécessaires pour faire certains types d'identification non normalisés.

Deux mises en œuvre ne vont interopérer que si chacune peut générer un type d'ID acceptable à l'autre. Pour assurer un maximum d'interopérabilité, les mises en œuvre DOIVENT être configurables à envoyer au moins un de ID_IPV4_ADDR, ID_FQDN, ID_RFC822_ADDR, ou ID_KEY_ID, et DOIVENT être configurables à accepter ces quatre types. Les mises en œuvre DEVRAIENT être capables de générer et accepter tous ces types. Les mises en œuvre à capacité IPv6 DOIVENT de plus être configurables à accepter ID_IPV6_ADDR. Les mises en œuvre seulement IPv6 PEUVENT être configurables à n'envoyer que ID_IPV6_ADDR au lieu de ID_IPV4_ADDR pour les adresses IP.

EAP [RFC3748] ne rend pas obligatoire l'utilisation d'un type d'identifiant particulier, mais EAP est souvent utilisé avec des identifiants d'accès réseau (NAI, *Network Access Identifier*) définis dans la [RFC4282]. Bien que les NAI ressemblent un peu à des adresses de messagerie électronique (par exemple, "joe@exemple.com") la syntaxe n'est pas exactement la même que celle de l'adresse de messagerie de la [RFC5322]. Pour les NAI qui incluent le composant de domaine, le type d'identification ID_RFC822_ADDR DEVRAIT être utilisé. Les mises en œuvre de répondant ne devraient pas tenter de vérifier que le contenu se conforme bien à la syntaxe exacte donnée dans la [RFC5322], mais devrait plutôt accepter tout NAI dont l'aspect est raisonnable. Pour les NAI qui n'incluent pas le composant de domaine, le type d'identification ID_KEY_ID DEVRAIT être utilisé. Voir plus d'informations dans la [RFC4945] "Profil Internet de PKI de sécurité IP de IKEv1/ISAKMP, IKEv2, et PKIX" sur les charges utiles Identification correspondantes et le contenu des certificats PKIX.

3.6 Charge utile Certificat

La charge utile Certificat, notée CERT dans le présent document, donne le moyen de transporter les certificats ou autres informations relatives à l'authentification via IKE. Les charges utiles Certificat DEVRAIENT être incluses dans un échange si des certificats sont disponible à l'envoyeur. Les formats Hash et URL des charges utiles Certificat devraient être utilisés dans le cas où l'homologue a indiqué sa capacité à restituer cette information à partir d'ailleurs en utilisant une charge utile Notify

HTTP_CERT_LOOKUP_SUPPORTED. Noter que le terme "charge utile Certificat" est un 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 Certificat est définie comme suit :

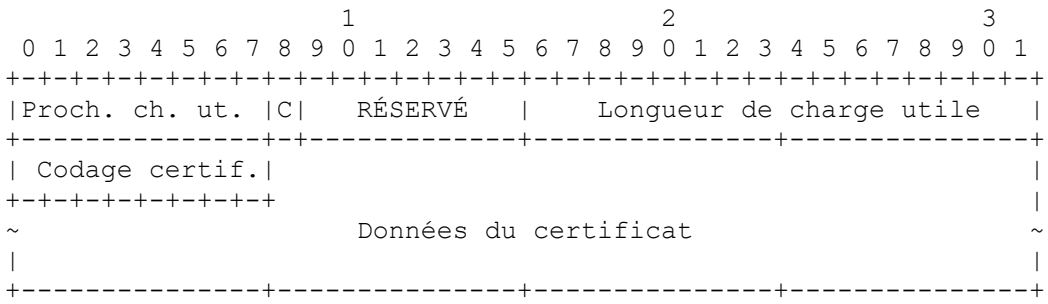


Figure 12 : Format de charge utile Certificat

o Codage de certificat (1 octet) - ce champ indique le type de certificat ou les informations en rapport avec le certificat contenues dans le champ Données de certificat. Les valeurs du tableau qui suit ne sont que celles en cours à la date de publication de RFC 4306. D'autres valeurs peuvent avoir été ajoutées depuis ou le seront après la publication du présent document. Le lecteur devrait se référer à [IKEV2IANA] pour avoir les dernières valeurs.

Codage	Nom du certificat	Commentaire
1	certificat X.509 enveloppé dans PKCS n°7	non spécifiée
2	certificat PGP	non spécifiée
3	clé signée DNS	non spécifiée
4	certificat X.509 - Signature	4
6	jeton Kerberos	non spécifié
7	liste de révocation de certificat (CRL)	
8	liste de révocation d'autorité (ARL)	non spécifiée
9	certificat SPKI	non spécifiée
10	certificat X.509 - Attribut	non spécifiée
11	clé RSA brute	déconseillé
12	hachage et URL de certificat X.509	
13	hachage et URL de bouquet X.509	

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 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 dans le présent document sont :

o "Certificat X.509 - Signature" contient un certificat X.509 codé en DER dont la clé publique est utilisée pour valider la charge utile AUTH de l'envoyeur. Noter qu'avec ce codage, si une chaîne de certificats doit être envoyée, plusieurs charges utiles CERT sont utilisées, dont seule la première contient la clé publique utilisée pour valider la charge utile AUTH de l'envoyeur.

o "Liste de révocation de certificats" contient une liste de révocation de certificats X.509 codée en DER.

o Les codages Hachage et URL permettent aux messages IKE de rester courts en remplaçant les longues structures de données par un hachage SHA-1 de 20 octets (voir [FIPS.180-4.2012]) de la valeur remplacée suivie par un URL de longueur variable qui se résout en la structure de données elle-même codée en DER. Cela améliore l'efficacité quand les points d'extrémité ont des 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 grands pour exiger la fragmentation IP [DOS].

Le type "Hachage et URL d'un bouquet" utilise la définition ASN.1 suivante pour le bouquet X.509 :

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

DÉFINITIONS DES ÉTIQUETTES EXPLICITES ::=
```

```
DÉBUT

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

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

  CertificateBundle ::= SEQUENCE DE CertificateOrCRL

FIN
```

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

Les mises en œuvre DOIVENT prendre en charge le schéma "http:" pour la recherche de hachage-et-URL. Le comportement des autres schémas d'URL [RFC3986] n'est actuellement pas spécifié, et de tels schémas NE DEVRAIENT PAS être utilisés en l'absence d'un document qui les spécifie.

3.7 Charge utile Demande de certificat

La charge utile Demande de certificat, notée CERTREQ dans le présent document, fournit un moyen pour demander les certificats préférés via IKE et peut apparaître dans la réponse IKE_INIT_SA et/ou la demande IKE_AUTH. Les charges utiles Demande de certificat PEUVENT être incluses dans un échange quand l'envoyeur a besoin d'obtenir le certificat du receveur.

La charge utile Demande de certificat est définie comme suit :

Figure 13 : Format de charge utile Demande de certificat

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

- 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 l'autorité de certification est une liste de hachages SHA-1 des clés publiques des autorités de certification (CA, *Certification Authorities*) de confiance. Chacune est codée comme hachage SHA-1 de l'élément Informations de clé publique sujette (voir au paragraphe 4.1.2.7 de la [RFC5280]) provenant de chaque certificat d'ancre de confiance. Les hachages de 20 octets sont enchaînés et inclus sans autre formatage.

Le contenu du champ Autorité de certification est défini seulement pour les certificats X.509, qui sont les types 4, 12, et 13. D'autres valeurs NE DEVRAIENT PAS être utilisées tant que des spécifications sur la voie de la normalisation qui spécifient leur utilisation ne sont pas publiées.

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

La charge utile Demande de certificat est traitée par l'inspection du champ Codage de certificat pour déterminer si le processeur a des certificats de ce type. Si il en a, le champ Autorité de certification est inspecté pour déterminer si le processeur a des certificats qui peuvent être validés jusqu'à celui des autorités de certification spécifiées. Ce peut être une chaîne de certificats.

Si il existe un certificat d'entité d'extrémité qui satisfait aux critères spécifiés dans le CERTREQ, un certificat ou une chaîne de certificats DEVRAIT être renvoyé au demandeur de certificat si le receveur de la CERTREQ :
- o est configuré à utiliser l'authentification de certificat,
- o est autorisé à envoyer une charge utile CERT,
- o a une politique de confrontation des CA de confiance qui gouverne la négociation en cours, et
- o a au moins un chaînage de certificat d'entité d'extrémité à durée limitée et approprié à l'usage à une CA fournie dans la CERTREQ.

La vérification de la révocation de certificat doit être considérée durant le processus de chaînage utilisé pour choisir un certificat. Noter que même si les deux homologues sont configurés à utiliser deux CA différentes, des relations de certification croisée devraient être prises en charge par une logique de sélection appropriée.

L'intention n'est pas d'empêcher la communication par la stricte adhérence du choix d'un certificat sur la base de la CERTREQ, quand un certificat de remplacement pourrait être choisi par l'envoyeur qui permettrait quand même au receveur de réussir à le valider et lui faire confiance sur la base de la certification croisée, des CRL, ou d'autres moyens configurés hors bande. Donc, le traitement d'une CERTREQ devrait être vu comme une suggestion d'un certificat à choisir, pas de le rendre obligatoire. Si il n'existe pas de certificat, la CERTREQ est alors ignorée. 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 la CERTREQ, mais une autre peut être acceptable (peut-être après une invite à un opérateur humain).

La notification HTTP_CERT_LOOKUP_SUPPORTED PEUT être incluse dans tout message qui peut inclure une charge utile CERTREQ et indique que l'envoyeur est capable de chercher des certificats sur la base d'un URL fondé sur HTTP (et qui va probablement préférer recevoir les spécifications de certificat dans ce format).

3.8 Charge utile Authentification

La charge utile Authentification, notée AUTH dans le présent document, contient des données utilisées pour les besoins de l'authentification. La syntaxe des données d'authentification varie selon la méthode d'authentification comme spécifié ci-dessous. La charge utile Authentification est définie comme suit :

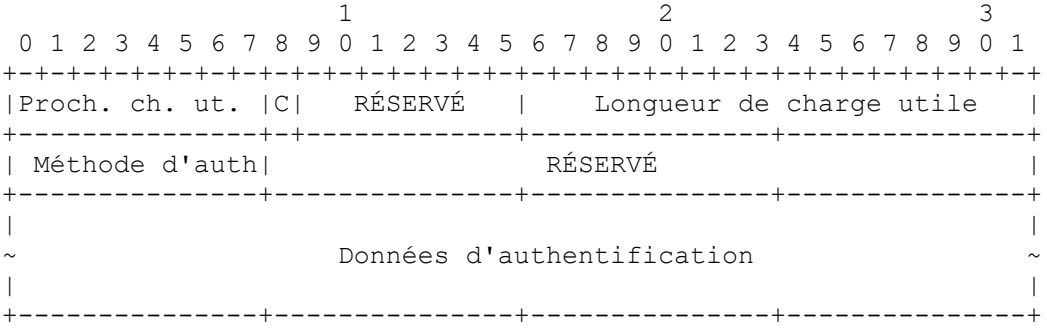


Figure 14 : Format de charge utile Authentification

- o Méthode d'authentification (1 octet) - spécifie la méthode d'authentification utilisée. La liste des types de signatures figure ici. Les valeurs du tableau qui suit ne sont que celles en cours à la date de publication de RFC 4306. D'autres valeurs peuvent avoir été ajoutées depuis ou le seront après la publication du présent document. Le lecteur devrait se référer à [IKEV2IANA] pour avoir les dernières valeurs.

Valeur	Mécanisme
1	Signature numérique RSA : calculée comme spécifié au paragraphe 2.15 en utilisant une clé privée RSA avec le schéma de signature RSASSA-PKCS1-v1_5 spécifié dans la [RFC3447] (les développeurs devraient noter que IKEv1 utilisait une méthode différente pour les signatures RSA). Pour promouvoir l'interopérabilité, les mises en œuvre qui prennent en charge ce type DEVRAIENT prendre en charge les signatures qui utilisent SHA-1 comme fonction de hachage et DEVRAIENT utiliser SHA-1 comme fonction de hachage par défaut quand elles génèrent des signatures. Les mises en œuvre peuvent utiliser les certificats reçus d'un certain homologue comme un conseil pour choisir une fonction de hachage mutuellement comprise pour la signature de la charge utile AUTH. Noter cependant que l'algorithme de hachage utilisé dans la signature de la charge utile AUTH n'a pas à être le même que celui ou ceux utilisés dans le ou les certificats.
2	Code d'intégrité de message à clé partagée : calculé comme spécifié au paragraphe 2.15 en utilisant la clé partagée associée à l'identité dans la charge utile ID et la PRF négociée.
3	Signature numérique DSS : calculée comme spécifié au paragraphe 2.15 en utilisant une clé privée DSS (voir [DSS]) sur un hachage SHA-1.

- RÉSERVÉ - DOIT être envoyé à zéro ; DOIT être ignoré à réception.

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

Le type de charge utile pour la charge utile Authentification est trente neuf (39).

3.9 Charge utile Nom occasionnel

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

La charge utile Nom occasionnel est définie comme suit :

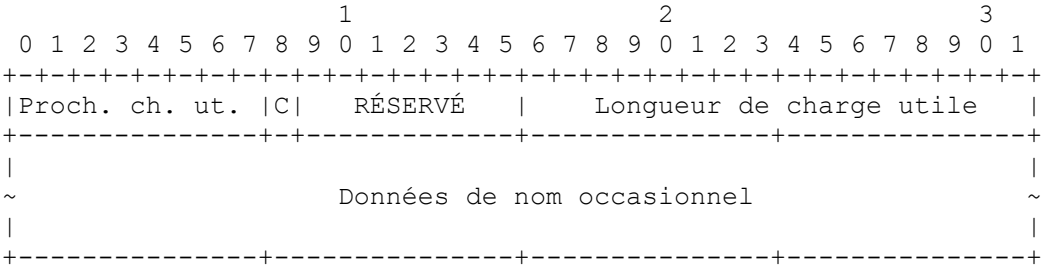


Figure 15 : Format de charge utile Nom occasionnel

- Données de nom occasionnel (longueur variable) - contient les données aléatoires générées par l'entité émettrice.

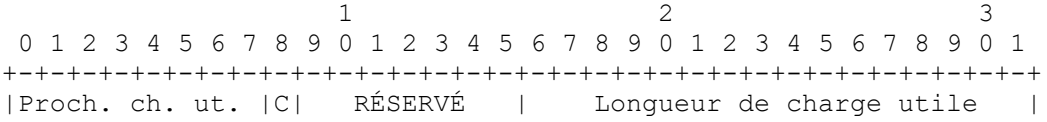
Le type de charge utile pour la charge utile Nom occasionnel est quarante (40).

La taille des données de nom occasionnel DOIT être entre 16 et 256 octets, inclus. Les valeurs de nom occasionnel NE DOIVENT PAS être réutilisées.

3.10 Charge utile Notifie

La charge utile Notifie, notée N dans le présent document, est utilisée pour transmettre des données d'information, telles que des conditions d'erreur et des transitions d'état, à un homologue IKE. Une charge utile Notifie peut apparaître dans un message de réponse (généralement qui spécifie pourquoi une demande a été rejetée) dans un échange INFORMATIONAL (pour rapporter une erreur qui n'est pas dans 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 Notifie est définie comme suit :



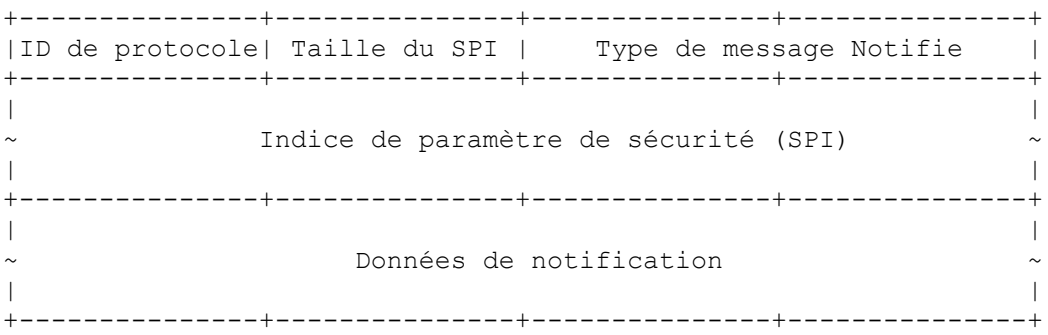


Figure 16 : Format de charge utile Notifie

- Identifiant de protocole (1 octet) - si cette notification concerne une SA existante dont le SPI est donné dans le champ SPI, ce champ indique le type de cette SA. Pour les notifications concernant des SA filles, ce champ DOIT contenir soit (2) pour indiquer AH, soit (3) pour indiquer ESP. Dans les notifications définies dans le présent document, le SPI n'est inclus qu'avec INVALID_SELECTORS, REKEY_SA, et CHILD_SA_NOT_FOUND. Si le champ SPI est vide, ce champ DOIT être envoyé à zéro et DOIT être ignoré à réception.

- Taille de SPI (1 octet) - longueur en octets du SPI comme défini par l'identifiant de protocole IPsec ou zéro si aucun SPI n'est applicable. Pour une notification concernant la SA IKE, Taille de SPI DOIT être zéro et le champ doit être vide.

- Type de message Notifie (2 octets) - spécifie le type du message de notification.

- SPI (longueur variable) - indice de paramètre de sécurité.

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

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

3.10.1 Types de message Notifie

Les informations des notifications peuvent être des messages d'erreur qui spécifient pourquoi une SA n'a pas pu être établie. Elles peuvent aussi être des données d'état qu'un processus de gestion d'une base de données de SA souhaite communiquer avec un processus d'homologue.

Le tableau ci-dessous fait la liste des messages de notification et de leurs valeurs correspondantes. Le nombre des états d'erreur différents a été largement réduit par rapport à IKEv1 à la fois pour simplifier et pour éviter de donner des informations de configurations aux sondeurs.

Les types dans la gamme de 0 à 16383 sont destinés à rapporter des erreurs. Une mise en œuvre qui reçoit une charge utile Notifie avec un de ces types qu'il ne reconnaît pas dans une réponse DOIT supposer que la demande correspondante a entièrement échoué. Les types d'erreur non reconnus dans une demande et les types d'état dans une demande ou réponse DOIVENT être ignorés, et ils devraient être enregistrés dans le journal d'événements.

Des charges utiles Notifie 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 les capacités, et au titre de la négociation de SA, sont utilisées pour négocier les paramètres non cryptographiques.

On trouvera plus d'informations sur le traitement des erreurs au paragraphe 2.21.

Les valeurs du tableau suivant ne sont que celles en cours à la date de publication de la RFC 4306, plus deux types d'erreur ajoutés dans le présent document. D'autres valeurs peuvent avoir été ajoutées depuis ou le seront après la publication du présent document. Le lecteur devrait se référer à [IKEV2IANA] pour avoir les dernières valeurs.

Types d'erreur de messages Notifie	Valeur	Commentaire
UNSUPPORTED_CRITICAL_PAYLOAD	1	voir au paragraphe 2.5.
INVALID_IKE_SPI	4	voir au paragraphe 2.21.
INVALID_MAJOR_VERSION	5	voir au paragraphe 2.5.
INVALID_SYNTAX	7	Indique que le message IKE reçu était invalide à cause d'un type, longueur, ou valeur hors gamme ou parce que la demande a été

		rejetée pour des raisons de politique. Pour éviter une attaque de DoS utilisant des messages contrefaits, cet état ne peut être retourné que pour et dans un paquet chiffré si l'identifiant de message et la somme de contrôle cryptographique sont valides. Pour éviter des fuites d'information à quelqu'un qui sonde un nœud, cet état DOIT être envoyé en réponse à toute erreur non couverte par un des autres types d'état. Pour aider au débogage, des informations d'erreur plus détaillées devraient être écrites à la console ou journal d'événements.
INVALID_MESSAGE_ID	9	voir au paragraphe 2.3.
INVALID_SPI	11	voir au paragraphe 1.5.
No_PROPOSAL_CHOSEN	14	Aucune des suites de chiffrement proposées n'était acceptable. Ceci peut être envoyé dans tous les cas où les propositions offertes (incluant, mais non limité aux valeurs de charge utile SA, notifie USE_TRANSPORT_MODE, notifie IPCOMP_SUPPORTED) ne sont pas acceptables pour le répondant. Ceci peut aussi être utilisé comme erreur "générique" de SA fille quand la SA fille ne peut pas être créée pour quelque autre raison. Voir aussi le paragraphe 2.7.
INVALID_KE_PAYLOAD	17	voir les paragraphes 1.2 et 1.3.
AUTHENTICATION_FAILED	24	Envoyé dans la réponse à un message IKE_AUTH quand, pour une raison quelconque, l'authentification a échoué. Il n'y a pas de données associées. Voir aussi le paragraphe 2.21.2.
SINGLE_PAIR_REQUIRED	34	voir au paragraphe 2.9.
NO_ADDITIONAL_SAS	35	voir au paragraphe 1.3.
INTERNAL_ADDRESS_FAILURE	36	voir au paragraphe 3.15.4.
FAILED_CP_REQUIRED	37	voir au paragraphe 2.19.
TS_UNACCEPTABLE	38	voir au paragraphe 2.9.
INVALID_SELECTORS	39	PEUT être envoyé dans un échange INFORMATIONAL IKE quand un nœud reçoit un paquet ESP ou AH dont les sélecteurs ne correspondent pas à ceux de la SA sur laquelle il a été livré (et cela a causé l'élimination 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 à correspondre au SPI de la SA fille.
TEMPORARY_FAILURE	43	voir au paragraphe 2.25.
CHILD_SA_NOT_FOUND	44	voir au paragraphe 2.25.

Types d'état des messages Notifie	**Valeur**	**Commentaire**
INITIAL_CONTACT	16384	voir au paragraphe 2.4.
SET_WINDOW_SIZE	16385	voir au paragraphe 2.3.
ADDITIONAL_TS_POSSIBLE	16386	voir au paragraphe 2.9.
IPCOMP_SUPPORTED	16387	voir au paragraphe 2.22.
NAT_DETECTION_SOURCE_IP	16388	voir au paragraphe 2.23.
NAT_DETECTION_DESTINATION_IP	16389	voir au paragraphe 2.23.
COOKIE	16390	voir au paragraphe 2.6.
USE_TRANSPORT_MODE	16391	voir au paragraphe 1.3.1.
HTTP_CERT_LOOKUP_SUPPORTED	16392	voir au paragraphe 3.6.
REKEY_SA	16393	voir au paragraphe 1.3.3.
ESP_TFC_PADDING_NOT_SUPPORTED	16394	voir au paragraphe 1.3.1.
NON_FIRST_FRAGMENTS_ALSO	16395	voir au paragraphe 1.3.1.

3.11 Charge utile Supprime

La charge utile Supprime *(Delete)* notée D dans le présent document, 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 Supprime. Il est possible d'envoyer plusieurs SPI dans une charge utile Supprime, cependant, chaque SPI DOIT être pour le même protocole. Le mélange d'identifiants de protocole NE DOIT PAS être effectué dans la charge utile Supprime. Il est cependant permis d'inclure plusieurs charges utiles Supprime dans un seul échange INFORMATIONAL où chaque charge utile Supprime fait la liste des SPI pour un protocole différent.

La suppression de la SA IKE est indiquée par un identifiant de protocole de 1 (IKE) mais pas de SPI. La suppression d'une SA fille, comme ESP ou AH, va contenir l'identifiant de protocole IPsec de ce protocole (2 pour AH, 3 pour ESP) et le SPI est celui que le point d'extrémité envoyeur attendrait dans les paquets ESP ou AH entrants.

La charge utile Supprime est définie comme suit :

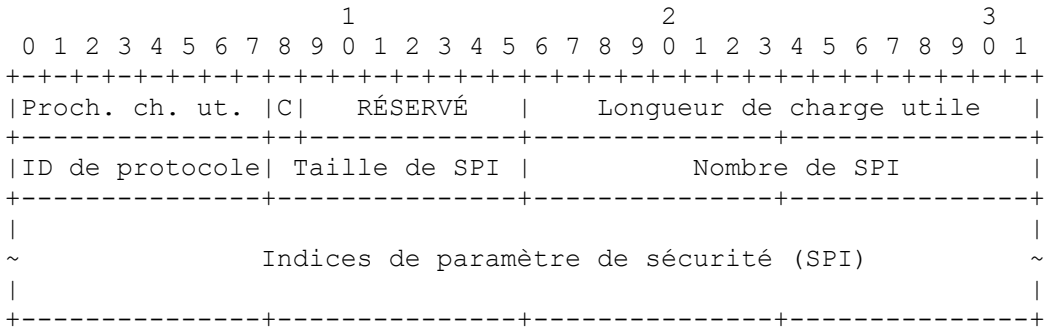


Figure 17 : Format de charge utile Supprime

- Identifiant de protocole (1 octet) - doit être 1 pour une SA IKE, 2 pour AH, ou 3 pour ESP.
- Taille de SPI (1 octet) - longueur en octets du SPI comme défini par l'identifiant de protocole. Il DOIT être zéro pour IKE (le SPI est dans l'en-tête du message) ou quatre pour AH et ESP.
- Nombre de SPI (2 octets, entier non signé) - nombre de SPI contenus dans la charge utile Supprime. La taille de chaque SPI est définie par le champ Taille de SPI.
- Indices de paramètres de sécurité (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 Supprime est quarante deux (42).

3.12 Charge utile Identifiant de fabricant

La charge utile Identifiant de fabricant, notée V dans le présent document, contient une constante définie par le fabricant. La constante est utilisée par les fabricants pour identifier et reconnaître les instances distantes de leurs mises en œuvre. Ce mécanisme permet à un fabricant d'expérimenter de nouvelles caractéristiques tout en conservant la rétro compatibilité.
Une charge utile Identifiant de fabricant PEUT annoncer que l'envoyeur est capable d'accepter certaines extensions au protocole, ou elle PEUT simplement identifier la mise en œuvre comme une aide au débogage. Une charge utile Identifiant de fabricant NE DOIT PAS changer l'interprétation des informations définies dans cette spécification (c'est-à-dire, le bit critique DOIT être réglé à 0). Plusieurs charges utiles Identifiant de fabricant PEUVENT être envoyées. Une mise en œuvre n'est pas obligée d'envoyer du tout de charge utile Identifiant de fabricant.
Une charge utile Identifiant de fabricant peut être envoyée au titre de tout message. La réception d'une charge utile Identifiant de fabricant familière permet à une mise en œuvre de faire usage de numéros d'utilisation privée décrits tout au long du présent document, comme des charges utiles privées, des échanges privés, des notifications privées, etc. Les identifiants de fabricant non familiers DOIVENT être ignorés.
Les rédacteurs de documents qui souhaitent étendre ce protocole DOIVENT définir une charge utile Identifiant de fabricant pour annoncer la capacité de mettre en œuvre l'extension dans le document. Il est prévu que les documents qui sont acceptés et sont normalisés recevront des "numéros magiques" dans la gamme "Utilisation future" de l'IANA, et l'exigence d'utiliser un identifiant de fabricant disparaîtra.

Les champs de la charge utile Identifiant de fabricant sont définis comme suit :

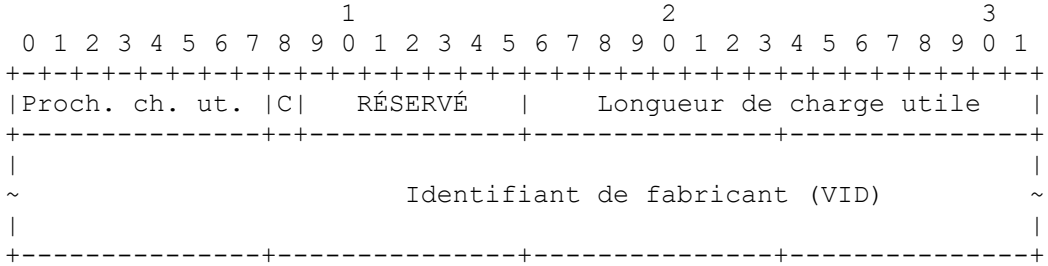


Figure 18 : Format de charge utile Identifiant de fabricant

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 un nom de société, un nom de personne, ou des informations de ce genre. Si on veut faire le malin, on peut inclure la latitude, la longitude, et l'heure à laquelle l'identifiant a été choisi et des éléments aléatoires. Un résumé de message d'une longue chaîne unique est préférable à la longue chaîne unique elle-même.

Le type de charge utile pour la charge utile Identifiant de fabricant est quarante trois (43).

3.13 Charge utile Sélecteur de trafic

La charge utile Sélecteur de trafic, notée TS dans le présent document, permet aux homologues d'identifier les flux de paquets à traiter par les services de sécurité IPsec. La charge utile Sélecteur de trafic consiste en l'en-tête générique de charge utile IKE suivi par les sélecteurs de trafic individuels comme suit :

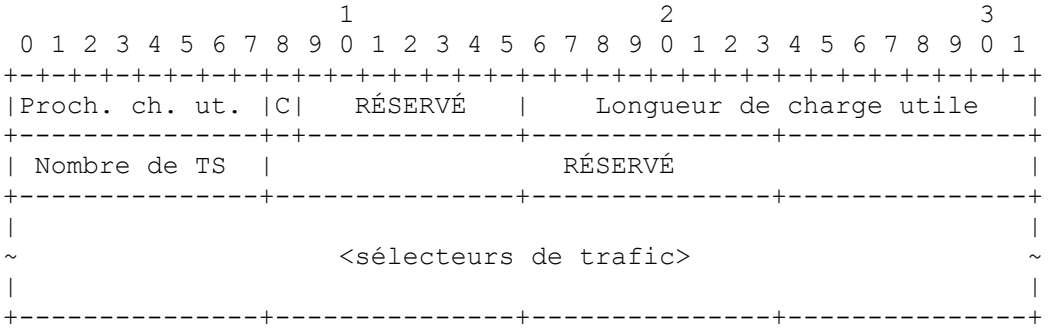


Figure 19 : Format de charge utile Sélecteur de trafic

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 Sélecteur de trafic inclut l'en-tête TS et tous les sélecteurs de trafic.

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

Il n'est pas exigé que TSi et TSr contiennent le même nombre de sélecteurs de trafic individuels. Donc, ils sont interprétés comme suit : un paquet correspond à un certain TSi/TSr si il correspond au moins à un des sélecteurs individuels dans TSi, et au moins un des sélecteurs individuels dans TSr.

Par exemple, les sélecteurs de trafic suivants :

TSi = ((17, 100, 198.51.100.66-198.51.100.66), (17, 200, 198.51.100.66-198.51.100.66))
TSr = ((17, 300, 0.0.0.0-255.255.255.255), (17, 400, 0.0.0.0-255.255.255.255))

vont correspondre aux paquets UDP de 198.51.100.66 à n'importe où, avec une des quatre combinaisons d'accès de source/destination (100,300), (100,400), (200,300), et (200, 400).

Donc, certains types de politiques peuvent exiger plusieurs paires de SA filles. Par exemple, une politique correspondant seulement aux accès de source/destination (100,300) et (200,400), mais pas aux deux autres combinaisons,ne peut pas être négociée comme une seule paire de SA filles.

3.13.1 Sélecteur de trafic

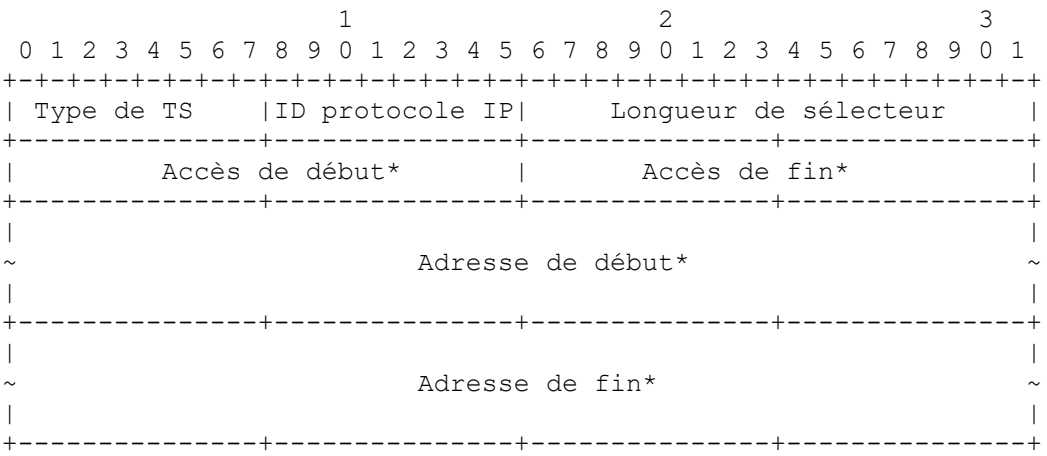


Figure 20 : Sélecteur de trafic

*Note : tous les champs autres que Type de TS et Longueur de sélecteur dépendent du Type de TS. Les champs montrés sont pour les Types de TS 7 et 8, les 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 identifiant de protocole IP associé (comme UDP, TCP, et ICMP). Une valeur de zéro signifie que l'identifiant de protocole n'est pas pertinent pour ce sélecteur de trafic -- la SA peut porter tous les protocoles.
- Longueur de sélecteur (2 octets, entier non signé) - spécifie la longueur de cette sous structure de sélecteur de trafic incluant l'en-tête.
- Accès de début (2 octets, entier non signé) - valeur qui spécifié le plus petit numéro d'accès permis par ce sélecteur de trafic. Pour les protocoles pour lesquels l'accès est indéfini (incluant le protocole 0) ou si tous les accès sont permis, ce champ DOIT être zéro. Les valeurs de type et codes ICMP et ICMPv6, ainsi que les valeurs de type d'en-tête de mobilité (MH, *Mobility Header*) IP mobile version 6 (MIPv6) sont représentées dans ce champ comme spécifié au paragraphe 4.4.1.1 de la [RFC4301]. Les valeurs de type et code ICMP sont traitées comme un seul numéro d'accès entier de 16 bits, avec le type dans les huit bits de poids fort et le code dans les huit bits de moindre poids. Les valeurs de type MH MIPv6 sont traitées comme un seul numéro d'accès entier de 16 bits, avec le type dans les huit bits de poids fort et les huit bits de moindre poids réglés à zéro.
- Accès de fin (2 octets, entier non signé) - valeur qui spécifie le plus grand numéro d'accès permis par ce sélecteur de trafic. Pour les protocoles pour lesquels l'accès est indéfini (incluant le protocole 0) ou si tous les accès sont permis, ce champ DOIT être 65535. Les valeurs de type et code ICMP et ICMPv6, ainsi que les valeurs de type MH MIPv6, sont représentées dans ce champ comme spécifié au paragraphe 4.4.1.1 de la [RFC4301]. Les valeurs de type et code ICMP sont traitées comme un seul numéro d'accès entier de 16 bits, avec le type dans les huit bits de poids fort et le code dans les huit bits de moindre poids. Les valeurs de type MH MIPv6 sont traitées comme un seul numéro d'accès entier de 16 bits, avec le type dans les huit bits de poids fort et les huit bits de moindre poids réglés à zéro.
- Adresse de début - plus petite adresse incluse dans ce sélecteur de trafic (longueur déterminée par le type de TS).
- Adresse de fin - 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 "TOUS" accès DOIVENT réglet l'accès de début à 0 et l'accès de fin à 65535 ; noter que selon la [RFC4301], "TOUS" inclut "OPAQUE". Les systèmes qui fonctionnent avec la [RFC4301] qui souhaitent indiquer des accès "OPAQUE", mais pas "TOUS" les accès, DOIVENT régler l'accès de début à 65535 et l'accès de fin à 0.

Les types de sélecteur de trafic 7 et 8 peuvent aussi se référer aux champs de type et code ICMP ou ICMPv6, ainsi qu'aux champs de type MH pour l'en-tête de mobilité IPv6 [RFC6275]. Noter cependant que les paquets ICMP ou MIPv6 n'ont pas de champs séparés de source et de destination. La méthode pour spécifier les sélecteurs de trafic pour ICMP et MIPv6 est montrée par exemple au paragraphe 4.4.1.3 de la [RFC4301].

La tableau qui suit fait la liste des valeurs pour le champ Type de sélecteur de trafic et les données correspondantes de sélecteur d'adresse. Les valeurs du tableau qui suit ne sont que celles en cours à la date de publication de RFC 4306. D'autres valeurs peuvent avoir été ajoutées depuis ou le seront après la publication du présent document. Le lecteur devrait se référer à [IKEV2IANA] pour avoir les dernières valeurs.

Type de TS	**Valeur**	
TS_IPV4_ADDR_RANGE	7	Gamme d'adresses IPv4 représentée par deux valeurs de quatre octets. La première valeur est le début des adresses IPv4 (inclusif) et la seconde valeur est la fin des adresses IPv4 (inclusive). Toutes les adresses entre les deux adresses spécifiées sont considérées être dans la liste.
TS_IPV6_ADDR_RANGE	8	Gamme d'adresses IPv6 représentée par deux valeurs de seize octets. La première valeur est le début des adresses IPv6 (inclusif) et la seconde valeur est la fin des adresses IPv6 (inclusive). Toutes les adresses entre les deux adresses spécifiées sont considérées être dans la liste.

3.14 Charge utile Chiffré

La charge utile Chiffré, notée SK {...} dans le présent document, contient d'autres charges utiles en forme chiffrée. La charge utile Chiffré, si elle est présente dans un message, DOIT être la dernière charge utile du message. Souvent, elle est la seule charge utile du message. Cette charge utile est aussi appelée la charge utile "chiffrée et authentifiée".

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

Le présent document spécifie le traitement cryptographique des charges utiles Chiffré en utilisant un chiffrement de bloc en mode CBC et un algorithme de vérification de l'intégrité qui calcule une somme de contrôle de longueur fixe sur un message de taille variable. La conception est modélisée d'après les algorithmes ESP décrits dans les [RFC2104], [RFC2451], et [RFC4303]. Le présent document spécifie complètement le traitement cryptographique des données de IKE, mais ces documents devraient être consultés pour les raisons du concept. De futurs documents pourront spécifier le traitement des charges utiles Chiffré pour les autres types de transformations, comme le chiffrement en mode compteur et les algorithmes de chiffrement authentifié. Les homologues NE DOIVENT PAS négocier de transformations pour lesquelles il n'existe pas de telles spécifications.

Quand un algorithme de chiffrement authentifié est utilisé pour protéger la SA IKE, la construction de la charge utile Chiffré est différente de ce qui est décrit ici. Voir dans la [RFC5282] plus d'informations sur les algorithmes de chiffrement authentifié et leur utilisation dans IKEv2.

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

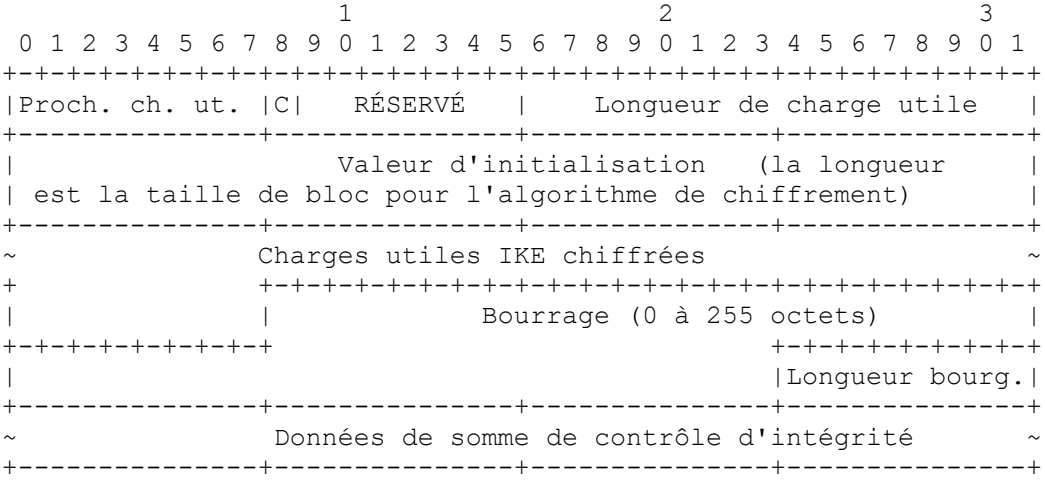


Figure 21 : Format de charge utile Chiffré

- Prochaine charge utile - type de charge utile de la première charge utile incorporée. Noter que c'est une exception au format d'en-tête standard, car la charge utile Chiffré est la dernière charge utile du message et donc, le champ Prochaine charge utile devrait normalement être zéro. Mais parce que le contenu de cette charge utile est constitué de charges utiles incorporées et qu'il n'y avait pas de place naturelle pour mettre le type de la première, ce type est placé ici.

- Longueur de charge utile - inclut les longueurs de l'en-tête, de la valeur d'initialisation (IV), des charges utiles IKE chiffrées, du bourrage, de la longueur du bourrage, et des données de somme de contrôle d'intégrité.

- Valeur d'initialisation -pour les chiffrements en mode CBC, la longueur de la valeur d'initialisation (IV) est égale à la longueur de bloc de l'algorithme de chiffrement sous-jacent. Les envoyeurs DOIVENT choisir pour chaque message une nouvelle IV imprévisible ; les receveurs DOIVENT accepter toute valeur. Le lecteur est invité à consulter [MODES] pour des conseils sur la génération d'une IV. En particulier, utiliser le bloc final du texte chiffré du message précédent n'est pas considéré comme imprévisible. Pour les modes autres que CBC, le format d'IV et le traitement sont spécifiés dans le document qui spécifie l'algorithme et le mode de chiffrement.

- Charges utiles IKE qui sont spécifiées plus haut dans cette section. Ce champ est chiffré avec le chiffrement négocié.

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

- Longueur de bourrage est la longueur du champ Bourrage. L'envoyeur DEVRAIT régler la longueur de bourrage à la valeur minimum qui rend 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 résulte en un alignement approprié. Ce champ est chiffré avec le chiffrement négocié.

- Données de somme de contrôle d'intégrité est la somme de contrôle cryptographique du message entier en commençant par l'en-tête fixe IKE jusqu'à 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 Configuration

La charge utile Configuration, notée CP dans le présent document, est utilisée pour échanger des informations de configuration entre les homologues IKE. L'échange est d'un IRAC qui demande une adresse IP interne à un IRAS et pour échanger d'autres informations de la sorte qu'on acquerrait avec le protocole de configuration dynamique d'hôte (DHCP) si l'IRAC était directement connecté à un LAN.

La charge utile Configuration est définie comme suit :

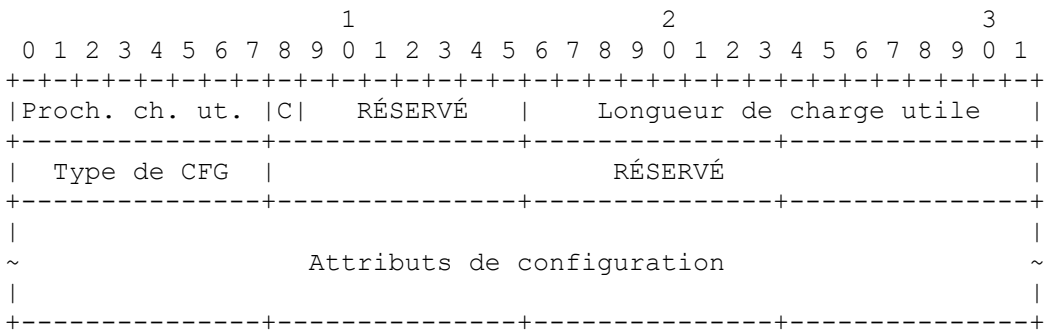


Figure 22 : Format de charge utile Configuration

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

- Type de CFG (1 octet) - type d'échange représenté par les attributs de configuration. Les valeurs du tableau qui suit ne sont que celles en cours à la date de publication de RFC 4306. D'autres valeurs peuvent avoir été ajoutées depuis ou le seront après la publication du présent document. Le lecteur devrait se référer à [IKEV2IANA] pour avoir les dernières valeurs.

Type de CFG	Valeur
CFG_REQUEST	1
CFG_REPLY	2

CFG_SET	3
CFG_ACK	4

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

o Attributs de configuration (longueur variable) - ce sont des structures de type longueur valeur (TLV) spécifiques de la charge utile 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

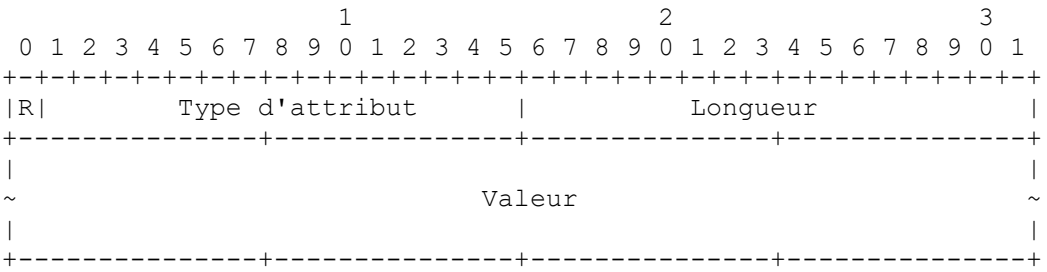


Figure 23 : Format de l'attribut de configuration

o Réservé (1 bit) - ce bit DOIT être réglé à zéro et DOIT être ignoré en réception.
o Type d'attribut (15 bits) - identifiant unique pour chaque type d'attribut de configuration.
o Longueur (2 octets, entier non signé) - longueur en octets de la valeur.
o Valeur (0, un ou plusieurs octets) - valeur de longueur variable de cet attribut de configuration. La liste suivante énumère les types d'attribut.

Les valeurs du tableau suivant sont seulement celles en cours à la date de publication de la RFC 4306 (sauf INTERNAL_ADDRESS_EXPIRY et INTERNAL_IP6_NBNS, qui ont été retirées par la RFC 5996). D'autres valeurs peuvent avoir été ajoutées depuis ou le seront après la publication du présent document. Le lecteur devrait se référer à [IKEV2IANA] pour avoir les dernières valeurs.

Type d'attribut	Valeur	Multi valeurs	Longueur
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_IP4_DHCP	6	oui	0 ou 4 octets
APPLICATION_VERSION	7	non	0 ou more
INTERNAL_IP6_ADDRESS	8	oui*	0 ou 17 octets
INTERNAL_IP6_DNS	10	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 peuvent être multi valeurs en retour seulement si des valeurs multiples étaient demandées.

o INTERNAL_IP4_ADDRESS, INTERNAL_IP6_ADDRESS - adresse sur le réseau interne, parfois appelé adresse de nœud rouge ou adresse privée, et ce PEUT être une adresse privée sur l'Internet. Dans un message de demande, l'adresse spécifiée est une adresse demandée (ou une adresse de longueur zéro si aucune adresse spécifique n'est demandée). Si une adresse spécifique est demandée, cela indique probablement qu'une connexion précédente existait avec cette adresse et le demandeur voudrait réutiliser cette adresse. Avec IPv6, un demandeur PEUT fournir les octets de moindre poids de l'adresse qu'il veut utiliser. Plusieurs adresses internes PEUVENT être demandées en demandant plusieurs attributs d'adresse interne. Le répondant PEUT seulement envoyer jusqu'au nombre des adresses demandées. L'attribut INTERNAL_IP6_ADDRESS est constitué de deux champs : le premier est une adresse IPv6 de 16 octets, et le second est une longueur de préfixe d'un octet comme défini dans la [RFC4291]. L'adresse demandée est valide tant qu'est valide la SA IKE (ou ses successeurs après changement de clé) qui demande l'adresse. Ceci est décrit plus en détails au paragraphe 3.15.3.

- INTERNAL_IP4_NETMASK - gabarit réseau du réseau interne. Un seul gabarit réseau est permis dans les messages de demande et de réponse (par exemple, 255.255.255.0) et il DOIT n'être utilisé qu'avec un attribut INTERNAL_IP4_ADDRESS. INTERNAL_IP4_NETMASK dans une CFG_REPLY signifie en gros la même chose qu'un INTERNAL_IP4_SUBNET contenant les mêmes informations ("envoyer le trafic à ces adresses à travers moi") mais aussi implique une limite de liaison. Par exemple, le client pourrait utiliser sa propre adresse et le gabarit réseau pour calculer l'adresse de diffusion de la liaison. Un attribut INTERNAL_IP4_NETMASK vide peut être inclus dans une CFG_REQUEST pour demander ces informations (bien que la passerelle puisse envoyer les informations même quand elles ne sont pas demandées). Des valeurs non vides pour cet attribut dans une CFG_REQUEST n'ont pas de sens et donc NE DOIVENT PAS être incluses.

- INTERNAL_IP4_DNS, INTERNAL_IP6_DNS - spécifie une adresse d'un serveur DNS au sein du réseau. Plusieurs serveurs DNS PEUVENT être demandés. Le répondant PEUT répondre avec zéro, un ou plusieurs attributs de serveur DNS.

- INTERNAL_IP4_NBNS - spécifie une adresse d'un serveur de noms NetBios (WINS) dans le réseau. Plusieurs serveurs NBNS PEUVENT être demandés. Le répondant PEUT répondre avec zéro, un ou plusieurs attributs de serveur NBNS.

- INTERNAL_IP4_DHCP, INTERNAL_IP6_DHCP - ordonne à l'hôte d'envoyer toutes les demandes DHCP internes à l'adresse contenue dans l'attribut. Plusieurs serveurs DHCP PEUVENT être demandés. Le répondant PEUT répondre avec zéro, un ou plusieurs attributs de serveur DHCP.

- APPLICATION_VERSION - informations de version ou application de l'hôte IPsec. C'est une chaîne de caractères ASCII imprimables qui N'EST PAS terminée par un Nul.

- INTERNAL_IP4_SUBNET - sous réseaux protégés que cet appareil de bordure protège. Cet attribut est constitué 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 par zéro, un ou plusieurs attributs de sous réseau. Ceci est discuté plus en détails au paragraphe 3.15.2.

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

- INTERNAL_IP6_SUBNET - sous réseaux protégés que protège cet appareil bordure. Cet attribut est constitué de deux champs : le premier est une adresse IPv6 de 16 octets, et le second est une longueur de préfixe de un octet comme défini dans la [RFC4291]. Plusieurs sous réseaux PEUVENT être demandés. Le répondant PEUT répondre avec zéro, un ou plusieurs attributs de sous réseau. Ceci est discuté plus en détails au paragraphe 3.15.2.

Noter qu'aucune recommandation n'est faite dans le présent document sur la façon dont une mise en œuvre représente en fait quelles informations sont envoyées dans une réponse. C'est-à-dire qu'on ne recommande aucune méthode spécifique d'un IRAS pour déterminer quel serveur DNS devrait être retourné à la demande d'un IRAC.

La paire CFG_REQUEST et CFG_REPLY permet à un point d'extrémité IKE de demander des informations à son homologue. Si un attribut dans la charge utile Configuration CFG_REQUEST n'est pas de longueur zéro, il est pris comme une suggestion pour cet attribut. La charge utile 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 attributs non reconnus ou non pris en charge DOIVENT être ignorés aussi bien dans les demandes que dans les réponses.

La paire CFG_SET et 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 Configuration CFG_SET contient des attributs que l'initiateur veut qu'altère son homologue. Le répondant DOIT retourner une charge utile Configuration pour chaque message si il accepte des données de configuration, et la charge utile Configuration DOIT contenir les attributs que le répondant a acceptés avec des données de longueur zéro. Les attributs qu'il n'a pas accepté NE DOIVENT PAS être dans la charge utile Configuration CFG_ACK. Si aucun attribut n'a été accepté, le répondant DOIT retourner soit une charge utile CFG_ACK vide, soit un message de réponse sans charge utile CFG_ACK. Il n'y a actuellement pas d'utilisation définie pour 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 de la présente spécification PEUT ignorer les charges utiles CFG_SET.

3.15.2 Signification de INTERNAL_IP4_SUBNET et INTERNAL_IP6_SUBNET

Les attributs INTERNAL_IP4/6_SUBNET peuvent indiquer des sous réseaux supplémentaires, de ceux qui ont besoin d'une ou plusieurs SA séparées, qui peuvent être accédées à travers la passerelle qui annonce les attributs. Les attributs

INTERNAL_IP4/6_SUBNET peuvent aussi exprimer la politique de la passerelle quant au trafic qui devrait être envoyé à travers la passerelle ; le client peut choisir si d'autre trafic (couvert par TSr, mais non dans INTERNAL_IP4/6_SUBNET) est envoyé à travers la passerelle ou directement à la destination. Donc, le trafic pour les adresses dont la liste figure dans l'attribut INTERNAL_IP4/6_SUBNET devrait être envoyé à travers la passerelle qui annonce les attributs. Si il n'existe pas de SA filles dont les sélecteurs de trafic couvrent l'adresse en question, de nouvelles SA devraient être créées.

Par exemple, si il y a deux sous réseaux, 198.51.100.0/26 et 192.0.2.0/24, et si la demande du client contient ce qui suit :

CP(CFG_REQUEST) = INTERNAL_IP4_ADDRESS()
TSi = (0, 0-65535, 0.0.0.0-255.255.255.255)
TSr = (0, 0-65535, 0.0.0.0-255.255.255.255)

alors une réponse valide pourrait être la suivante (dans laquelle TSr et INTERNAL_IP4_SUBNET contiennent les mêmes informations) :

CP(CFG_REPLY) =
  INTERNAL_IP4_ADDRESS(198.51.100.234)
  INTERNAL_IP4_SUBNET(198.51.100.0/255.255.255.192)
  INTERNAL_IP4_SUBNET(192.0.2.0/255.255.255.0)
TSi = (0, 0-65535, 198.51.100.234-198.51.100.234)
TSr = ((0, 0-65535, 198.51.100.0-198.51.100.63), (0, 0-65535, 192.0.2.0-192.0.2.255))

Dans ces cas, le INTERNAL_IP4_SUBNET ne porte pas réellement d'informations utiles.

Une réponse différente possible aurait été celle-ci :
CP(CFG_REPLY) =
  INTERNAL_IP4_ADDRESS(198.51.100.234)
  INTERNAL_IP4_SUBNET(198.51.100.0/255.255.255.192)
  INTERNAL_IP4_SUBNET(192.0.2.0/255.255.255.0)
TSi = (0, 0-65535, 198.51.100.234-198.51.100.234)
TSr = (0, 0-65535, 0.0.0.0-255.255.255.255)

Cette réponse signifierait que le client peut envoyer tout son trafic à travers la passerelle, mais la passerelle n'a pas d'objection à ce que le client envoie du trafic non inclus dans INTERNAL_IP4_SUBNET directement à la destination (sans passer à travers la passerelle).

Une situation différente se produit si la passerelle a une politique qui exige que le trafic pour les deux sous réseaux soit porté dans des SA séparées. Alors une réponse comme celle-ci indiquerait au client que si il veut accéder au second sous réseau, il doit créer une SA séparée :

CP(CFG_REPLY) =
  INTERNAL_IP4_ADDRESS(198.51.100.234)
  INTERNAL_IP4_SUBNET(198.51.100.0/255.255.255.192)
  INTERNAL_IP4_SUBNET(192.0.2.0/255.255.255.0)
TSi = (0, 0-65535, 198.51.100.234-198.51.100.234)
TSr = (0, 0-65535, 198.51.100.0-198.51.100.63)

INTERNAL_IP4_SUBNET peut aussi être utile si le TSr du client incluait seulement une partie de l'espace d'adresses. Par exemple, si le client demande :

CP(CFG_REQUEST) = INTERNAL_IP4_ADDRESS()
TSi = (0, 0-65535, 0.0.0.0-255.255.255.255)
TSr = (0, 0-65535, 192.0.2.155-192.0.2.155)

La réponse de la passerelle pourrait être :

CP(CFG_REPLY) =
  INTERNAL_IP4_ADDRESS(198.51.100.234)
  INTERNAL_IP4_SUBNET(198.51.100.0/255.255.255.192)
  INTERNAL_IP4_SUBNET(192.0.2.0/255.255.255.0)
TSi = (0, 0-65535, 198.51.100.234-198.51.100.234)
TSr = (0, 0-65535, 192.0.2.155-192.0.2.155)

Comme la signification de INTERNAL_IP4_SUBNET/INTERNAL_IP6_SUBNET dans une CFG_REQUEST n'est pas clair, ils ne peuvent pas être utilisés de façon fiable dans les CFG_REQUEST.

3.15.3 Charges utiles de configuration pour IPv6

Les charges utiles Configuration pour IPv6 se fondent sur les charges utiles IPv4 correspondantes, et ne suivent pas complètement la "façon normale dont IPv6 fait les choses". En particulier, l'auto configuration IPv6 sans état ou les messages d'annonce de routeur ne sont pas utilisés, ni la découverte de voisin. Noter qu'il y a un document supplémentaire qui discute de la configuration IPv6 dans IKEv2, la [RFC5739]. Pour l'instant, c'est un document expérimental, mais il est espéré qu'avec plus d'expérience de mise en œuvre, il obtiendra le même traitement de normalisation que le présent document.

Une adresse IPv6 peut être allouée à un client en utilisant la charge utile Configuration INTERNAL_IP6_ADDRESS. Un échange minimal pourrait ressembler à :

```
CP(CFG_REQUEST) =
   INTERNAL_IP6_ADDRESS()
   INTERNAL_IP6_DNS()
TSi = (0, 0-65535, :: - FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF)
TSr = (0, 0-65535, :: - FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF)

CP(CFG_REPLY) =
   INTERNAL_IP6_ADDRESS(2001:DB8:0:1:2:3:4:5/64)
   INTERNAL_IP6_DNS(2001:DB8:99:88:77:66:55:44)
TSi = (0, 0-65535, 2001:DB8:0:1:2:3:4:5 - 2001:DB8:0:1:2:3:4:5)
TSr = (0, 0-65535, :: - FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF)
```

Le client PEUT envoyer un attribut INTERNAL_IP6_ADDRESS non vide dans la CFG_REQUEST pour demander une adresse ou identifiant d'interface spécifique. La passerelle vérifie d'abord si l'adresse spécifiée est acceptable, et si elle l'est, la retourne. Si l'adresse n'était pas acceptable, la passerelle tente d'utiliser l'identifiant d'interface avec quelque autre préfixe ; si même cela échoue, la passerelle choisit un autre identifiant d'interface.

L'attribut INTERNAL_IP6_ADDRESS contient aussi un champ de longueur de préfixe. Quand il est utilisé dans une CFG_REPLY, cela correspond à l'attribut INTERNAL_IP4_NETMASK dans le cas IPv4.

Bien que cette approche de configuration des adresses IPv6 soit raisonnablement simple, elle a quelques limitations. Les tunnels IPsec configurés en utilisant IKEv2 ne sont pas des "interfaces" pleinement constituées dans le sens de l'architecture d'adressage IPv6 [RFC4291]. En particulier, elles n'ont pas nécessairement des adresses de liaison locale, et cela peut compliquer l'utilisation des protocoles qui les supposent, comme la [RFC3810].

3.15.4 Échecs d'allocation d'adresse

Si le répondant rencontre une erreur lorsque il tente d'allouer une adresse IP à l'initiateur durant le traitement d'une charge utile Configuration, il répond par une notification INTERNAL_ADDRESS_FAILURE. La SA IKE est quand même créée même si la SA fille initiale ne peut pas être créée à cause de cet échec. Si cette erreur est générée au sein d'un échange IKE_AUTH, aucune SA fille ne va être créée. Cependant, il y a des cas d'erreur plus complexes.

Si le répondant ne prend pas du tout en charge les charges utiles Configuration, il peut simplement ignorer toutes les charges utiles Configuration. Ce type de mise en œuvre n'envoie jamais de notifications INTERNAL_ADDRESS_FAILURE.

Si l'initiateur exige l'allocation d'une adresse IP, il va traiter une réponse sans CFG_REPLY comme une erreur.

L'initiateur peut demander un type particulier d'adresse (IPv4 ou IPv6) que le répondant ne prend pas en charge, même si le répondant accepte les charges utiles Configuration. Dans ce cas, le répondant ignore simplement le type d'adresse qu'il ne prend pas en charge et traite le reste de la demande comme d'habitude.

Si l'initiateur demande plusieurs adresses d'un type que le répondant prend en charge, et si certaines (mais pas toutes) des demandes échouent, le répondant répond avec les seules adresses réussies. Le répondant n'envoie une INTERNAL_ADDRESS_FAILURE que si aucune adresse ne peut être allouée.

Si l'initiateur ne reçoit pas la ou les adresses IP demandées par sa politique, il PEUT garder la SA IKE active et réessayer la charge utile Configuration dans un échange INFORMATIONAL séparé après une temporisation convenable, ou il PEUT supprimer la SA IKE en envoyant une charge utile Supprime dans un échange INFORMATIONAL séparé et ensuite ressayer

une AS IKE depuis le début après une certaine temporisation. Une telle temporisation ne devrait pas être trop courte (en particulier si la SA IKE est commencée depuis le début) parce que ces situations d'erreur peuvent ne pas pouvoir être réparées rapidement ; la temporisation devrait probablement être de plusieurs minutes. Par exemple, un problème de pénurie d'adresses chez le répondant ne va probablement être repéré que quand plus d'entrées sont retournées au réservoir d'adresses quand d'autres clients se déconnectent ou quand le répondant est reconfiguré avec un plus grand réservoir d'adresses.

3.16 Charge utile Protocole d'authentification extensible (EAP)

La charge utile Protocole d'authentification extensible, notée EAP dans le présent document, permet aux SA IKE d'être authentifiées en utilisant le protocole défini dans la [RFC3748] et dans les extensions ultérieures à ce protocole. Quand on utilise EAP, une méthode appropriée doit être choisie. Beaucoup de méthodes ont été définies, qui spécifient l'utilisation du protocole avec divers mécanismes d'authentification. Les types de méthodes EAP sont énumérés dans [EAP-IANA]. Un bref résumé du format EAP est inclus ici pour en faciliter la compréhension.

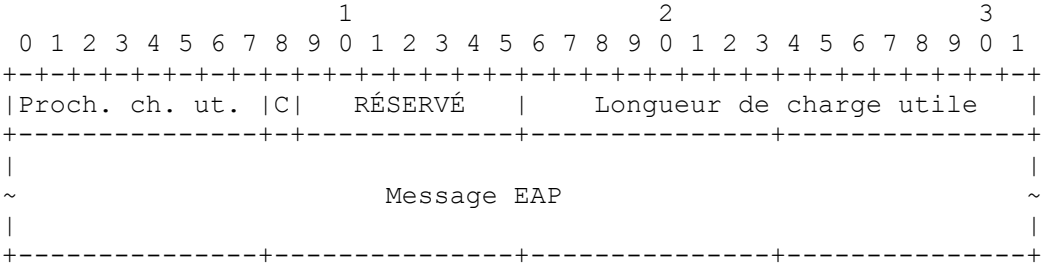


Figure 24 : Format de charge utile EAP

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

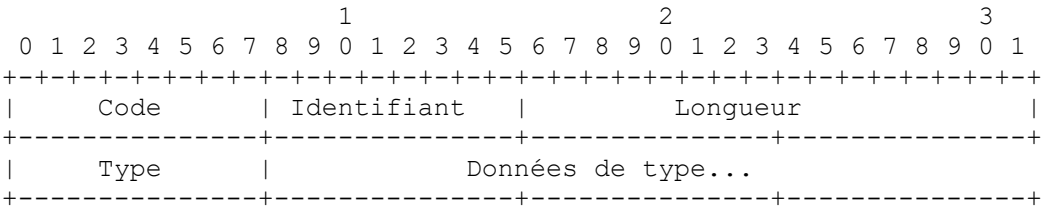


Figure 25 : Format de message EAP

- Code (1 octet) - indique si ce message est une demande (1), une réponse (2), un succès (3), ou un échec (4).

- Identifiant (1 octet) - utilisé dans PPP pour distinguer les messages dupliqués de ceux qui sont répétés. Comme dans IKE EAP fonctionne sur un protocole fiable, l'identifiant n'a pas de fonction ici. Dans un message de réponse, cet octet DOIT être réglé à correspondre à l'identifiant dans la demande correspondante.

- Longueur (2 octets, entier non signé) - longueur du message EAP. DOIT être quatre de moins que la longueur de charge utile de la charge utile encapsulante.

- Type (1 octet) - n'est présent que si le champ Code est Demande (1) ou Réponse (2). Pour les autres codes,la longueur du message EAP DOIT être 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 soit être NAK, soit correspondre au type des données demandées. Noter que comme IKE passe une indication de l'identité de l'initiateur dans le premier message de l'échange IKE_AUTH, le répondant NE DEVRAIT PAS envoyer de demande d'identité EAP (type 1). L'initiateur PEUT cependant répondre à de telles demandes si il en reçoit.

- Données de type (longueur variable) - varie selon le type de demande et la réponse associée. Pour la documentation des méthodes EAP, voir la [RFC3748].

Noter que comme IKE passe une indication de l'identité de l'initiateur dans le premier message de l'échange IKE_AUTH, le répondant NE DEVRAIT PAS envoyer de demande Identité EAP. L'initiateur PEUT cependant répondre à de telles demandes si il en reçoit.

4. Exigences de conformité

Afin de s'assurer que toutes les mises en œuvre de IKEv2 peuvent interopérer, il y a des exigences "DOIT prendre en charge" en plus de celles mentionnées 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 le complet établissement des SA. Ainsi une mise en œuvre particulière peut être configurée avec un certain nombre de restrictions concernant les algorithmes et les autorités de confiance qui vont empêcher une interopérabilité universelle.

IKEv2 est destiné à permettre que des mises en œuvre minimales puissent interopérer avec toutes les mises en œuvre conformes. Les caractéristiques suivantes sont celles qui peuvent être omises dans une mise en œuvre minimale :
- o Capacité de négocier les SA à travers un NAT et tunneler la SA ESP résultante sur UDP.
- o Capacité de demander (et répondre à une demande) une adresse IP temporaire sur l'extrémité distante d'un tunnel.
- o Capacité de prendre en charge l'authentification fondée sur EAP.
- o Capacité de prendre en charge des tailles de fenêtre supérieures à un.
- o Capacité d'établir plusieurs SA ESP ou AH au sein d'une seule SA IKE.
- o Capacité de changer 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 de juste 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 utile. Si le bit critique est établi dans un en-tête de charge utile non prise en charge, toutes les mises en œuvre DOIVENT rejeter les messages contenant ces charges utiles.

Chaque mise en œuvre DOIT être capable de faire les échanges de quatre messages IKE_SA_INIT et IKE_AUTH pour établir deux SA (une pour IKE, une pour ESP ou AH). Les mises en œuvre PEUVENT être seulement initiateur ou seulement répondant si c'est approprié pour leur plate-forme. Chaque mise en œuvre DOIT être capable de répondre à un échange INFORMATIONAL, mais une mise en œuvre minimale PEUT répondre à toute demande dans l'échange INFORMATIONAL avec une réponse vide (noter que dans le contexte d'une SA IKE, un message "vide" consiste en un en-tête IKE suivi par une charge utile Chiffré qui ne contient aucune charge utile). Une mise en œuvre minimale PEUT ne prendre en charge l'échange CREATE_CHILD_SA que dans la mesure de la reconnaissance des demandes et de leur rejet avec une charge utile Notifie de type NO_ADDITIONAL_SAS. Une mise en œuvre minimale n'a pas besoin d'être capable d'initier un échange CREATE_CHILD_SA ou INFORMATIONAL. Quand une SA arrive à expiration (sur la base de valeurs configurées en local de durée de vie ou de nombre d'octets passés) une mise en œuvre PEUT soit essayer de la renouveler avec un échange CREATE_CHILD_SA soit elle PEUT supprimer (clore) 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 à la place être capable de supprimer la vieille SA et d'en créer une nouvelle.

Les mises en œuvre ne sont pas obligées de prendre en charge la demande d'adresse IP temporaire ou la réponse à de telles demandes. Si une mise en œuvre prend en charge la production de telles demandes et si sa politique exige d'utiliser des adresses IP temporaires, elle DOIT inclure une charge utile CP dans le premier message de l'échange IKE_AUTH 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 prend en charge de répondre à de telles demandes, elle DOIT analyser la charge utile CP de type CFG_REQUEST dans le premier message de l'échange IKE_AUTH et reconnaître un champ de type INTERNAL_IP4_ADDRESS ou INTERNAL_IP6_ADDRESS. Si elle prend en charge le prêt d'adresse du type approprié, elle DOIT retourner une charge utile CP de type CFG_REPLY contenant une adresse du type demandé. Le répondant peut inclure tout autre attribut en rapport.

Pour qu'une mise en œuvre soit dite conforme à la présente spécification, elle DOIT être capable de se configurer à accepter :
- o l'infrastructure de clé publique utilisant les certificats X.509 (PKIX) contenant, et signés par, des clés RSA de taille 1024 ou 2048 bits, où l'identifiant passé est ID_KEY_ID, ID_FQDN, ID_RFC822_ADDR, ou ID_DER_ASN1_DN ;
- o l'authentification par clé partagée où l'identifiant passé est ID_KEY_ID, ID_FQDN, ou ID_RFC822_ADDR ;
- o l'authentification où le répondant est authentifié en utilisant les certificats PKIX et l'initiateur est authentifié en utilisant l'authentification par clé partagée.

5. Considérations sur la sécurité

Bien que le présent protocole soit destiné à minimiser la divulgation des informations de configuration aux homologues non autorisés, de telles divulgations sont inévitables. Un homologue ou l'autre doit d'abord s'identifier et prouver son identité. Pour éviter d'être sondé, il est exigé de l'initiateur qu'il s'identifie en premier, et généralement il est exigé qu'il s'authentifie en premier. L'initiateur peut, cependant, apprendre que le répondant prend en charge IKE et quels protocoles de chiffrement il

accepte. Le répondant (ou quelqu'un qui se fait passer pour le répondant) non seulement peut sonder l'initiateur sur son identité mais peut, en utilisant les charges utiles CERTREQ, être capable de déterminer quels certificats l'initiateur veut utiliser.

L'utilisation de l'authentification EAP change un peu les possibilités de sondage. Quand l'authentification EAP est utilisée, le répondant prouve son identité avant que l'initiateur le fasse, de sorte qu'un initiateur ayant connaissance du nom d'un initiateur valide pourrait sonder le répondant pour à la fois son nom et ses certificats.

Des changements de clé répétés en utilisant CREATE_CHILD_SA sans échanges Diffie-Hellman supplémentaires laisse toutes les SA vulnérables à la cryptanalyse d'une seule clé. Les développeurs devraient prendre note de ce fait et établir une limite aux échanges CREATE_CHILD_SA entre les exponentiations. Le présent document ne prescrit pas de telle limite.

La force d'une clé déduite d'un échange Diffie-Hellman utilisant un des groupes définis ici dépend de la force inhérente au groupe, de la taille de l'exposant utilisé, et de l'entropie fournie par le générateur de nombres aléatoires utilisé. Du fait de ces entrées, il est difficile de déterminer la force d'une clé pour un des groupes définis. Le groupe Diffie-Hellman numéro deux, quand il est utilisé avec un fort générateur de nombres aléatoires et un exposant pas inférieur à 200 bits, est d'utilisation courante avec 3DES. Le groupe cinq fournit une sécurité supérieure à celle du groupe deux. Le groupe un est seulement pour des raisons historiques et ne fournit pas une force suffisante sauf avec DES, qui est aussi seulement d'utilisation historique. Les mises en œuvre devraient prendre note de ces estimations quand elles établissent une politique et négocient leurs 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 de plus forts groupes ni rien qui dilue la force obtenue de groupes plus forts (limités par la force des autres algorithmes négociés incluant la PRF). En fait, le cadre extensible de IKE encourage la définition de plus de groupes ; l'utilisation des groupes à courbe elliptique peut augmenter considérablement la force tout en utilisant de bien plus petits nombres.

Il est supposé que tous les exposants Diffie-Hellman sont écrasés dans la mémoire après usage.

Les échanges IKE_SA_INIT et IKE_AUTH se produisent avant que l'initiateur ait été authentifié. Par suite, une mise en œuvre de ce protocole a besoin d'être très robuste quand elle est déployée sur un réseau non sûr. Les vulnérabilités des mises en œuvre, en particulier les attaques de déni de service, peuvent être exploitées par des homologues non authentifiés. Cette question est particulièrement préoccupante à cause du nombre illimité de messages dans l'authentification fondée sur EAP.

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

La sécurité de ce protocole est très dépendante de l'aléa des paramètres choisis au hasard. Ils devraient être générés par un aléa fort ou une source pseudo aléatoire avec un germe approprié (voir la [RFC4086]). Les développeurs devraient prendre soin de s'assurer que l'utilisation de nombres aléatoires pour les clés et les noms occasionnels est gérée d'une façon qui ne sape pas la sécurité des clés.

Pour des informations sur les raisons de beaucoup des choix de conception cryptographique de ce protocole, voir [SIGMA] et [SKEME]. Bien que la sécurité des SA filles négociées ne dépende pas de la force du chiffrement et de la protection d'intégrité négociées dans la SA IKE, les mises en œuvre NE DOIVENT PAS négocier "AUCUN" comme algorithme de protection de l'intégrité IKE ou ENCR_NULL comme algorithme de chiffrement IKE.

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

Les notifications NAT_DETECTION_*_IP contiennent un hachage des adresses et accès pour tenter de cacher les adresses IP internes derrière un NAT. Comme l'espace d'adresses IPv4 est seulement de 32 bits, et qu'il est généralement très peu dense, il serait possible à un attaquant de trouver l'adresse interne utilisée derrière le boîtier de NAT en essayant toutes les adresses IP possibles et en essayant de trouver le hachage qui leur correspond. Les numéros d'accès sont normalement fixés à 500, et les SPI peuvent être extraits du paquet. Cela réduit le nombre de calculs de hachage à 2^32. Avec une hypothèse réfléchie sur l'utilisation de l'espace des adresses privées, le nombre de calculs de hachages est beaucoup plus petit. Les concepteurs devraient donc ne pas supposer que l'utilisation de IKE ne va pas laisser fuir d'informations d'adresse interne.Quand on utilise une méthode d'authentification EAP qui ne génère pas de clé partagés pour protéger une charge utile AUTH qui suit, certaines attaques par interposition et de contrefaçon de serveur sont possibles [EAPMITM].

Ces vulnérabilités se produisent quand EAP est aussi utilisé dans des protocoles qui ne sont pas protégés avec un tunnel sécurisé. Comme EAP est un protocole d'authentification d'utilisation générale, qui est souvent utilisé pour fournir des facilités de signature automatique, une solution IPsec déployée qui s'appuie sur une méthode d'authentification EAP qui ne génère pas

de clé partagée (aussi appelée méthode EAP non génératrice de clé) peut être compromise à cause du déploiement d'une application sans aucun rapport qui utilise aussi la méthode EAP non génératrice de clé, mais de façon non protégée. Noter que cette vulnérabilité n'est pas limitée juste à EAP, mais peut se produire dans d'autres scénarios où est réutilisée une infrastructure d'authentification. Par exemple, si le mécanisme EAP utilisé par IKEv2 se sert d'un jeton authentificateur, un attaquant interposé peut se faire passer pour le serveur de la Toile, intercepter l'échange d'authentification de jeton, et l'utiliser pour initier une connexion IKEv2. Pour cette raison, l'utilisation des méthodes EAP non génératrices de clé DEVRAIT être évitée lorsque possible. Lorsque elles sont utilisées, il est extrêmement important que tous les usages de ces méthodes EAP utilisent un tunnel protégé, où l'initiateur valide le certificat du répondant avant d'initier l'authentification EAP. Les développeurs devraient décrire les vulnérabilités de l'utilisation des méthodes EAP non génératrices de clé dans la documentation de leurs mises en œuvre afin que les administrateurs qui déploient des solutions IPsec soient conscients de ces dangers.

Une mise en œuvre qui utilise EAP DOIT aussi utiliser une authentification fondée sur une clé publique du serveur auprès du client avant que commence l'authentification EAP, même si la méthode EAP offre l'authentification mutuelle. Cela évite d'avoir des variantes supplémentaires de IKEv2 et protège les données EAP des attaques actives.

Si les messages de IKEv2 sont assez longs pour que la fragmentation de niveau IP soit nécessaire, il est possible que des attaquants puissent empêcher l'échange de s'achever en épuisant les capacités des mémoires tampon de réassemblage. Les chances que cela arrive peuvent être minimisées en utilisant les codages de hachage et URL au lieu d'envoyer des certificats (voir au paragraphe 3.6). Des moyens d'atténuation supplémentaires sont exposés dans [DOS].

Le contrôle d'admission est critique pour la sécurité du protocole. Par exemple, les ancres de confiance utilisées pour identifier les homologues IKE devraient probablement être différentes de celles utilisées pour les autres formes de confiance, comme celles utilisées pour identifier les serveurs publics de la Toile. De plus, bien que IKE laisse une grande liberté d'action pour définir la politique de sécurité pour l'identité, les accréditifs d'un homologue de confiance, et la corrélation entre eux, avoir une telle politique de sécurité définie explicitement est essentiel pour une mise en œuvre sûre.

5.1 Autorisation de sélecteur de trafic

IKEv2 s'appuie sur les informations de la base de données d'autorisation d'homologue (PAD, *Peer Authorization Database*) quand il détermine quelles sortes de SA filles un homologue est autorisé à créer. Ce processus est décrit au paragraphe 4.4.3 de la [RFC4301]. Quand un homologue demande la création d'une SA fille avec des sélecteurs de trafic, la PAD doit contenir des "données d'autorisation de SA fille" reliant l'identité authentifiée par IKEv2 et les adresses permises pour les sélecteurs de trafic.

Par exemple, la PAD peut être configurée de telle sorte qu'il soit permis à l'identité authentifiée "sgw23.exemple.com" de créer des SA filles pour 192.0.2.0/24, ce qui signifie que cette passerelle de sécurité est un "représentant" valide pour ces adresses. IPsec d'hôte à hôte exige des entrées similaires, reliant, par exemple, "fooserveur4.exemple.com" avec 198.51.100.66/32, signifiant que cette identité est un "propriétaire" ou "représentant" valide de l'adresse en question.

Comme il est noté dans la [RFC4301], "Il est nécessaire d'imposer ces contraintes sur la création des SA filles pour empêcher un homologue authentifié d'usurper des identifiants associés à d'autres homologues légitimes". Dans l'exemple donné ci-dessus, une configuration correcte de la PAD empêchera sgw23 de créer des SA filles avec l'adresse 198.51.100.66, et empêche fooserveur4 de créer des SA filles avec des adresses de 192.0.2.0/24.

Il est important de noter que simplement envoyer des paquets IKEv2 en utilisant une certaine adresse n'implique pas la permission de créer des SA filles avec cette adresse dans les sélecteurs de trafic. Par exemple, même si sgw23 était capable de contrefaire son adresse IP comme étant 198.51.100.66, il ne pourrait pas créer des SA filles correspondant au trafic de fooserveur4.

La spécification IKEv2 ne spécifie pas exactement comment l'allocation d'adresse IP en utilisant des charges utiles Configuration interagit avec la PAD. Notre interprétation est que quand une passerelle de sécurité alloue une adresse en utilisant des charges utiles Configuration, elle crée aussi une entité temporaire de PAD reliant l'identité authentifiée de l'homologue et l'adresse interne nouvellement allouée.

Il a été reconnu que configurer correctement la PAD peut être difficile dans certains environnements. Par exemple, si IPsec est utilisé entre une paire d'hôtes dont les adresses sont allouées dynamiquement en utilisant DHCP, il est extrêmement difficile de s'assurer que la PAD spécifie le "propriétaire" correct pour chaque adresse IP. Cela exigerait un mécanisme pour porter en toute sécurité les allocations d'adresse depuis le serveur DHCP, et de les relier aux identités authentifiées qui utilisent IKEv2.

Du fait de ces limitations, certains fabricants sont connus pour configurer leurs PAD à permettre aux homologues authentifiés de créer des SA filles avec des sélecteurs de trafic contenant la même adresse qu'utilisé pour les paquets IKEv2. Dans des

environnements où l'usurpation d'identité IP est possible (c'est-à-dire, presque partout) cela permet essentiellement à tout homologue de créer des SA filles avec tout sélecteur de trafic. Ceci n'est pas une configuration appropriée ou sûre dans la plupart des circonstances. Voir dans [H2HIPSEC] une discussion extensive de cette question, et les limitations de IPsec d'hôte à hôte en général.

6. Considérations relatives à l'IANA

La [RFC4306] a défini de nombreux types et valeurs de champs. L'IANA a déjà enregistré ces types et ces valeurs dans [IKEV2IANA], de sorte qu'ils ne sont pas répétés ici.

Un élément du registre "IKEv2 Certificate Encodings" a été déconseillé : "Raw RSA Key".

L'IANA a mis à jour toutes les références à la RFC 5996 pour pointer sur le présent document.

7. Références

7.1 Références normatives

[IKEV2IANA] IANA, "Internet Key Exchange Version 2 (IKEv2) Parameters", < http://www.iana.org/assignments/ikev2-parameters/ >.

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

[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. ; MàJ par* RFC8311)

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

[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 (EAP)", juin 2004. *(P.S., MàJ par* RFC5247)

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

[RFC3986] T. Berners-Lee, R. Fielding et L. Masinter, "Identifiant de ressource uniforme (URI) : Syntaxe générique", STD 66, janvier 2005.

[RFC4291] R. Hinden, S. Deering, "Architecture d'adressage IP version 6", février 2006. (*MàJ par* 5952 et 6052, 8064) *(D.S.)*

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

[RFC4307] J. Schiller, "Algorithmes cryptographiques à utiliser avec la version 2 de l'échange de clés sur Internet (IKEv2)", décembre 2005. *(P.S. ; obsolète, voir* RFC8247)

[RFC4434] P. Hoffman, "Algorithme AES-XCBC-PRF-128 pour le protocole d'échange de clés Internet (IKE)", février 2006. *(P.S.)*

[RFC4615] J. Song et autres, "Algorithme évolué de la fonction 128 de code pseudo aléatoire d'authentification de message fondée sur le chiffrement standard (AES-CMAC-128) pour le protocole d'échange de clés sur Internet (IKE)", août 2006. *(P.S.)*

[RFC5280] D. Cooper et autres, "Profil de certificat d'infrastructure de clé publique X.509 et de liste de révocation de certificat (CRL) pour l'Internet", mai 2008. (*Remplace les* RFC3280, RFC4325, RFC4630) *(P.S. ; MàJ par* RFC8398, 8399)

[RFC5282] D. Black, D. McGrew, "Utilisation des algorithmes de chiffrement authentifiés avec la charge utile du protocole d'échange de clé Internet version 2 (IKEv2)", août 2008. (*MàJ* RFC4306) *(P.S.)*

7.2 Références pour information

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

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

[DOS] Kaufman, C., Perlman, R., et B. Sommerfeld, "DoS protection for UDP-based protocols", ACM Conference on Computer and Communications Security, octobre 2003.

[DSS] National Institute of Standards and Technology, "Digital Signature Standard (DSS)", FIPS 186-4, juillet 2013, <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf >.

[EAP-IANA] IANA, "Extensible Authentication Protocol (EAP) Registry: Method Types", <http://http://www.iana.org/assignments/eap-eke/ >.

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

[ECH] Perlman, R. et C. Kaufman, "Analysis of the IPsec key exchange Standard", WET-ICE Security Conference, MIT, 2001, < http://www.computer.org/csdl/proceedings/wetice/2001/1269/00/12690150.pdf >.

[FIPS.180-4.2012] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS 180-4, mars 2012, <http://csrc.nist.gov/publications/fips/fips180-4fips-180-4.pdf >.

[H2HIPSEC] Aura, T., Roe, M., et A. Mohammed, "Experiences with Host-to-Host IPsec", 13th International Workshop on Security Protocols, Cambridge, UK, avril 2005.

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

[MODES] Dworkin, M., "Recommendation for Block Cipher Modes of Operation", National Institute of Standards and Technology, publication spéciale du NIST 800-38A, édition 2001, décembre 2001.

[REUSE] Menezes, A. et B. Ustaoglu, "On Reusing Ephemeral Keys In Diffie-Hellman Key Agreement Protocols", décembre 2008, < http://www.cacr.math.uwaterloo.ca/techreports/2008/cacr2008-24.pdf >.

[RFC0791] J. Postel, éd., "Protocole Internet - Spécification du protocole du programme Internet", STD 5, septembre 1981.

[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.

[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, "Protocole d'association de sécurité et gestion de clés Internet (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. *(P.S. ; MàJ par* RFC3168, RFC3260, RFC8436)

[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)*

[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)*

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

[RFC3810] R. Vida, L. Costa, éditeurs, "Découverte d'écouteur de diffusion groupée version 2 (MLDv2) pour IPv6", juin 2004.

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

[RFC4282] B. Aboba et autres, "L'identifiant d'accès réseau", décembre 2005. *(P.S., Remplacée par* RFC7542)

[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.)*

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

[RFC4478] Y. Nir, "Authentification répétée dans le protocole d'échange de clés Internet (IKEv2)", avril 2006. *(Expérimentale)*

[RFC4555] P. Eronen, "Protocole IKEv2 de mobilité et de rattachement multiple (MOBIKE)", juin 2006. *(P.S.)*

[RFC4718] P. Eronen, P. Hoffman, "Précisions et lignes directrices pour la mise en œuvre de IKEv2", octobre 2006. *(Info.)*

[RFC4945] B. Korver, "Profil Internet de PKI de sécurité IP de IKEv1/ISAKMP, IKEv2, et PKIX", août 2007. *(P.S.)*

[RFC5322] P. Resnick, éd., "Format du message Internet", octobre 2008. (*Remplace* RFC2822) (*MàJ* RFC4021) *(D.S.)*

[RFC5739] P. Eronen, J. Laganier, C. Madson, "Configuration IPv6 dans le protocole d'échange de clés Internet version 2 (IKEv2)", février 2010. *(Expérimentale)*

[RFC5857] E. Ertekin, C. Christou, R. Jasani, T. Kivinen, C. Bormann, "Extensions à IKEv2 pour la prise en charge de la compression d'en-tête robuste sur IPsec", mai 2010. *(P. S.)*

[RFC5890] J. Klensin, "Noms de domaine internationalisés pour les applications (IDNA) : Définitions et cadre documentaire", août 2010. *(Remplace RFC*3490) *(P.S.)*

[RFC5996] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen, "Protocole d'échange de clés sur Internet, version 2 (IKEv2)", septembre 2010 (*Remplace* RFC4306, RFC4718 ; *MàJ par* RFC5998, *rendue obsolète par la présente RFC.)*

[RFC6275] C. Perkins, D. Johnson, J. Arkko, "Prise en charge de la mobilité dans IPv6", juillet 2011. *(P.S. ; Remplace la*

RFC3775)

[RFC6532] A. Yang, S. Steele, N. Freed, "En-têtes de messagerie électronique internationalisée", février 2012. (*Remplace la* RFC5335) (*MàJ la* RFC2045) *(P.S.)*

[RFC6989] Y. Sheffer, S. Fluhrer, "Essais Diffie-Hellman supplémentaires pour le protocole d'échange de clés Internet version 2 (IKEv2)", juillet 2013. (*MàJ* RFC5996) *(P.S.)*

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

[SKEME] Krawczyk, H., "SKEME: A Versatile Secure Key Exchange Mechanism for Internet", IEEE Proceedings of the 1996 Symposium on Network and Distributed Systems Security, 1996.

Appendice A. Résumé des changements par rapport à IKEv1

Les buts de cette révision de IKE sont :

1. de définir le protocole IKE entier dans un seul document, remplaçant les RFC 2407, 2408, et 2409 et incorporant les changements ultérieurs pour prendre en charge la traversée de NAT, l'authentification extensible, et l'acquisition d'adresse à distance ;

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 affectant seulement une charge utile AUTH plutôt que de restructurer l'échange complet) voir [ECH] ;

3. de retirer les champs Domaine d'interprétation (DOI), Situation (SIT), et Identifiant de domaine étiqueté, et les bits "Commit" et "Authentification seulement" ;

4. de diminuer la latence de IKE dans le cas courant en faisant que l'échange initial soit de deux allers-retours (4 messages), et en permettant de porter l'établissement d'une SA fille sur cet échange ;

5. de remplacer la syntaxe cryptographique de la protection des messages IKE eux-mêmes par une 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 (tous les messages font l'objet d'un accusé de réception) et séquencé. Cela permet de raccourcir les échanges CREATE_CHILD_SA de trois messages à deux ;

7. d'augmenter la robustesse en permettant au répondant de ne pas faire de traitement significatif avant qu'il ait reçu un message prouvant que l'initiateur peut recevoir les messages à l'adresse IP qu'il prétend avoir ;

8. de corriger des faiblesses cryptographiques telles que le problème des symétries dans les hachages utilisés pour l'authentification (documenté par Tero Kivinen) ;

9. de spécifier des sélecteurs de trafic à la place de leur type de charges utiles plutôt que de surcharger les charges utiles ID, et de rendre plus flexibles les sélecteurs de trafic qui peuvent être spécifiés ;

10. de spécifier le comportement requis dans certaines conditions d'erreur ou quand des données qui ne sont pas comprises sont reçues afin de faciliter de futures révisions d'une façon qui ne rompe pas la rétro compatibilité ;

11. de simplifier et préciser comment l'état partagé est conservé en présence de défaillances du réseau et d'attaques de déni de service ;

12. de conserver la syntaxe et les numéros magiques existants dans la mesure du possible pour rendre probable que les mises en œuvre de IKEv1 puissent être améliorées pour prendre en charge IKEv2 avec un minimum d'efforts.

Appendice B. Groupes Diffie-Hellman

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

La force fournie par le groupe 1 peut n'être pas suffisante pour les utilisations normales et n'est 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

L'identifiant 1 (un) est alloué à ce groupe.

Le nombre premier est: 2^768 - 2 ^704 - 1 + 2^64 * { [2^638 pi] + 149686 }. 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 A63A3620 FFFFFFFF FFFFFFFF

Le générateur est 2.

B.2. Groupe 2 - MODP à 1024 bits

L'identifiant 2 (deux) est alloué à ce groupe.

Le nombre premier 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.

Appendice C. Échanges et charges utiles

Cet appendice contient un bref résumé des échanges IKEv2, et des charges utiles qui peuvent apparaître dans un message. Cet appendice est purement informatif ; en cas de désaccord avec le corps du présent document, c'est celui du corps qui est considéré comme correct.

Les charges utiles Identifiant de fabricant (V) peuvent être incluses à tout endroit de tout message. Les séquences montrées ici donnent ce qui est la place la plus logique pour elles.

C.1 Échange IKE_SA_INIT

```
demande             --> [N(COOKIE),]
                        SA, KE, Ni,
                        [N(NAT_DETECTION_SOURCE_IP)+,
                         N(NAT_DETECTION_DESTINATION_IP),]
                        [V+][N+]

réponse normale     <-- SA, KE, Nr,
  (pas de cookie)           [N(NAT_DETECTION_SOURCE_IP),
                         N(NAT_DETECTION_DESTINATION_IP),]
                        [[N(HTTP_CERT_LOOKUP_SUPPORTED),] CERTREQ+,]
                        [V+][N+]

réponse cookie      <-- N(COOKIE),
```

```
                    [V+][N+]

veut un groupe Diffie-Hellman différent  <-- N(INVALID_KE_PAYLOAD),
                    [V+][N+]
```

C.2 Échange IKE_AUTH sans EAP

```
demande             --> IDi, [CERT+,]
                    [N(INITIAL_CONTACT),]
                    [[N(HTTP_CERT_LOOKUP_SUPPORTED),] CERTREQ+,]
                    [IDr,]
                    AUTH,
                    [CP(CFG_REQUEST),]
                    [N(IPCOMP_SUPPORTED)+,]
                    [N(USE_TRANSPORT_MODE),]
                    [N(ESP_TFC_PADDING_NOT_SUPPORTED),]
                    [N(NON_FIRST_FRAGMENTS_ALSO),]
                    SA, TSi, TSr,
                    [V+][N+]

réponse             <-- IDr, [CERT+,]
                    AUTH,
                    [CP(CFG_REPLY),]
                    [N(IPCOMP_SUPPORTED),]
                    [N(USE_TRANSPORT_MODE),]
                    [N(ESP_TFC_PADDING_NOT_SUPPORTED),]
                    [N(NON_FIRST_FRAGMENTS_ALSO),]
                    SA, TSi, TSr,
                    [N(ADDITIONAL_TS_POSSIBLE),]
                    [V+][N+]

erreur à la création de SA fille  <--  IDr, [CERT+,]
                    AUTH,
                    N(error),
                    [V+][N+]
```

C.3 Échange IKE_AUTH avec EAP

```
première demande    --> IDi,
                    [N(INITIAL_CONTACT),]
                    [[N(HTTP_CERT_LOOKUP_SUPPORTED),] CERTREQ+,]
                    [IDr,]
                    [CP(CFG_REQUEST),]
                    [N(IPCOMP_SUPPORTED)+,]
                    [N(USE_TRANSPORT_MODE),]
                    [N(ESP_TFC_PADDING_NOT_SUPPORTED),]
                    [N(NON_FIRST_FRAGMENTS_ALSO),]
                    SA, TSi, TSr,
                    [V+][N+]

première réponse    <-- IDr, [CERT+,] AUTH,
                    EAP,
                    [V+][N+]

                  / --> EAP
répété 1..N fois  |
                  \ <-- EAP

dernière demande    --> AUTH
```

```
dernière response      <-- AUTH,
                           [CP(CFG_REPLY),]
                           [N(IPCOMP_SUPPORTED),]
                           [N(USE_TRANSPORT_MODE),]
                           [N(ESP_TFC_PADDING_NOT_SUPPORTED),]
                           [N(NON_FIRST_FRAGMENTS_ALSO),]
                           SA, TSi, TSr,
                           [N(ADDITIONAL_TS_POSSIBLE),]
                           [V+][N+]
```

C.4 Échange CREATE_CHILD_SA pour créer ou changer les clés des SA filles

```
demande          --> [N(REKEY_SA),]
                     [CP(CFG_REQUEST),]
                     [N(IPCOMP_SUPPORTED)+,]
                     [N(USE_TRANSPORT_MODE),]
                     [N(ESP_TFC_PADDING_NOT_SUPPORTED),]
                     [N(NON_FIRST_FRAGMENTS_ALSO),]
                     SA, Ni, [KEi,] TSi, TSr,
                     [V+][N+]

réponse normale          <-- [CP(CFG_REPLY),]
                     [N(IPCOMP_SUPPORTED),]
                     [N(USE_TRANSPORT_MODE),]
                     [N(ESP_TFC_PADDING_NOT_SUPPORTED),]
                     [N(NON_FIRST_FRAGMENTS_ALSO),]
                     SA, Nr, [KEr,] TSi, TSr,
                     [N(ADDITIONAL_TS_POSSIBLE),]
                     [V+][N+]

cas d'erreur         <-- N(error)

veut un groupe Diffie-Hellman différent  <-- N(INVALID_KE_PAYLOAD),
                     [V+][N+]
```

C.5 Échange CREATE_CHILD_SA pour le changement de clé de SA IKE

```
demande          --> SA, Ni, KEi,
                     [V+][N+]

réponse          <-- SA, Nr, KEr,
                     [V+][N+]
```

C.6 Échange INFORMATIONAL

```
demande          --> [N+,]
                     [D+,]
                     [CP(CFG_REQUEST)]

réponse          <-- [N+,]
                     [D+,]
                     [CP(CFG_REPLY)]
```

Remerciements

De nombreuses personnes du groupe de travail IPsecME ont apporté une grande aide en contribuant par leurs idées et des textes au présent document, ainsi qu'en relisant les éclaircissements suggérés par d'autres.

Les remerciements pour le document IKEv2 étaient : Le présent document est un effort collaboratif du groupe de travail IPsec tout entier. Si il n'y avait pas de limite au nombre des auteurs qui peut apparaître sur une RFC, les personnes suivantes, par ordre alphabétique, y auraient leur place : 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. C'est une évolution de IKEv1, ISAKMP, et du DOI IPsec, dont chacun a sa propre liste d'auteurs. Hugh Daniel a suggéré la caractéristique d'avoir l'initiateur, dans le message 3, qui spécifie un nom pour celui qui répond, et a donné à cette caractéristique le joli nom de "Toi Tarzan, Moi Jane". David Faucher et Valery Smyslov ont aidé à rafiner le concept de négociation de sélecteur de trafic.

Adresse des auteurs

Charlie Kaufman
Microsoft
1 Microsoft Way
Redmond, WA 98052
United States of America
mél : charliekaufman@outlook.com

Paul Hoffman
VPN Consortium
127 Segre Place
Santa Cruz, CA 95060
United States of America
téléphone : 1-831-426-9827
mél : paul.hoffman@vpnc.org

Yoav Nir
Check Point Software Technologies Ltd.
5 Hasolelim St.
Tel Aviv 6789735
Israel
mél : ynir.ietf@gmail.com

Pasi Eronen
Indépendant
mél : pe@iki.fi

Tero Kivinen
INSIDE Secure
Eerikinkatu 28
HELSINKI FI-00180
Finland
mél : kivinen@iki.fi