Internet Engineering Task Force (IETF)
Request for Comments : 7296
STD 79
RFC rendues obsolètes : 5996
Catégorie : Norme
ISSN: 2070-1721

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

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

Résumé
Le présent document décrit la version 2 du protocole d'échange de clé Internet (IKE, *Internet Key Exchange*). IKE est un composant de IPsec utilisé pour effectuer l'authentification mutuelle et établir et maintenir les associations de sécurité (SA, *Security Association*). Le présent document rend obsolète la RFC 5996 et inclut tous ses errata. Il avance IKEv2 au statut de norme de l'Internet.

Statut de ce mémoire
Ceci est un document de l'Internet sur la voie de la normalisation.

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). Tous les documents approuvés par l'IESG ne sont pas candidats à devenir une norme de l'Internet ; voir 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

La sécurité IP (IPsec) assure 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 l'état partagé entre la source et le débouché d'un datagramme IP. Cet état définit, entre autres choses, les services spécifiques fournis au datagramme, quels algorithmes de chiffrement vont être utilisés pour fournir les services, et les clés utilisées comme entrées aux algorithmes de chiffrement.

Établir cet état partagé de façon manuelle ne convient pas bien. Donc, un protocole pour établir cet état de façon dynamique est nécessaire. Le présent document décrit ce protocole – l'échange de clés 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 dans la [RFC4718]. La [RFC5996] a remplacé et mis à jour les RFC 4306 et 4718. Le présent document remplace la RFC 5996. IKEv2 tel que décrit dans la RFC 4306 était un changement au protocole IKE qui n'était pas rétro-compatible. La RFC 5996 révisait la RFC 4306 pour fournir des éclaircissements à IKEv2, faisant des changements minimaux au protocole IKEv2. Le présent document remplace la RFC 5996, la révisant légèrement pour qu'il soit convenable de la passer au statut de norme de l'Internet. Une liste des différences significatives entre les RFC 4306 et 5996 est donnée au paragraphe 1.7, et les différences entre la RFC 5996 et le présent document sont données au paragraphe 1.8.

IKE effectue l'authentification mutuelle entre deux parties et établit une association de sécurité (SA, Se*curity Association*) IKE qui inclut des informations de secret partagé qui peuvent être utilisées pour établir efficacement les SA pour une charge utile de sécurité encapsulante (ESP, E*ncapsulating Security Payload*) [RFC4303] ou un en-tête d'authentification (AH, *Authentication Header*) [RFC4302] et un ensemble d'algorithmes de chiffrement à utiliser par les SA pour protéger le trafic qu'elles transportent. Dans ce document, le terme "suite" ou "suite de chiffrement" 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 dans des suites mélangées. IKE peut aussi négocier l'utilisation de la compression IP (IPComp) [RFC3173] en connexion avec une SA ESP ou AH. Les SA pour ESP ou AH qui sont établies à travers cette SA IKE sont appelées des "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 et l'échange IKE_AUTH ; les échanges IKE suivants sont appelés soit des échanges CREATE_CHILD_SA, soit des échanges INFORMATIONAL. Dans le cas courant, il y a un seul échange IKE_SA_INIT et un seul échange IKE_AUTH (un total de quatre messages) pour établir la SA IKE et la première SA fille. Dans des cas exceptionnels, il peut y avoir plus d'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, ensuite tous les échanges IKE_AUTH DOIVENT s'achever, et après cela, un nombre quelconque d'échanges CREATE_CHILD_SA et INFORMATIONAL peuvent se produire dans n'importe quel ordre. Dans certains scénarios, seulement une SA fille est nécessaire entre les points d'extrémité IPsec, et donc il n'y a pas d'échange supplémentaire. Les échanges suivants PEUVENT être utilisés pour établir des SA filles supplémentaires entre la même paire de points d'extrémité authentifiés et pour effectuer des fonctions de ménage.

Un flux de messages IKE consiste toujours en une demande suivie d'une réponse. Il est de la responsabilité du demandeur d'assurer la fiabilité. Si la réponse n'est pas reçue dans l'intervalle 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, et envoie les valeurs Diffie-Hellman.

Le second échange, IKE_AUTH, transmet les identités, prouve la connaissance des secrets correspondants aux deux identités, et établit une SA pour la première (et souvent la seule) SA fille AH ou ESP (sauf si il y a une défaillance dans l'établissement de la SA fille AH ou ESP, et dans ce cas la SA IKE est quand même établie sans la SA fille).

Les types des échanges suivants sont CREATE_CHILD_SA (qui crée une SA fille) et INFORMATIONAL (qui supprime une SA, rapporte des conditions d'erreur, ou fait d'autres tâches ménagères). Chaque demande exige une réponse. Une demande INFORMATIONAL sans charge utile (autre que la charge utile Chiffrée vide exigée par la syntaxe) est couramment utilisée comme preuve de vie. Ces échanges suivants ne peuvent pas être utilisés tant que les échanges initiaux ne sont pas achevés.

Dans la description qui suit, on suppose qu'il ne se produit pas d'erreur. Les modifications au flux quand des erreurs se

produisent sont décrites au paragraphe 2.21.

1.1 Scénarios d'usage

IKE est utilisé pour négocier des SA ESP ou AH dans un certain nombre de scénarios différents, chacun avec ses propres exigences particulières.

1.1.1 Passerelle de sécurité à passerelle de sécurité en mode tunnel

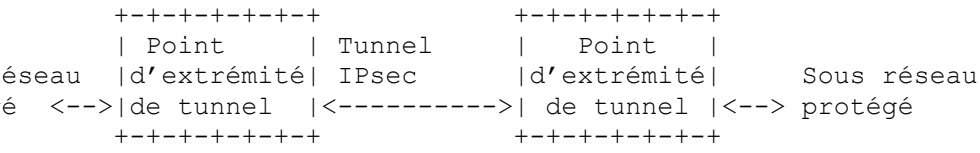


Figure 1 : Tunnel de passerelle de sécurité à passerelle de sécurité

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 pour une part 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 va annoncer l'ensemble d'adresses "derrière" lui, et les paquets vont être envoyés en mode tunnel lorsque l'en-tête IP interne contient les adresses IP des points d'extrémité réels.

1.1.2 Mode de transport de point d'extrémité à point d'extrémité

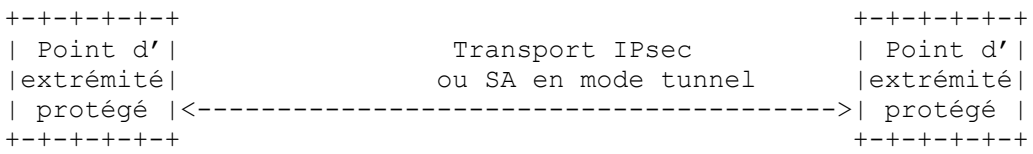


Figure 2 : Point d'extrémité à point d'extrémité

Dans ce scénario, les deux points d'extrémité de la connexion IP mettent en œuvre IPsec, comme exigé des hôtes dans la [RFC4301]. Le mode transport va être couramment utilisé en l'absence d'en-tête IP interne. Une seule paire d'adresses va être négociée pour les paquets à protéger par cette SA. Ces points d'extrémité PEUVENT mettre en œuvre des contrôles d'accès de couche d'application fondés sur les identités authentifiées par IPsec des participants. Ce scénario permet la sécurité de bout en bout qui a été le principe directeur de l'Internet depuis la [RFC1958], la [RFC2775], et une méthode de limitation des problèmes inhérents à la complexité des réseaux notée par la [RFC3439]. Bien que ce scénario puisse n'être pas pleinement applicable à l'Internet IPv4, il a été déployé avec succès dans des scénarios spécifiques au sein d'intranets utilisant IKEv1. Il devrait être plus largement activé durant la transition à IPv6 et avec l'adoption de IKEv2.

Il est possible dans ce scénario qu'un des points d'extrémité protégés ou les deux soient derrière un nœud de traduction d'adresse réseau (NAT, *Network Address Translation*) et dans ce cas, les paquets tunnelés vont devoir être encapsulés dans UDP afin que les numéros d'accès dans les en-têtes UDP puissent être utilisés pour identifier les points d'extrémité individuels "derrière" le NAT (voir le paragraphe 2.23).

1.1.3 Point d'extrémité à passerelle de sécurité en mode tunnel

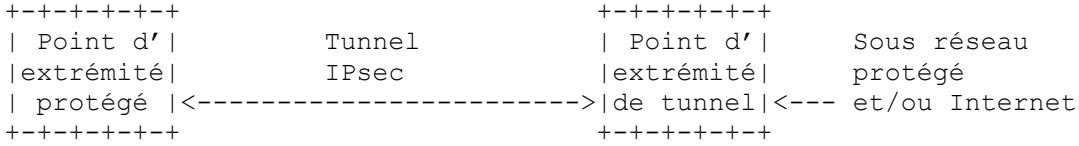


Figure 3 : Tunnel de point d'extrémité à passerelle de sécurité

Dans ce scénario, un point d'extrémité protégé (typiquement un ordinateur portable en itinérance) se connecte à son réseau

d'entreprise à travers un tunnel protégé par IPsec. Il pourrait utiliser ce tunnel seulement pour accéder à des informations sur le réseau d'entreprise, ou il pourrait tunneler tout son trafic à travers 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 l'un et l'autre cas, le point d'extrémité protégé va vouloir une adresse IP associée à la passerelle de sécurité afin que les paquets qui lui sont retournés passent par la passerelle de sécurité et lui soient re-tunnelés. Cette adresse IP peut être statique ou peut être allouée dynamiquement par la passerelle de sécurité. En soutien de ce dernier cas, IKEv2 inclut un mécanisme (à savoir, les charges utiles 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 va contenir l'adresse IP de source associée à sa localisation actuelle (c'est-à-dire, l'adresse qui va avoir le trafic acheminé directement au point d'extrémité) tandis que l'en-tête IP interne va contenir l'adresse IP de source allouée par la passerelle de sécurité (c'est-à-dire, l'adresse qui va obtenir le trafic acheminé à la passerelle de sécurité pour transmission au point d'extrémité). L'adresse de destination externe va toujours être celle de la passerelle de sécurité, tandis que l'adresse de destination interne va être 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 vue par la passerelle de sécurité ne va pas être la même que l'adresse IP envoyée par le point d'extrémité protégé, et les paquets vont devoir être encapsulés dans UDP afin d'être acheminés correctement. L'interaction avec les NAT est traitée en détail au paragraphe 2.23.

1.1.4 Autres scénarios

D'autres scénarios sont possibles, comme le sont des combinaisons 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, où les adresses sur le sous réseau sont acheminées à la passerelle de sécurité par le reste de l'Internet. Un exemple serait le réseau de rattachement de quelqu'un qui serait virtuellement sur l'Internet avec des adresses IP statiques bien que la connexité soit fournie par un FAI qui alloue de façon dynamique une seule adresse IP à la passerelle de sécurité de l'utilisateur (où les adresses IP statiques et un relais IPsec sont fournis par un tiers situé ailleurs).

1.2 Échanges initiaux

La communication utilisant IKE commence 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 croître. Toutes les communications qui utilisent IKE consistent en paires de demande/réponse. On va décrire d'abord l'échange de base, et ensuite les variantes. La première paire de messages (IKE_SA_INIT) négocie les algorithmes de chiffrement, échange les noms occasionnels, et fait un échange Diffie-Hellman [DH].

La seconde paire de messages (IKE_AUTH) authentifie les messages précédents, échange les identités et les certificats, et établit la première SA fille. Des parties de ces messages sont chiffrées et protégées en intégrité avec les clés établies par l'échange IKE_SA_INIT, afin que les identités soient 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 interposé qui ne peut pas achever l'échange IKE_AUTH peut néanmoins voir l'identité de l'initiateur.)

Tous les messages qui suivent l'échange initial sont protégés cryptographiquement en utilisant les algorithmes de chiffrement et les clés négociées dans l'échange IKE_SA_INIT. Ces messages suivants utilisent la syntaxe de charge utile Chiffrée décrite au paragraphe 3.14, chiffrés avec les clés qui sont déduites comme décrit au paragraphe 2.14. Tous les messages suivants incluent une charge utile Chiffrée, même si elles sont mentionnées 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, y compris l'en-tête, est protégé en intégrité en utilisant les algorithmes de chiffrement 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 faire correspondre les demandes et les réponses, et pour identifier les retransmissions de messages.

Dans les descriptions qui suivent, les charges utiles contenues dans le message sont indiquées par les noms indiqués ci-dessous :

Notation	**Charge utile**
AUTH	Authentification

CERT	Certificat
CERTREQ	Demande de certificat
CP	Configuration
D	Suppression
EAP	Authentification extensible
HDR	En-tête IKE (pas une charge utile)
IDi	Identification - initiateur
IDr	Identification - répondant
KE	Échange de clé
Ni, Nr	Nom occasionnel
N	Notify
SA	Association de sécurité
SK	Chiffré et authentifié
TSi	Sélecteur de trafic - initiateur
TSr	Sélecteur de trafic - répondant
V	Identifiant de fabricant

Les détails du contenu de chaque charge utile sont décrits à la Section 3. Les charges utiles qui peuvent facultativement apparaître sont montrées entre crochets, comme dans [CERTREQ] ; cela indique qu'une charge utile Demande de certificat peut facultativement être incluse.

Les échanges initiaux sont comme suit :

Initiateur **Répondant**

HDR, SAi1, KEi, Ni -->

HDR contient les indices de paramètre de sécurité (SPI, *Security Parameter Index*) les numéros de version, le type d'échange, l'identifiant de message, et des fanions de diverses sortes. La charge utile SAi1 déclare les algorithmes de chiffrement que l'initiateur prend en charge pour la SA IKE. 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]

Celui qui répond choisit une suite cryptographique dans les choix offerts par l'initiateur et exprime ce choix dans la charge utile SAr1, il 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 le paragraphe 2.14) à partir de laquelle toutes les clés sont déduites pour cette SA IKE. Les messages qui suivent sont entièrement chiffrés et protégés en intégrité, à l'exception des en-têtes de message. Les clés utilisées pour le chiffrement et la protection de l'intégrité sont déduites de SKEYSEED et sont appelées SK_e (chiffrement) et SK_a (authentification, autrement dit, protection de l'intégrité) ; voir aux paragraphes 2.13 et 2.14 les détails de la déduction de clé. Des SK_e et SK_a séparées sont calculées pour chaque direction. En plus des clés SK_e et SK_a déduites de la valeur Diffie-Hellman pour la protection de la SA IKE, une autre quantité SK_d est déduite et utilisée pour la déduction du matériel de chiffrement pour les SA filles. La notation SK { ... } indique que ces charges utiles sont chiffrées et protégées en intégrité en utilisant ces SK_e et SK_a de 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 l'intégrité du contenu du premier message en utilisant la charge utile AUTH (voir le paragraphe 2.15). Il pourrait aussi envoyer son ou ses certificats dans la charge utile CERT et une liste de ses ancres de confiance dans la ou les charges utiles CERTREQ. Si 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 quelles identités de celui qui répond il veut parler. Ceci est utile quand la machine chez celui qui répond héberge plusieurs identités à la même adresse IP. Si le IDr proposé par l'initiateur n'est pas acceptable pour celui qui répond, celui-ci pourrait utiliser d'autres IDr pour finir l'échange. Si l'initiateur n'accepte pas alors le fait que celui qui répond utilise un IDr différent de celui qui était demandé, l'initiateur peut

clore la SA après avoir noté 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 (en commençant par SAi2) sont décrits dans l'échange CREATE_CHILD_SA.

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

Celui qui répond affirme son identité avec la charge utile IDr, envoie facultativement un ou plusieurs certificats (là encore, avec mentionné en premier le certificat contenant la clé publique utilisée pour vérifier AUTH) 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 dans 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 ou l'autre des côtés utilise un secret partagé pour l'authentification, les noms dans l'identifiant de charge utile 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, il doit deviner le groupe Diffie-Hellman que celui qui répond va choisir dans sa liste des groupes pris en charge. Si l'initiateur devine mal, celui qui répond va répondre avec une charge utile Notify de type INVALID_KE_PAYLOAD indiquant le groupe choisi. Dans ce cas, l'initiateur DOIT réessayer le IKE_SA_INIT avec le groupe Diffie-Hellman corrigé. L'initiateur DOIT encore proposer son ensemble complet de suites de chiffrement 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 préféreraient.

Si la création de la SA fille échoue durant l'échange IKE_AUTH pour une raison quelconque, la SA IKE est quand même créée comme d'habitude. La liste des types de message Notify dans l'échange IKE_AUTH qui n'empêchent pas une SA IKE d'être établie inclut au moins : NON_PROPOSAL_CHOSEN *(choix non proposé)*, TS_UNACCEPTABLE *(sélecteur de trafic non acceptable)*, SINGLE_PAIR_REQUIRED *(une seule paire exigée)*, INTERNAL_ADDRESS_FAILURE *(échec d'adresse interne)*, et FAILED_CP_REQUIRED *(configuration défaillante exigée)*.

Si la défaillance est relative à la création de la SA IKE (par exemple, un message d'erreur Notify AUTHENTICATION_FAILED 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 Notify d'erreur 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 prudence. Plus précisément, en supposant que le MAC se vérifie correctement, l'envoyeur du message Notify d'erreur est connu pour être celui qui répond 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 charges utiles KEi/KEr ou Ni/Nr. Donc, les charges utiles SA dans l'échange IKE_AUTH ne peuvent pas contenir de transformation de type 4 (groupe Diffie-Hellman) avec toute valeur autre que NONE *(aucune)*. Les mises en œuvre DEVRAIENT omettre toute la sous structure de transformation au lieu d'envoyer la valeur NONE.

1.3. 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 sa fonction était appelée échange de phase 2 dans IKEv1. Il PEUT être initié par l'une ou l'autre extrémité de la SA IKE après l'achèvement des échanges initiaux.

Les clés d'une SA sont changées en créant une nouvelle SA et en supprimant ensuite l'ancienne. Ce paragraphe décrit la première partie du changement de clé, la création des nouvelles SA ; le paragraphe 2.8 traite des mécanismes de changement de clé, incluant de déplacer le trafic de l'ancienne à la nouvelle SA et la suppression de l'ancienne SA. Les deux paragraphes doivent être lus ensemble pour comprendre le processus complet de changement de clé.

L'un et l'autre des points d'extrémité peut initier un échange CREATE_CHILD_SA, donc dans ce paragraphe, le terme initiateur se réfère au point d'extrémité qui initie cet échange. Une mise en œuvre PEUT refuser toutes les demandes CREATE_CHILD_SA dans une SA IKE.

La demande CREATE_CHILD_SA PEUT facultativement contenir une charge utile KE pour un échange Diffie-Hellman

supplémentaire pour permettre de plus fortes garanties de secret de transmission pour la SA fille. Le matériel de chiffrement pour la SA fille est une fonction de SK_d établie durant l'établissement de la SA IKE, 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 la 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 à ce que celui qui répond acceptera (des groupes Diffie-Hellman supplémentaires peuvent être proposés). Si celui qui répond choisit une proposition utilisant un groupe Diffie-Hellman différent (autre que NONE) celui qui répond 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 associées à cette notification : le numéro du groupe Diffie-Hellman accepté en ordre gros boutien. En cas de 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 celui qui répond a donné dans la charge utile Notify INVALID_KE_PAYLOAD.

Celui qui répond envoie une notification NON_ADDITIONAL_SAS pour indiquer qu'une demande CREATE_CHILD_SA est inacceptable parce que celui qui répond ne veut pas accepter plus de 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 initial IKE et rejeter toutes les tentatives suivantes d'en ajouter.

1.3.1 Création de nouvelles SA filles avec l'échange CREATE_CHILD_SA

Une SA fille peut être créée en envoyant une demande CREATE_CHILD_SA. La demande CREATE_CHILD_SA pour créer une nouvelle SA fille est :

Initiateur **Répondant**

HDR, SK {SA, Ni, [KEi,] TSi, TSr} -->

L'initiateur envoie 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.

La réponse CREATE_CHILD_SA pour créer une nouvelle SA fille est :

<-- HDR, SK {SA, Nr, [KEr,] TSi, TSr}

Celui qui répond 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 de la réponse, qui peuvent être un sous ensemble de ce que l'initiateur de la SA fille proposait.

La notification USE_TRANSPORT_MODE PEUT être incluse dans un message de demande qui inclut aussi une charge utile 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 celui qui répond refuse la demande, la SA fille va être établie en mode tunnel. Si c'est inacceptable pour l'initiateur, l'initiateur DOIT supprimer la SA. Note : sauf 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 les paquets qui contiennent du bourrage de confidentialité du flux de trafic (TFC, *Traffic Flow Confidentiality*) sur la SA fille négociée. Si aucun des points d'extrémité n'accepte le bourrage de TFC, cette notification est incluse dans la demande et la réponse. Si cette notification est incluse seulement dans un des messages, le bourrage de TFC peut quand même être envoyé dans l'autre direction.

La notification NON_FIRST_FRAGMENTS_ALSO est utilisée pour le contrôle de fragmentation. Voir les explications

dans la [RFC4301]. Les deux parties doivent s'accorder pour envoyer des fragments non premiers avant qu'une 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 la réponse qui l'accepte. Si celui qui répond ne veut pas envoyer ou recevoir des fragments non premiers, il omet seulement la notification NON_FIRST_FRAGMENTS_ALSO de sa réponse, mais ne rejette pas la création de la SA fille entière.

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

Un échec de la tentative de création de 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 de messages d'erreur qui pourraient survenir si la création d'une SA fille échoue.

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

La demande CREATE_CHILD_SA pour changer la clé d'une SA IKE 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 nouveau SPI d'initiateur est fourni dans le champ SPI de la charge utile SA. Une fois qu'un homologue reçoit une demande de changer la clé d'une SA IKE ou envoie 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 dont la clé est en cours de changement.

La réponse CREATE_CHILD_SA pour le changement de clé d'une SA IKE est :

<-- HDR, SK {SA, Nr, KEr}

Celui qui répond 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 de chiffrement choisie inclut ce groupe. Un nouveau SPI de répondant est fourni dans le champ SPI de la charge utile SA.

La nouvelle SA IKE a son compteur de messages réglé à 0, sans considération de ce qu'il était dans la SA IKE antérieure. Les premières demandes IKE provenant des deux côtés sur la nouvelle SA IKE vont avoir l'identifiant de message 0. L'ancienne SA IKE conserve sa numérotation, de sorte que toutes les autres demandes (par exemple, pour supprimer la SA IKE) vont avoir des numéros consécutifs. La nouvelle SA IKE a aussi sa taille de fenêtre remise à 1, et l'initiateur dans cet échange de changement de clé est le nouvel "initiateur original" de la nouvelle SA IKE. Le paragraphe 2.18 traite aussi en détails du changement de clé de SA IKE.

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

La demande CREATE_CHILD_SA de changement de clé 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 clé. Généralement, elles vont être les mêmes notifications que celles utilisées dans l'échange original ; par exemple, quand on change la clé d'une SA en mode transport, la notification USE_TRANSPORT_MODE va être 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 dont les clés sont changées est identifiée par le champ SPI dans la charge

utile Notify ; c'est le SPI que l'initiateur de l'échange attendrait dans les 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 à correspondre au protocole de la SA dont on change la clé, par exemple, 3 pour ESP et 2 pour AH.

La réponse CREATE_CHILD_SA pour le changement de clé d'une SA fille est :

<-- HDR, SK {SA, Nr, [KEr,] TSi, TSr}

Celui qui répond 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 de chiffrement 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 l'initiateur de la SA fille proposait.

1.4 Échange INFORMATIONAL

À divers moments durant le fonctionnement d'une SA IKE, les homologues peuvent désirer porter des messages de contrôle à chaque autre concernant des erreurs ou des notifications de certains événements. Pour réaliser cela, IKE définit un échange INFORMATIONAL. Les échanges INFORMATIONAL DOIVENT seulement se produire après les échanges initiaux et sont cryptographiquement protégés avec les clés négociées. Noter que certains messages d'information, pas des échanges, peuvent être envoyés hors du contexte d'une SA IKE. Le paragraphe 2.21 traite aussi des messages d'erreur en grand détail.

Les messages de contrôle qui relèvent d'une SA IKE DOIVENT être envoyés sous cette SA IKE. Les messages de contrôle qui relèvent des SA filles DOIVENT être envoyés sous la protection de la SA IKE qui les a générées (ou son successeur si la SA IKE a changé de clés).

Les messages dans 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 attendue dont un point d'extrémité peut demander à l'autre point d'extrémité de vérifier qu'il est en vie.

L'échange INFORMATIONAL est défini comme :

Initiateur	**Répondant**
HDR, SK {[N,] [D,] [CP,] ...} -->	
	<-- HDR, SK {[N,] [D,] [CP,] ...}

Le traitement d'un échange INFORMATIONAL est déterminé par ses charges utiles composantes.

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. Quand une SA est close, les deux membres de la paire DOIVENT être clos (c'est-à-dire, supprimés). Chaque point d'extrémité DOIT clore ses SA entrantes et permettre à l'autre point d'extrémité de clore les autres SA dans chaque paire. Pour supprimer une SA, un échange INFORMATIONAL avec une ou plusieurs charges utiles Delete est envoyé en faisant la liste des SPI (comme ils sont supposés être 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 plusieurs 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 des 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 de suppression, il DOIT supprimer les SA sortantes en traitant la demande et les SA entrantes en traitant la réponse. Dans ce

cas, les réponses NE DOIVENT PAS inclure de charges utiles Delete pour les SA supprimées, car cela résulterait en une duplication de la suppression et pourrait théoriquement 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.

Les connexions ESP ou AH semi closes sont anormales, et un nœud avec une capacité d'examen devrait probablement se pencher sur leur existence si elles persistent. Noter que la présente spécification ne spécifie pas de durée, de sorte qu'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 semi closes mais NE DOIT PAS les clore unilatéralement et réutiliser les SPI. Si l'état de connexion devient trop embrouillé, un nœud PEUT clore la SA IKE, comme décrit ci-dessus. Il peut alors reconstruire les SA dont il a besoin sur une base propre sous une nouvelle SA IKE.

1.5 Messages d'information en dehors d'une SA IKE

Il y a des 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 pourrait être dû au fait que le nœud qui reçoit est récemment tombé en panne et a perdu l'état, ou à cause d'un dysfonctionnement ou d'une attaque de l'autre système.

- Si un paquet chiffré de demande IKE arrive sur l'accès 500 ou 4500 avec un SPI IKE non reconnu. Cela pourrait être dû au fait que le nœud qui reçoit est récemment tombé en panne et a perdu l'état, ou à cause d'un dysfonctionnement ou d'une attaque de l'autre système.

- Si un paquet de demande IKE arrive avec numéro de version majeure supérieur à ce que la mise en œuvre supporte.

Dans le premier cas, si le nœud qui reçoit a une SA IKE active à l'adresse IP d'où le paquet est venu, il PEUT envoyer une notification INVALID_SPI du paquet vagabond 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 vont être différents pour les deux. Si aucune SA IKE convenable n'existe, 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, il devrait seulement être utilisé par le receveur comme indication que quelque chose pourrait aller mal (parce qu'il pourrait facilement être falsifié). Ce message ne fait pas partie d'un échange INFORMATIONAL, et le nœud qui reçoit NE DOIT PAS y répondre parce que le faire 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 aussi acceptable, cela étant l'exception à la règle du paragraphe 3.1 qui interdit les SPI d'initiateur IKE à zéro. Le fanion Initiateur est établi 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 et le troisième cas, le message est toujours envoyé sans protection cryptographique (en-dehors d'une SA IKE) et inclut une notification INVALID_IKE_SPI ou INVALID_MAJOR_VERSION (sans données de notification). Le message est un message de réponse, et donc il est envoyé à l'adresse et accès IP d'où il est venu avec les mêmes SPI IKE et l'identifiant de message et le type d'échange sont copiés de la demande. Le fanion Réponse est établi à 1, et les fanions de version sont réglés de façon normale.

1.6 Terminologie des exigences

Les définitions des termes de primitives dans le présent document (comme Association de sécurité ou SA) se trouvent dans la [RFC4301]. On devrait noter que des parties de IKEv2 s'appuient sur certaines des règles de traitement de la [RFC4301], comme décrit dans divers paragraphes du présent 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 sont fondés sur la [RFC4718]. Les changements mentionnés dans ce document ont été discutés dans le groupe de travail IPsec et, après la dissolution du groupe de travail, sur la liste de diffusion IPsec. Le présent document contient des explications détaillées des parties qui n'étaient pas claires dans IKEv2, et est donc utile pour la mise en œuvre de IKEv2.

Le protocole décrit dans le présent document conserve le même numéro de version majeure (2) et de numéro de version mineure (0) qu'utilisé dans la [RFC4306]. C'est-à-dire, le numéro de version *n'est pas* changé par rapport à la RFC 4306. Le petit nombre de changements techniques mentionnés 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 que 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 est accepté de ne pas respecter les exigences. Ils ont aussi noté qu'il y a des exigences de niveau DOIT qui sont sans rapport avec l'interopérabilité. Le présent document donne plus d'explications sur certaines de ces exigences. Toutes les utilisations non en majuscules des mots DEVRAIT et DOIT signifient maintenant leur sens normal , et non le sens d'interopérabilité de la [RFC2119].

Les développeurs de IKEv2 (et IKEv1) ont noté qu'il y a une grande quantité de matériel dans les tableaux de codes dans le paragraphe 3.10.1 de la RFC 4306. Cela conduit les mises en œuvre a ne pas avoir toutes les informations nécessaires dans le corps principal du document. Beaucoup du matériel de ces tableaux a été déplacé dans les parties associées du corps principal du document.

Le présent document supprime la discussion de l'incorporation de AH et ESP. C'était une faute de la RFC 4306 causée par le délai entre la finition de la RFC 4306 et de la RFC 4301. Fondamentalement, IKEv2 se fonde sur la RFC 4301, qui n'inclut pas les "bouquets de SA" qui faisaient 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 transmission. Dans IKEv2, chaque 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 était très problématique. Les mises en œuvre qui se conforment au présent document DOIVENT ignorer les propositions qui ont un attribut de configuration de type 5, l'ancienne valeur pour INTERNAL_ADDRESS_EXPIRY. Le présent document a aussi supprimé INTERNAL_IP6_NBNS comme attribut de configuration. *(Voir erratum n ° 5056)*

Le présent document supprime la tolérance de rejeter les messages dans lesquels les charges utiles n'étaient pas dans le "bon" ordre ; maintenant, les mises en œuvre NE DOIVENT PAS les rejeter. Ceci est dû au manque de clarté avec lequel l'ordre des charges utiles est décrit.

Les listes d'éléments de la RFC 4306 qui figurent dans le registre de l'IANA ont été réduites pour inclure seulement les éléments qui sont réellement définis dans la RFC 4306. Aussi, beaucoup de ces listes sont maintenant précédées de très importantes instructions aux développeurs pour qu'il cherchent dans le registre de l'IANA au moment du développement parce que de nouveaux éléments ont été ajoutés depuis la RFC 4306.

Le présent document ajoute des éclaircissements sur quand les notifications sont et ne sont pas envoyées chiffrées, selon l'état de la négociation à ce moment.

Le présent document développe la discussion sur 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 traitant de comment le SPI et/ou IP de l'initiateur est utilisé 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, quand on change la clé, la nouvelle SA fille PEUT avoir des sélecteurs de trafic et des

algorithmes différents de ceux de l'ancienne" a été changé en "Noter que, quand on change la clé, la nouvelle SA fille NE DEVRAIT PAS avoir des sélecteurs de trafic et algorithmes différents de ceux de l'ancienne".

Le nouveau paragraphe 2.8.2 couvre le changement de clé simultané de SA IKE.

Le présent document ajoute au paragraphe 2.13 la restriction que toutes les fonctions pseudo aléatoires (PRF, *PseudoRandom Function*) utilisées avec IKEv2 DOIVENT prendre des clés de taille variable. Ceci ne devraient pas affecter les mises en œuvre parce que il n'y a pas de PRF normalisée qui ait des clés de taille fixe.

Le paragraphe 2.18 exige de faire un échange Diffie-Hellman quand on change la clé de la SA IKE. En théorie, la RFC 4306 permet une politique où l'échange Diffie-Hellman est facultatif, mais cela n'était pas utile (ou approprié) quand on change la clé de la SA IKE.

Le paragraphe 2.21 a été développé pour couvrir les différents cas où des réponses d'erreur sont nécessaires, et les réponses appropriées à ces réponses.

Le paragraphe 2.23 précise que, dans la traversée de NAT, maintenant aussi bien les paquets IPsec encapsulés dans UDP que ceux non encapsulés dans UDP doivent être compris de celui qui les reçoit.

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

Ajout du paragraphe 2.25 pour expliquer comment agir quand il y a des collisions de temps lors de la suppression et/ou du changement de clé 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 la recherche de hachage et d'URL. Le comportement des autres schémas d'URL [RFC3986] n'est pas spécifié actuellement, et de tels schémas NE DEVRAIENT PAS être utilisés en l'absence d'un document qui les spécifient." a été ajouté.

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

L'Appendice C a été étendu et précisé.

1.8 Différences entre la RFC 5996 et le présent document

Précisé dans le Résumé et l'Introduction que le statut du présent document est Norme de l'Internet.

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

Ajout d'une référence à la RFC 6989 quand on réutilise des exponentielles Diffie-Hellman (paragraphe 2.12).

Ajout du nom "Dernière sous structure" pour l'en-tête Sous structure Proposition et Sous structure Transformation (paragraphes 3.3.1 et 3.3.2) pour le champ 0 (dernière) ou 2/3 (plus).

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

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

Les clés publique brutes RSA sont déconseillées au paragraphe 3.6. Un nouveau travail en cours 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 RFC 5996 (identifiants 2707 et 3036).

Ajout d'une note dans les Considérations relatives à l'IANA (Section 6) sur les clés brutes RSA déconseillées, et suppression de l'ancien contenu (ce qui a déjà été fait durant le traitement de la RFC 5996). Ajout d'une note que l'IANA devraient mettre à jour toutes les références à la RFC 5996 pour pointer sur le présent document.

2. Détails et variantes du protocole IKE

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 la perte de paquet, la répétition de paquet, et la falsification de paquet. IKE est conçu pour fonctionner tant que (1) au moins une des séries 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 au point d'épuiser les capacités du réseau ou de CPU de l'un ou l'autre des points d'extrémité. Même en l'absence de ces exigences miniméles de performances, IKE est conçu pour un échec propre (même si le réseau est cassé).

Bien que les messages IKEv2 soient destinés à être courts, ils contiennent des structures sans limite ferme de taille supérieure (en particulier, les certificats numériques) et IKEv2 lui-même n'a pas de mécanisme pour fragmenter les grands messages. IP définit un mécanisme de fragmentation des messages UDP de grande taille, mais les mises en œuvre varient sur la taille maximum de message prise en charge. De plus, l'utilisation de la fragmentation IP ouvre les mises en œuvre à des attaques de déni de service (DoS) [DOSUDPPROT]. Finalement, certaines mises en œuvre de NAT et/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 des 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 PEUT raccourcir les messages en laissant de côté certains certificats ou propositions de suite de chiffrement si cela garde les messages en dessous du maximum. L'utilisation de formats "Hachage et URL" plutôt que d'inclure des certificats dans les échanges lorsque possible peut éviter la plupart des problèmes. Les mises en œuvre et la configuration doivent cependant garder en mémoire que si des recherches d'URL ne sont possibles qu'après que la SA fille est établie, 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 va pas savoir comment les traiter.

2.1 Utilisation de temporisateurs de retransmission

Tous les messages dans IKE existent par paires : 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 moment. 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 celui qui répond.

Pour chaque paire de messages IKE, l'initiateur est responsable de la retransmission en cas d'une fin de temporisation. Celui qui répond NE DOIT jamais retransmettre une réponse si il ne reçoit pas une retransmission de la demande. Dans ce cas, celui qui répond DOIT ignorer la demande retransmise sauf si elle cause une retransmission de la réponse. L'initiateur DOIT se souvenir de chaque demande jusqu'à ce qu'il reçoive la réponse correspondante. Celui qui répond 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 de la réponse plus sa taille de fenêtre (voir le paragraphe 2.3). Afin de permettre d'économiser la mémoire, il est permis à celui qui répond d'oublier la réponse après une temporisation de plusieurs minutes. Si celui qui répond 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 qu'il reçoive une réponse correspondante ou estime que la SA IKE est défaillante. Dans ce dernier cas, l'initiateur élimine tout l'état associé à la SA IKE et à toutes les SA filles qui ont été négociées en utilisant cette SA IKE. Une retransmission de l'initiateur DOIT être identique au bit près à la demande originale. C'est-à-dire, tout ce qui commence depuis 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 spécial. Quand celui qui répond 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 celui qui répond retransmet la même réponse) ou une nouvelle demande (dans ce cas celui qui répond crée une nouvelle SA IKE et envoie une réponse fraîche) ou si il appartient à une SA IKE existante où la demande

IKE_AUTH a déjà été reçue (dans ce cas celui qui répond 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 pourrait choisir le même SPI d'initiateur. À la place, un répondant robuste va faire une recherche de SA IKE en utilisant le paquet complet, son hachage, ou la charge utile Ni.

La politique de retransmission pour les messages unidirectionnels est un peu différente de celle des messages réguliers. Comme aucun accusé de réception n'est envoyé, il n'y a pas de raison de retransmettre gratuitement des messages unidirectionnels. Étant donné que tous ces messages sont des erreurs, il y a du sens à ne les envoyer que seulement une fois par paquet en cause, et ne les retransmettre que si d'autres paquets en défaut sont reçus. Cependant, il y a aussi du sens à limiter les retransmissions de tels messages d'erreur.

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

Chaque message IKE contient un identifiant de message au titre de son en-tête fixe. Cet identifiant de message est utilisé pour faire correspondre 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 original.

L'identifiant de message est une quantité de 32 bits, qui est zéro pour les messages IKE_SA_INIT (y compris les retransmissions du message dues à des réponses telles que COOKIE et INVALID_KE_PAYLOAD) et incrémentée pour chaque échange suivant. Donc, la première paire de messages IKE_AUTH va avoir un identifiant de 1, la seconde (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 le changement de clé de la SA IKE.

Chaque point d'extrémité dans l'association de sécurité IKE maintient deux identifiants de message "courants" : le prochain à utiliser pour une demande qu'il initie et le prochain qu'il s'attend à voir dans une demande provenant de l'autre extrémité. Ces compteurs s'incrémentent lorsque 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 nième demande provenant le l'initateur IKE original, la réponse correspondante, la nième demande provenant du répondant IKE original, et la réponse correspondante. Si les deux extrémité font un nombre très différent de demandes, les identifiants de message dans les deux directions peuvent être très différents. Il n'y a cependant pas d'ambiguïté dans les messages, parce que les fanions Initiateur et Réponse dans l'en-tête de message spécifient lequel des quatre messages est un message particulier.

Dans le présent document, "initiateur" se réfère à la partie qui a initié l'échange décrit. L'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 l'initiateur "original" de la nouvelle SA IKE.

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

2.3 Taille de fenêtre pour les demandes en chevauchement

La notification SET_WINDOW_SIZE affirme que le point d'extrémité envoyeur est capable de garder l'état pour plusieurs échanges en cours, permettant au receveur d'envoyer plusieurs demandes avant d'obtenir la réponse à la première. Les données associées à la notification SET_WINDOW_SIZE DOIVENT être longues de 4 octets et contenir la représentation en ordre gros boutien du nombre de messages que l'envoyeur promet de garder. La taille de fenêtre est toujours un jusqu'à la fin des échanges initiaux.

Un point d'extrémité IKE DOIT attendre la réponse de 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 garder l'état pour plusieurs messages en instance afin de permettre un meilleur 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 les unes les autres dans 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 de l'homologue pour les demandes IKE transmises. En d'autres termes, si celui qui répond a déclaré que sa taille de fenêtre est N, alors quand l'initiateur a besoin de faire une demande X, il DOIT attendre qu'il ait reçu les réponses à toutes les demandes jusqu'à la demande X-N. Un point d'extrémité IKE DOIT garder une copie de (ou être capable de régénérer exactement) chaque demande qu'il a envoyé jusqu'à ce qu'il reçoive la réponse correspondante. Un point d'extrémité IKE DOIT garder une copie du (ou être capable de régénérer exactement) nombre de réponses précédentes égal à sa taille de fenêtre déclarée au cas où sa réponse serait perdue et où l'initiateur demanderait 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 pour maximiser les performances en cas de défaillances du réseau ou de réarrangement de paquets.

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

La notification INVALID_MESSAGE_ID est envoyée quand est reçu un identifiant de message IKE 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 être acquittée. À la place, on informe l'autre côté en initiant un échange INFORMATIONAL avec les données de notification contenant les quatre octets de l'identifiant de message invalide. L'envoi de cette notification est FACULTATIF, et le taux des notifications de ce type DOIT être limité.

2.4 Synchronisation d'état et fins de temporisation de connexion

Il est permis à un point d'extrémité IKE d'oublier tout son état associé à une SA IKE et la collection des SA filles correspondante à tout moment. C'est le comportement prévu dans le cas d'une panne d'un point d'extrémité et son redémarrage. 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 de 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 affirme 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 SA qui ont 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, les accréditifs d'un utilisateur en itinérance où il est permis à l'utilisateur de se connecter au pare-feu de l'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 qui reçoivent PEUVENT l'ignorer dans les autres messages.

Comme IKE est conçu pour fonctionner en dépit des attaques de DoS provenant du réseau, un point d'extrémité NE DOIT PAS en conclure que l'autre point d'extrémité a eu 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 se plaignant de SPI inconnus). 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 protégée cryptographiquement est reçue sur une SA IKE différente pour la même identité authentifiée. Un point d'extrémité devraient soupçonner que l'autre point d'extrémité est défaillant sur la base des 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 Chiffrée qui ne contient pas de charge utile). Si un message protégé cryptographiquement (frais, c'est-à-dire, non retransmis) a été reçu récemment de l'autre côté, les messages Notify non protégés PEUVENT être ignorés. Les mises en œuvre DOIVENT limiter le taux auquel elles prennent des actions sur la base de messages non protégés.

Le nombre d'essais et la longueur des temporisations ne sont pas traités dans cette 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 temps de retransmission DOIVENT augmenter exponentiellement pour éviter d'inonder le réseau et rendre pire une situation d'encombrement existante. Si il y a seulement du trafic sortant sur toutes les SA associées à une SA IKE, il est essentiel de confirmer la vie de l'autre point d'extrémité pour éviter des trous noirs. Si aucun message protégé cryptographiquement n'a été reçu récemment sur une SA IKE ou une de ses SA filles, le système doit effectuer une vérification de vie afin d'empêcher l'envoi de messages à un homologue mort. (Ceci est parfois appelé "détection d'homologue mort" ou "DPD", bien qu'en réalité ce soit la détection d'homologue vivant, et non pas mort.) La réception d'un message frais protégé cryptographiquement sur une SA IKE ou une de ses SA filles assure la vie de la SA IKE et de toutes ses SA filles. Noter que cela fait peser des exigences sur les modes de défaillance d'un point d'extrémité IKE. Une mise en œuvre doit arrêter d'envoyer sur toute SA si une défaillance l'empêche de recevoir sur toutes les SA associées. Si un système crée des SA filles qui peuvent avoir une défaillance indépendamment les unes des autres sans que la SA IKE associée soit capable d'envoyer un message de suppression, alors le système DOIT 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 mesures appropriées : comme les deux premiers messages de l'établissement d'une SA ne sont pas protégés cryptographiquement, un attaquant pourrait répondre au message de l'initiateur avant le répondant authentique et empoisonner la tentative d'établissement de connexion. Pour empêcher cela, l'initiateur PEUT vouloir accepter plusieurs réponses à son premier message, traiter chaque réponse comme potentiellement légitime, répondre à chacune, et ensuite éliminer toutes les connexions invalides semi ouvertes 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 se mettre d'accord sur la durée de vie d'une SA. Si IKE suppose que le partenaire est mort, sur la base du manque répété d'accusé de réception à un message IKE, alors la SA IKE et toutes ses SA filles établis par cette SA IKE sont supprimées.

Un point d'extrémité IKE peut à tout moment supprimer les SA filles inactives pour récupérer les ressources utilisées pour tenir leur état. Si un point d'extrémité IKE choisit de supprimer ses SA filles, il DOIT envoyer des charges utiles Delete à l'autre extrémité en lui notifiant la suppression. Il PEUT de même périmer la SA IKE. Clôre la SA IKE clôt implicitement toutes les SA filles associées. Dans ce cas, un point d'extrémité IKE DEVRAIT envoyer une charge utile Delete indiquant qu'il a clôt la SA IKE sauf si l'autre point d'extrémité ne répond plus.

2.5 Numéros de version et compatibilité à l'avenir

Le présent document décrit la version 2.0 de IKE, ce qui signifie que le numéro de version majeur est 2 et le numéro de version mineur est 0. Le présent document remplace la [RFC4306]. Il est probable que certaines mises en œuvre vont vouloir prendre en charge la version 1.0 et version 2.0, et à l'avenir, d'autres versions.

Le numéro de version majeur devrait être incrémenté seulement si les formats de paquet ou les actions requises ont changé de façon si importante qu'un nœud d'une version plus ancienne ne serait plus capable d'interopérer avec un nœud d'une version plus récente si il ignorait simplement les champs qu'il ne comprend pas et prendre les actions spécifiées dans la spécification plus ancienne. Le numéro de version mineur indique de nouvelles capacités, et DOIT être ignoré par un nœud avec un numéro de version mineur plus petit, mais être utilisé pour des besoins d'information par le nœud avec le plus grand numéro de version mineur. Par exemple, il pourrait indiquer la capacité de traiter un nouveau type de message Notify défini. Le nœud avec le plus grand numéro de version mineur va simplement noter que son correspondant ne va pas être capable de comprendre ce message et donc ne va pas l'envoyer.

Si un point d'extrémité reçoit un message avec un numéro de version majeur plus élevé, il DOIT éliminer le message et DEVRAIT envoyer un message Notify non authentifié de type INVALID_MAJOR_VERSION contenant le plus fort (plus proche) numéro de version qu'il prend en charge. Si un point d'extrémité prend en charge la version majeure n et la version majeure m, il DOIT prendre en charge toutes les versions entre n et m. Si il reçoit un message avec une version majeure qu'il prend en charge, il DOIT répondre avec ce numéro de version. Afin d'empêcher que deux nœuds soient conduits par fraude à correspondre avec un numéro de version majeure inférieur au maximum qu'ils prennent tous deux en charge, IKE a un fanion qui indique que le nœud est capable de parler un numéro de version majeure supérieur.

Donc, le numéro de version majeure 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 parler les versions n, n+1, et n+2, et si celui qui répond est capable de parler les versions n et n+1, alors ils vont négocier de parler n+1, et l'initiateur va établir un fanion indiquant sa capacité de parler une version supérieure. Si ils négocient par erreur (peut-être parce qu'un attaquant actif envoie des messages d'erreur) la version n, alors 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 pas de moyen en v1 de noter qu'on est capable de parler un numéro de version supérieur. Ainsi un attaquant actif peut conduire deux nœuds capables de v2 à parler en v1. Quand un nœud capable de v2 négocie une v1, il devrait noter ce fait dans son journal.

Aussi, pour la compatibilité future, tous les champs marqués Réservé DOIVENT être réglés à zéro par une mise en œuvre de version 2.0, et leur contenu DOIT être ignoré par une mise en œuvre de version 2.0 ("être conservateur dans ce qu'on envoie et libéral dans ce qu'on reçoit" [RFC0791]). De cette façon, les futures versions du protocole pourront utiliser ces champs d'une façon qui garantisse qu'ils seront ignorés par les mises en œuvre qui ne les comprennent pas. De même, les types de charge utile qui ne sont pas définis sont réservés pour une utilisation future ; les mises en œuvre d'une version où ils sont indéfinis DOIVENT sauter ces charges utiles et ignorer leur contenu.

IKEv2 ajoute un fanion "critique" à chaque en-tête de charge utile pour plus de souplesse dans 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 sur 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, et pas à son contenu. Si le type de charge utile est reconnu, mais la charge utile contient quelque chose qui ne l'est pas (comme une transformation inconnue dans une charge utile SA, ou un type de message Notify inconnu dans une 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 cette spécification, les mises en œuvre DEVRAIENT envoyer les charges utiles définies dans la présente spécification dans l'ordre montré dans 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 les choisir de façon à être les identifiants uniques d'une SA IKE. Une valeur de SPI de zéro est spéciale : elle indique que la valeur du SPI distant n'est pas encore connue par l'envoyeur.

Les paquets IKE entrants sont transposés en une SA IKE seulement en utilisant 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 seul le SPI du receveur apparaît dans l'en-tête d'un message, dans IKE le SPI de l'envoyeur est aussi envoyé dans chaque message. Comme le SPI choisi par l'initiateur original de la SA IKE est toujours envoyé en premier, un point d'extrémité avec plusieurs SA IKE ouvertes qui veut trouver la SA IKE appropriée en utilisant le SPI qu'il a alloué doit regarder le fanion Initiateur dans l'en-tête pour déterminer si il a alloué les premiers ou les seconds huit octets.

Dans le premier message d'un échange initial IKE, l'initiateur ne va pas connaître la valeur du SPI de celui qui répond et va donc régler ce champ à zéro. Quand l'échange IKE_SA_INIT ne résulte pas en la création d'une SA IKE du fait d'un INVALID_KE_PAYLOAD, NON_PROPOSAL_CHOSEN, ou COOKIE, le SPI de celui qui répond va être zéro aussi dans le message de réponse. Cependant, si celui qui répond envoie un SPI de réponse non zéro, l'initiateur ne devrait pas rejeter la réponse pour cette seule raison.

Deux attaques attendues contre IKE sont l'épuisement d'état et de CPU, où la cible est inondée de demandes d'initialisation de sessions à partir d'adresses IP falsifiées. L'efficacité de ces attaques peut être diminuée si celui qui répond utilise une CPU minimale et n'engage pas d'état à une SA avant de savoir que l'initiateur peut recevoir les paquets à l'adresse d'où il

prétend les envoyer.

Quand celui qui répond détecte un grand nombre de SA IKE semi ouvertes, il DEVRAIT répondre aux demandes IKE_SA_INIT avec une réponse contenant la notification COOKIE. Les données associées à cette notification DOIVENT faire entre 1 et 64 octets (inclus) et sa génération est décrite plus loin. Si la réponse IKE_SA_INIT inclut 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 ou de celui qui répond sauf pour communiquer le mouchard *(cookie)*. 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, et 'B' est le SPI alloué par celui qui répond.

Une mise en œuvre IKE peut générer son mouchard de répondant de telle façon qu'elle n'exige aucun état sauvegardé pour reconnaître son mouchard valide quand arrive le second message IKE_SA_INIT. Les algorithmes exacts et la syntaxe 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 les mouchards pour mettre en œuvre une protection limitée contre le déni de service.

Une bonne façon de le faire est de régler le mouchard de celui qui répond à être :

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

où <secret> est un secret généré au hasard connu seulement de celui qui répond et périodiquement changé et | indique l'enchaînement. <VersionIDofSecret> devrait être changé chaque fois que <secret> est régénéré. Le mouchard peut être recalculé quand le IKE_SA_INIT arrive pour la seconde fois et est comparé au mouchard dans le message reçu. Si il s correspondent, celui qui répond va savoir que le mouchard a été généré depuis le dernier changement de <secret> et que IPi doit être la même que l'adresse de source qu'il a vu la première fois. L'incorporation de 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 choisit des SPIi uniques). Incorporer Ni dans le hachage assure qu'un attaquant qui voit seulement le message 2 ne peut pas réussir à fabriquer un faux message 3. Aussi, incorporer SPIi dans le hachage empêche un attaquant d'aller chercher un mouchard de l'autre extrémité, et ensuite d'initier de nombreux échanges IKE_SA_INIT avec tous des SPI d'initiateur (et peut-être des numéros d'accès) différents afin que celui qui répond pense qu'il y a un certain nombre de machines derrière un NAT qui essayent toutes de se connecter.

Si une nouvelle valeur de <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. Celui qui répond PEUT dans ce cas 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'une ou l'autre. Celui qui répond ne devrait pas accepter indéfiniment les mouchards après que <secret> est changé, car cela détruirait en partie la protection contre le DoS. Celui qui répond devrait changer la valeur de <secret> fréquemment, en particulier si il est soumis à une attaque.

Quand une partie reçoit une demande IKE_SA_INIT contenant un mouchard dont le contenu ne correspond pas 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 devraient limiter le nombre d'échanges de mouchards qu'il essaye avant d'abandonner, éventuellement en utilisant un retard exponentiel. Un attaquant peut forger de nombreuses réponses de mouchard au message IKE_SA_INIT de l'initiateur, et chaque réponse de ces mouchards falsifiés va causer l'envoi de deux paquets : un paquet de l'initiateur à celui qui répond (qui va rejeter ces mouchards) et une réponse du répondant à l'initiateur qui inclut le mouchard correct.

Note sur la terminologie : le terme de "cookie" *(mouchard en français)* a son origine dans Photuris de Karn et Simpson [RFC2522], proposition ancienne de gestion de clés avec IPsec, et il a persisté. L'en-tête de message fixe du protocole d'association de sécurité et de gestion de clé Internet (ISAKMP, *Internet Security Association and Key Management Protocol*) [RFC2408] inclut deux champs de huit octets appelés "mouchards", et cette syntaxe est utilisée par IKEv1 et IKEv2, bien que dans IKEv2 ils soient appelés "SPI IKE" et qu'il y ait un nouveau champ distinct dans une charge utile Notify pour contenir le mouchard.

2.6.1 Interaction de COOKIE et de INVALID_KE_PAYLOAD

Il y a deux raisons courantes pour que l'initiateur puisse devoir réessayer l'échange IKE_SA_INIT : celui qui répond demande un mouchard ou veut un groupe Diffie-Hellman différent de celui qui est inclus dans la charge utile KEi. Si l'initiateur reçoit un mouchard de celui qui répond, l'initiateur doit décider si il l'inclut ou non, seulement dans le prochain essai de la demande IKE_SA_INIT, ou aussi dans tous les essais suivants.

Si l'initiateur inclut le mouchard seulement 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 si celui qui répond ne le prend pas en charge. Par exemple, si celui qui répond 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 d'inclure le 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 DEVRAIENT prendre en charge cet échange plus court, mais NE DOIVENT PAS échouer si d'autres mises en œuvre ne le prennent pas en charge.

2.7 Négociation de l'algorithme de chiffrement

Le type de charge utile "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 des suites de chiffrement quand le nombre de suites prises en charge est grand parce que plusieurs valeurs sont acceptables pour de nombreuses transformations. Celui qui répond DOIT choisir une seule suite, qui peut être tout sous ensemble de la proposition de SA, en suivant les règles ci-dessous.

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

Chaque proposition de protocole IPsec contient une ou plusieurs transformations. Chaque transformation contient un type de transformation. La suite cryptographique acceptée DOIT contenir exactement une transformation de chaque type inclus dans la proposition. Par exemple: si une proposition ESP inclut des 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 et des chiffrement en mode combiné, alors deux propositions sont 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 permis avec un algorithme d'intégrité autre que "NONE").

2.8 Changement de clé

Les associations de sécurité IKE, ESP, et AH utilisent des clés secrètes qui devraient être utilisées seulement pour une durée limitée 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 il y a une demande, de nouvelles associations de sécurité PEUVENT être établies. Le rétablissement des associations de sécurité pour prendre la place de celles qui expirent est appelé un "changement de clé".

Pour permettre des mises en œuvre IPsec minimales, la capacité de changer la clé des SA sans redémarrer la SA IKE entière est facultative. Une mise en œuvre PEUT refuser toutes les demandes CREATE_CHILD_SA dans une SA IKE. Si une SA a expiré ou est sur le point d'expirer et que les tentatives de changement de clé en utilisant les mécanismes décrits ici échouent, une mise en œuvre DOIT clore la SA IKE et toutes les SA filles associées et PEUT ensuite en commencer de nouvelles. Les mises en œuvre peuvent souhaiter prendre en charge le changement de clé des SA en place, car le faire offre de meilleures performances et va probablement réduire le nombre de paquets perdus durant la transition.

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

Pour changer la clé d'une SA IKE, on établit une nouvelle SA IKE équivalente (voir le paragraphe 2.18) avec l'homologue avec qui l'ancienne SA IKE est partagée en utilisant un CREATE_CHILD_SA au sein de la SA IKE existante. Une SA IKE ainsi créée hérite de toutes les SA filles de la SA IKE originale, et la nouvelle SA IKE est utilisée pour tous les messages de contrôle nécessaires pour maintenir ces SA filles. Après la création de la nouvelle SA IKE équivalente, l'initiateur supprime l'ancienne SA IKE, et la charge utile Delete pour la supprimer DOIT être la dernière demande envoyée sur l'ancienne SA IKE.

Le changement de clé des SA devrait être fait de façon proactive, c'est-à-dire, la nouvelle SA devrait être établie avant que l'ancienne expire et devienne inutilisable. Assez de temps devrait s'écouler entre le moment où la nouvelle SA est établie et celui où l'ancienne devient inutilisable afin que le trafic puisse être commuté sur la nouvelle SA.

Une différence entre IKEv1 et IKEv2 est que dans la SA IKEv1 les durées de vie étaient négociées. Dans IKEv2, chaque extrémité de la SA est responsable d'appliquer sa propre politique de durée de vie sur la SA et de changer la clé de SA quand nécessaire. Si les deux extrémités ont des politiques de durée de vie différentes, l'extrémité avec la plus courte durée de cie va toujours être celle qui demande le changement de clé. Si une SA a été inactive pendant longtemps et si un point d'extrémité ne va pas initier la SA en l'absence de trafic, le point d'extrémité PEUT choisir de clore la SA au lieu de changer sa clé quand sa durée de vie expire. Il peut aussi le faire si il n'y a pas eu de trafic depuis la dernière fois que la clé de la SA a été changée.

Noter que IKEv2 permet délibérément des SA parallèles avec les mêmes sélecteurs de trafic entre des points d'extrémité communs. Une des raisons de cela est de prendre en charge les différences de qualité de service (QS) du trafic parmi 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 ne peut pas identifier de façon univoque une SA entre deux points d'extrémité, de sorte que l'heuristique de changement de clé de IKEv1 de supprimer les SA sur la base de sélecteurs de trafic dupliqués NE DEVRAIT PAS être utilisée.

Il y a des fenêtre de temporisation – en particulier en présence de pertes de paquets – où les points d'extrémité ne peuvent pas s'accorder sur l'état d'une SA. Celui qui répond 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, afin qu'il n'y ait pas d'ambiguïté pour l'initiateur. L'initiateur PEUT commencer à envoyer sur une SA aussitôt qu'il a traité la réponse. L'initiateur ne peut cependant pas recevoir sur une SA nouvellement créée tant qu'il n'a pas reçu et traité la réponse à sa demande CREATE_CHILD_SA. Comment dès lors celui qui répond va t-il savoir quand il peut envoyer sur la SA nouvellement créée ?

Du point de vue de la correction technique et de l'interopérabilité, celui qui répond PEUT commencer d'envoyer sur une SA aussitôt qu'il envoie sa réponse à la demande CREATE_CHILD_SA. Dans certaines situations, cependant, il pourrait

en résulter que les paquets soient éliminés sans nécessité, donc une mise en œuvre PEUT différer un tel envoi.

Celui qui répond peut être assuré que l'initiateur est prêt à recevoir les 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 la clé d'une SA existante et il reçoit une demande IKE de clore la SA remplacée. Quand il change la clé d'une SA, celui qui répond continue d'envoyer du trafic sur la vieille SA jusqu'à ce qu'un des événements suivants se produise. Quand il établit une nouvelle SA, celui qui répond PEUT différer d'envoyer des messages sur une nouvelle SA jusqu'à ce que il en reçoive un ou qu'une fin de temporisation se produise. Si un initiateur reçoit un message sur une SA pour laquelle il n'a pas reçu de réponse à sa demande CREATE_CHILD_SA, il l'interprète comme une probable perte 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 message en file d'attente afin d'assurer à celui qui répond que l'initiateur est prêt à recevoir des messages.

2.8.1 Changement simultané de clés de SA filles

Si les deux extrémités ont les mêmes politiques de durée de vie, il est possible que toutes deux initient un changement de clé en même temps (d'où vont résulter des SA redondantes). Pour réduire la probabilité que cela arrive, le rythme des demandes de changement de clé DEVRAIT être rendu aléatoire d'une durée variable après que le besoin de changement de clé est remarqué).

Cette forme de changement de clé peut résulter temporairement en plusieurs SA similaires entre les mêmes paires de nœuds. Quand il y a deux SA éligibles à recevoir les paquets, un nœud DOIT accepter les paquets entrants à travers l'une ou l'autre SA. Si des SA redondantes sont créées par une telle collision, la SA créée avec le plus faible des quatre noms occasionnels utilisés dans les deux échanges DEVRAIT être close par le point d'extrémité qui l'a créée. "Plus faible 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, passer au prochain octet, et ainsi de suite. Si on atteint la fin du nom occasionnel, ce nom occasionnel est le plus faible. Le nœud qui a initié la SA aux clés changées survivante devrait supprimer la SA remplacée après que la nouvelle est établie.

Une explication de l'impact que cela a sur les mises en œuvre suit. Supposons que les hôtes A et B aient une paire de SA filles avec les SPI (SPIa1,SPIb1), et que toutes deux commencent le changement de clé au même moment :

Hôte A **Hôte B**

---

envoie req1: N(REKEY_SA,SPIa1), SA(..,SPIa2,..),Ni1,.. -->
<-- envoie 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 faible nom occasionnel, donc il va juste noter la situation et répondre comme d'habitude.

envoie resp2: SA(..,SPIa3,..), Nr1,.. -->
--> reçoit req1

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

<-- envoie resp1 : SA(..,SPIb3,..), Nr2,..
reçoit resp1 <--
--> reçoit resp2

À ce point, il y a trois paires de SA filles entre A et B (l'ancienne et deux nouvelles). A et B peuvent maintenant comparer les noms occasionnels. Supposons que le plus faible nom occasionnel soit Nr1 dans le message resp2 ; dans ce cas, B (l'envoyeur de req2) supprime la nouvelle SA redondante, et A (le nœud qui a initié la SA survivante dont les clés ont été changées) supprime l'ancienne.

envoie req3: D(SPIa1) -->
<-- envoie req4 : D(SPIb2)
-> reçoit req3
<-- envoie resp3 : D(SPIb1)
reçoit req4 <--

envoie resp4 : D(SPIa3) -->

Le changement de clés est maintenant terminé.

Cependant, une seconde séquence d'événements possible peut se produire si des paquets sont perdus dans le réseau, résultant en des retransmissions. Le changement de clés se fait comme normalement, mais le premier paquet de A (req1) est perdu.

Hôte A **Hôte B**

---

envoie req1: N(REKEY_SA,SPIa1), SA(..,SPIa2,..), Ni1,.. --> (perdu)
<-- envoie req2 : N(REKEY_SA,SPIb1), SA(..,SPIb2,..),Ni2
reçoit req2 <--
envoie resp2: SA(..,SPIa3,..), Nr1,.. -->
--> reçoit resp2
<-- envoie req3 : D(SPIb1)
reçoit req3 <--
envoie resp3 : D(SPIa1) -->
--> reçoit resp3

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

renvoie req1 -->
--> reçoit req1

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

<-- envoie resp1 : N(CHILD_SA_NOT_FOUND)
reçoit resp1 <--

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

2.8.2 Changement simultané de SA IKE

Le cas probablement le plus complexe se produit quand les deux homologues essayent de changer la clé de la IKE_SA en même temps. 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 SA IKE correcte.

Le cas où les deux points d'extrémité remarquent le changement simultané de clé fonctionne de la même façon qu'avec les SA filles. Après les échanges CREATE_CHILD_SA, trois SA IKE existent entre A et B : l'ancienne SA IKE et deux nouvelles SA IKE. La nouvelle SA IKE contenant le plus faible 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 clé simultané, il y a un cas particulier où un homologue finit son changement de clé avant qu'il remarque que l'autre homologue fait un changement de clé. Si seulement un homologue a détecté un changement de clé simultané, les SA redondantes ne sont pas créées. Dans ce cas, quand l'homologue qui n'a pas remarqué le changement simultané de clé reçoit la demande de changement de clé de la SA IKE dont il a déjà réussi à changer les clés, il DEVRAIT retourner un TEMPORARY_FAILURE parce qu'il y a une SA IKE qu'il essaye actuellement de fermer (qu'il ait ou non déjà envoyé la notification de suppression pour la SA). Si l'homologue qui n'a pas remarqué le changement simultané de clé reçoit la demande de suppression provenant de l'autre homologue pour l'ancienne SA IKE, il sait que l'autre homologue n'a pas détecté le changement de clé simultané, et le premier homologue peut oublier sa propre tentative de changement de clé.

```
Hôte A                                          Hôte B
-------------------------------------------------------------------------
envoie req1 : SA(..,SPIa1,..),Ni1,.. -->
                                        <-- envoie req2 : SA(..,SPIb1,..),Ni2,..
                                        --> reçoit req1
                                        <-- envoie resp1 : SA(..,SPIb2,..),Nr2,..
reçoit resp1 <--
envoie req3 : D() -->
                                        --> reçoit req3
```

À ce point, l'hôte B voit une demande de clore la IKE_SA. Il n'y a pas grand chose de pls à faire que de répondre comme d'habitude. Cependant, à ce point, l'hôte B devrait arrêter de retransmetre req2, car une fois que l'hôte A a reçu resp3, il va supprimer tout l'état associé à l'ancienne IKE_SA et ne va pas être capable d'y répondre.

```
                                        <-- envoie resp3 : ()
```

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 anciens homologues qui mettent en œuvre la RFC 4306 mais pas le présent document peuvent recevoir ces notifications. Dans ce cas, ils vont la traiter de la même façon que toute autre notification d'erreur inconnue, et vont arrêter l'échange. Parce que l'autre homologue a déjà changé la clé de l'échange, faire comme cela n'a pas d'effet néfaste.

2.8.3 Changement de clé de SA IKE ou ré-authentification

Le changement de clé de SA IKE et la ré-authentification sont des concepts différents dans IKEv2. Le changement de clé de SA IKE établit de nouvelles clés pour la SA IKE et réinitialise 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 le changement de clé de SA IKE puisse être important dans certains environnements, la ré-authentification (la vérification que les parties ont toujours accès aux accréditifs à long terme) est souvent plus importante.

IKEv2 n'a pas de prise en charge particulière pour 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) en créant de nouvelles SA filles au sein de la nouvelle SA IKE (sans charge utile Notify REKEY_SA) et en supprimant finalement l'ancienne SA IKE (ce qui supprime aussi les anciennes 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 clé peut être effectué plus souvent que la ré-authentification, la situation où la "durée de vie de l'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 charges utiles EAP et/ou 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 à celui qui répond de demander la ré-authentification dans ce cas ; cependant, il y a des extensions qui ajoutent cette fonctionnalité comme la [RFC4478].

2.9 Négociation de sélecteur 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, il appartient à 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 pourraient 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, *Traffic Selector*) permettent aux points d'extrémité de communiquer certaines des informations de leur SPD à leurs homologues. Elles doivent être communiquées à IKE à partir de la SPD (par exemple, l'API PF_KEY [RFC2367] utilise le message SADB_ACQUIRE). Les charges utiles TS spécifient les critères de choix des 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 cohérentes. 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 (ou l’adresse de destination du trafic transmis à) du trafic transmis depuis l'initiateur de la paire de SA filles. TSr spécifie l’adresse de destination (ou l’adresse de source du trafic transmis de) du trafic transmis à celui qui répond 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 provenant du sous réseau 198.51.100.* sur le côté de l'initiateur au sous réseau 192.0.2.* sur le côté de celui qui répond, l'initiateur va inclure un seul sélecteur de trafic dans chaque charge utile TS. TSi va spécifier la gamme d’adresses (198.51.100.0 - 198.51.100.255) et TSr va spécifier la gamme d’adresses (192.0.2.0 - 192.0.2.255). En supposant que la proposition soit acceptable pour celui qui répond, il renverrait des charges utiles TS identiques.

IKEv2 permet à celui qui répond de choisir un sous ensemble du trafic proposé par l'initiateur. Cela pourrait arriver quand les configurations des deux points d'extrémité sont mises à jour mais que seulement une extrémité a reçu les nouvelles informations. Comme les deux points d'extrémité peuvent être configurés par des personnes différentes, l’incompatibilité peut persister pendant assez longtemps même en l’absence d’erreur. Cela permet aussi des configurations intentionnellement différentes, comme quand une extrémité est configurée à tunneler toutes les adresses et dépend de l’autre extrémité pour en avoir la liste à jour.

Quand celui qui répond 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, celui qui répond ignore ce sélecteur de trafic, de sorte que le type inconnu n’est pas retourné dans l’ensemble réduit.

Pour permettre à celui qui répond de choisir la gamme appropriée dans ce cas, si l'initiateur a demandé la SA à cause d’un paquet de données, l'initiateur DEVRAIT inclure comme premier sélecteur de trafic dans chaque TSi et TSr un sélecteur de trafic très spécifique incluant les adresses dans le paquet déclenchant 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) et l’accès de source et 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 similairement deux sélecteurs de trafic dans TSr. Si l'initiateur crée la paire de SA filles non en réponse à un paquet arrivant, mais plutôt, disons, au démarrage, il peut alors n’y avoir pas d’adresse spécifique que l'initiateur préfère à d’autres pour le tunnel initial. Dans ce cas, les premières valeurs dans TSi et TSr peuvent être des gammes plutôt que des valeurs spécifiques.

Celui qui répond effectue le rétrécissement comme suit :

- Si la politique de celui qui répond ne lui permet pas d’accepter une part quelconque des sélecteurs de trafic proposés, il répond par un message Notify TS_UNACCEPTABLE.

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

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

- Si la politique de celui qui répond ne lui permet pas d’accepter le premier sélecteur de TSi et TSr, celui qui répond le réduit à un sous ensemble acceptable de TSi et TSr.

Quand le rétrécissement est fait, il peut y avoir plusieurs sous ensembles acceptables mais dont l’union ne l’est pas. Dans ce cas, celui qui répond en choisit arbitrairement un, et PEUT inclure une notification ADDITIONAL_TS_POSSIBLE dans la réponse. La notification ADDITIONAL_TS_POSSIBLE affirme que celui qui répond a réduit les sélecteurs de trafic proposés mais que d’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 va se produire seulement quand l'initiateur et celui qui répond sont configurés différemment l’un de l’autre. Si l'initiateur et celui qui répond s’accordent sur la granularité des tunnels, l'initiateur ne va jamais demander un tunnel plus large que ce que celui qui répond va accepter.

Il est possible que la politique de celui qui répond contienne plusieurs gammes plus petites, toutes englobées par le sélecteur de trafic de l'initiateur, et la politique de celui qui répond étant que chacune de ces gammes devrait être envoyée sur une SA différente. En continuant l'exemple ci-dessus, celui qui répond pourrait avoir une politique voulant tunneler ces adresses de et vers l'initiateur, mais pourrait exiger que chaque paire d'adresses 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 aurait pas les très spécifiques premiers sélecteurs de trafic pour aider celui qui répond à choisir la gamme correcte. Il n'y aurait pas de moyen pour celui qui répond de déterminer quelle paire d'adresses devrait être incluse dans ce tunnel, et il devrait deviner ou rejeter la demande avec un message Notify SINGLE_PAIR_REQUIRED.

L'erreur SINGLE_PAIR_REQUIRED indique qu'une demande CREATE_CHILD_SA est inacceptable parce que son envoyeur veut seulement accepter des 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'adresses. À cause de cela, si seulement certaines parties des TSi et TSr proposées par l'initiateur sont acceptables à celui qui répond, 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 respectée, du trafic valide peut être éliminé. Si on utilise des politiques décorrélées de la [RFC4301], ces sortes de violations de politique ne peuvent 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 chiffré en utilisant AES via l'hôte B, 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 toute combinaison de AES et 3DES.

Si l'hôte A propose maintenant une SA qui utilise 3DES, et inclut un TSr contenant (198.51.100.0 – 198.51.100.255) cela va être 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 respecté sa propre politique, et plutôt inclure un TSr contenant ((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 X (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 voudrait accepter le trafic X' avec une SA' (! =SA), du trafic valide peut être éliminé sans nécessité parce que celui qui répond peut appliquer SA ou SA' au trafic X'.

2.9.2 Sélecteurs de trafic dans un changement de clé

Le changement de clé est utilisé pour remplacer une SA fille existante par une autre. Si il serait permis à la nouvelle SA d'avoir un ensemble plus étroit de sélecteurs qu'à celle d'origine, 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 SA aux clés changées a besoin d'avoir une portée plus étroite que la SA actuellement utilisée, cela signifierait que la politique a changé d'une façon telle que la SA actuellement utilisée est contraire à la politique. Dans ce cas, la SA devrait avoir été déjà supprimée après l'entrée en vigueur du changement de politique.

Quand l'initiateur tente de changer la clé de la SA fille, les sélecteurs de trafic proposés DEVRAIENT être soit les mêmes, soit un sur ensemble des sélecteurs de trafic utilisés dans l'ancienne SA fille. C'est-à-dire, ils seraient les mêmes, ou un sur-ensemble de la politique actuellement active (décorrélée). Celui qui répond NE DOIT PAS rétrécir les sélecteurs de trafic plus que la portée actuellement utilisée.

Parce que la SA dont la clé est changée ne peut jamais avoir une portée plus étroite que celle actuellement utilisée, il n'y a pas besoin de sélecteurs dans le paquet, de sorte que ces sélecteurs NE DEVRAIENT PAS être envoyés.

2.10 Noms occasionnels

Les messages IKE_SA_INIT contiennent chacun un nom occasionnel. 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 une SA fille, et pour assurer la création de bits pseudo aléatoires forts à partir de la clé Diffie-Hellman. Les noms occasionnels utilisés dans IKEv2 DOIVENT être choisis au hasard, 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 nombres aléatoires 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 Souplesse d'adresse et d'accès

IKE fonctionne sur les accès UDP 500 et 4500, et établit implicitement les associations ESP et AH pour les mêmes adresses IP sur lesquelles il fonctionne. Les adresses et accès IP dans l'en-tête externe ne sont cependant pas eux-mêmes protégés cryptographiquement, et IKE est conçu pour fonctionner même à travers des boîtes de traduction d'adresse réseau (NAT, *Network Address Translation*). Une mise en œuvre DOIT accepter les demandes entrantes même si l'accès de source n'est pas 500 ou 4500, et DOIT répondre à l'adresse et accès d'où la demande a été reçue. Elle DOIT spécifier l'adresse et l'accès où la demande a été reçue comme adresse de source et accès dans la réponse. IKE fonctionne de façon identique sur IPv4 ou IPv6.

2.12 Réutilisation des exponentielles Diffie-Hellman

IKE génère le matériel de chiffrement en utilisant un échange Diffie-Hellman éphémère afin de gagner la propriété de "secret parfait de transmission". Cela signifie que une fois qu'une connexion est close et que ses clés correspondantes sont oubliées, même quelqu'un qui a enregistré toutes les données provenant de la connexion et obtient l'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 des clés de session.

Réaliser un secret parfait de transmission 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.

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

Si et quand utiliser les exponentielles Diffie-Hellman est une décision privée en ce sens que cela ne va pas affecter 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 une est réutilisée pour éviter la seconde moitié du calcul. Voir [REUSE] et la [RFC6989] pour une analyse de la sécurité de cette pratique et pour des considérations de sécurité supplémentaires quand on réutilise des clés Diffie-Hellman éphémères.

2.13 Génération du matériel de chiffrement

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

On suppose que chaque algorithme de chiffrement et algorithme 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 clé) 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 (HMAC, *Hashed Message Authentication Code*) la taille de clé fixe est la taille du résultat de la fonction de hachage sous-jacente.

On suppose que les PRF acceptent des clés de toute longueur, mais ont une taille de clé préférée. La taille de clé préférée DOIT être utilisée comme longueur de SK_d, SK_pi, et SK_pr (voir le paragraphe 2.14). Pour les PRF fondées sur la 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 du 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, la PRF est utilisée itérativement. Le terme "prf+" décrit une fonction qui sort un flux pseudo aléatoire fondé sur les résultats d'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 ait été sorti de prf+. Les clés sont tirées de la chaîne de résultat sans égard aux limites (par exemple, si les clés requises sont une clé de 256 bits de la norme de chiffrement évoluée (AES, *Advanced Encryption Standard*) et une clé HMAC de 160 bits, et si la fonction prf génère 160 bits, la clé AES va venir de T1 et du début de T2, tandis que la clé HMAC va venir du reste de T2 et du début 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 du secret partagé Diffie-Hellman établi durant cet échange. SKEYSEED est utilisé pour calculer les 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 clé 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 la longueur de clé préférée de la PRF négociée.

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 des bits générés de la prf+). g^ir est le secret partagé provenant de l'échange éphémère Diffie-Hellman. g^ir est représenté comme une chaîne d'octets en ordre gros boutien bourrée avec des zéros si nécessaire pour qu'elle ait la longueur du module. Ni et Nr sont les noms occasionnels, débarrassés de tout en-tête. Pour des raisons historiques de rétro compatibilité, il y a deux PRF qui font l'objet d'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 dans le calcul de SKEYSEED, mais tous les bits sont utilisés en entrée de 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 ils n'utilisent pas l'authentification extensible (voir le paragraphe 2.16) les homologues sont authentifiés en étant chacun signé (ou muni d'un MAC en utilisant un secret partagé bourré comme clé, comme décrit plus loin) avec un bloc de données. Dans ces calculs, IDi' et IDr' sont les charges utiles d'identifiant entières en excluant l'en-tête fixe. Pour celui qui répond, les octets à signer commencent par 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 en finissant avec le dernier octet de la dernière charge utile. Ajouté à cela (pour les besoins du calcul de la signature) sont le nom occasionnel Nr de celui qui répond, 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 par :

InitiateurSignésOctets = RéelMessage1 | NonceRDonnées | MACéIDPourI
GénéIKEEnTête = [quatre octets de 0 si on utilise l'accès 4500] | RéelIKEEnTête
RéelIKEEnTête = SPIi | SPIr | . . . | Longueur
RéelMessage1 = RéelIKEEnTête | ResteDuMessage1
NonceRCharUtile = EnTêteCharUtile | NonceRDdonnées
InitiateurIDCharUtile = EnTêteCharUtile | ResteDeInitIDCharUtile
ResteDeInitIDCharUtile = IDType | Réservé | InitIDDonnées
MACéIDPourI = prf(SK_pi, RestDeInitIDCharUtile)

Les octets signés de celui qui répond peuvent être décrits par :

RépondantSignésOctets = RéelMessage2 | NonceIDonnées | MACéIDPourR
GénéIKEEnTête = [quatre octets de 0 si on utilise l'accès 4500] | RéelIKEEnTête
RéelIKEEnTête = SPIi | SPIr | . . . | Longueur
RéelMessage2 = RéelIKEEnTête | ResteDuMessage2
NonceICharUtile = EnTêteCharUtile | NonceIDonnées
RépondantIDCharUtile = EnTêteCharUtile | ResteDeRespIDCharUtile
ResteDeRespIDCharUtile = IDType | Réservé | RespIDDonnées
MACéIDPourR = prf(SK_pr, ResteDeRespIDCharUtile)

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 du 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 dans la charge utile Identifiant. La signature ou le MAC va être calculé en utilisant les algorithmes dictés par le type de clé utilisée par le signataire, et spécifié par le champ Méthode d'authentification dans la charge utile Authentification. Il n'est pas exigé que l'initiateur et celui qui répond signent avec les mêmes algorithmes de chiffrement. Le choix des algorithmes de chiffrement dépend du type de clé de chacun. En particulier, l'initiateur peut utiliser une clé partagée alors que celui qui répond peut avoir une clé et un certificat de signature publique. Il va être souvent le cas (mais ce n'est pas exigé) que, si un secret partagé est utilisé pour l'authentification, la même clé soit utilisée dans les deux directions.

Noter qu'il est une 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 n'ont probablement pas une imprévisibilité suffisante pour résister aux attaques de dictionnaire et ces attaques ne sont pas empêchées dans cette méthode d'authentification. (Les applications qui n'utilisent pas d'authentification fondée sur le mot de passe pour l'amorçage et l'établissement de 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 doit contenir autant d'imprévisibilité que la plus forte clé négociée. Dans le cas d'une clé pré-partagée,la valeur de AUTH est calculée comme :

Pour l'initiateur : AUTH = prf(prf(Secret partagé, "bourrage de clé pour IKEv2"), <InitiateurSignésOctets>)

Pour celui qui répond : AUTH = prf(prf(Secret partagé, "bourrage de clé pour IKEv2"), <RépondantSignésOctets>)

où la chaîne "bourrage de clé pour IKEv2" est de 17 caractères ASCII sans terminaison nulle. Le secret partagé peut être de longueur variable. La chaîne bourrage est ajouté 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 pour IKEv2") qui pourrait n'être pas utilisée comme un mot de passe équivalent pour des protocoles autres que IKEv2. Comme noté ci-dessus, déduire le secret partagé d'un mot de passe n'est pas sûr. Cette construction est utilisée parce que il est prévu que les gens vont le faire de toutes façons. L'interface de gestion par laquelle est fourni le secret partagé DOIT accepter des 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 (décrite au paragraphe 2.16) et chaque type utilise des valeurs différentes dans les calculs de AUTH montrés ci-dessus. Si la méthode EAP est génératrice de clé, on substitue la clé de session maîtresse (MSK, *master session key*) pour le secret partagé dans le calcul. Pour les méthodes non génératrices de clé, on substitue SK_pi et SK_pr, respectivement, pour le secret partagé dans les deux calculs de AUTH.

2.16 Méthodes du protocole d'authentification extensible

En plus de l'authentification qui utilise les signatures de clé publique et les secrets partagés, IKE prend en charge l'authentification avec les méthodes définies dans la [RFC3748]. Normalement, ces méthodes sont asymétriques (conçues pour qu'un utilisateur s'authentifie à un serveur) et elles ne peuvent pas être mutuelles. Pour cette raison, ces protocoles sont normalement utilisés pour authentifier l'initiateur à celui qui répond et DOIVENT être utilisés en conjonction avec une authentification fondée sur une signature à clé publique de celui qui répond auprès de l'initiateur. Ces méthodes sont souvent associées à des mécanismes dits "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 cette spécification, on documente ici des variantes plus simples. 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 terminés afin d'initialiser la SA IKE.

Un initiateur indique son désir d'utiliser EAP en écartant la charge utile AUTH du 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 le reste 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 celui qui répond veut utiliser une méthode EAP, il va placer une charge utile EAP dans la réponse de l'échange IKE_AUTH et différer l'envoi de SAr2, TSi, et TSr jusqu'à ce que l'initiateur achève son authentification dans un échange IKE_AUTH suivant. Dans le cas d'une méthode EAP minimale, l'établissement initial de SA va apparaître comme suit :

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

Comme décrit au paragraphe 2.2, quand EAP est utilisé, chaque paire de messages d'établissement initial de SA IKE va avoir son numéro de message incrémenté ; la première paire de messages IKE_AUTH va avoir un identifiant de 1, la seconde va avoir 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 celui qui répond pour générer les charges utiles AUTH dans les messages 7 et 8 en utilisant la syntaxe des secrets partagés spécifiée au paragraphe 2.15. La clé partagée provenant de EAP est le champ de la spécification EAP nommé MSK. Cette clé partagée générée durant un échange IKE NE DOIT être utilisée pour aucun autre objet.

Les méthodes EAP qui n'établissent pas de clé partagée NE DEVRAIENT PAS être utilisées, car elles sont sujettes à de nombreuses attaques par interposition [EAPMITM] si ces méthodes EAP sont utilisées dans d'autres protocoles qui n'utilisent pas de tunnel authentifié par lc serveur. Voir les détails dans la section des considérations sur la sécurité. Si des méthodes EAP qui ne génèrent pas de 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 utilisant EAP doit être capable d'étendre l'échange initial de protocole à au moins dix échanges IKE_AUTH pour le cas où celui qui répond envoie des messages de notification et/ou réessaye l'invite d'authentification. Une fois l'échange de protocole défini par la méthode d'authentification EAP choisie terminé avec succès, celui qui répond DOIT envoyer une charge utile EAP contenant le message Succès. De même, si la méthode d'authentification a échoué, celui qui répond DOIT envoyer une charge utile EAP contenant le message Échec. Celui qui répond PEUT à tout moment terminer l'échange IKE en envoyant une charge utile EAP contenant le message Échec.

À la suite d'un 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 de l'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) et le choix de 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. Souvent le serveur EAP est mis en œuvre dans un serveur AAA séparé qui communique avec le répondant IKEv2. Dans ce cas, l'identité authentifiée, si elle diffère de celle de la charge utile IDi, doit être envoyée du serveur AAA au répondant IKEv2.

2.17 Génération du matériel de chiffrement pour les SA filles

Une seule SA fille est créée par l'échange IKE_AUTH, et des SA filles supplémentaires peuvent facultativement être créées dans des échanges 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 suivante.

Pour les échanges CREATE_CHILD_SA incluant un échange Diffie-Hellman facultatif, le matériel de chiffrement est défini comme :

KEYMAT = prf+(SK_d, 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 par une chaîne d'octets en ordre gros boutien bourré avec des zéros dans les bits de poids fort si nécessaire pour lui donner la longueur du module).

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

- Toutes les clés pour les SA portant des données de l'initiateur à celui qui répond sont prises avant celles des SA qui vont de celui qui répond 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 exige plusieurs clés, l'ordre dans lequel ces clés sont tirées du 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 en est) DOIT être prise des premiers bits et la clé d'intégrité (si il en est) DOIT être prise sur 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 SA (voir le paragraphe 2.13 pour la description des longueurs de clé, et le paragraphe 3.3.5 pour la définition de l'attribut Transformation de longueur de clé).

2.18 Changement de clé des SA IKE en utilisant l'échange CREATE_CHILD_SA

L'échange CREATE_CHILD_SA peut être utilisé pour changer la clé 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 dans les structures de proposition à l'intérieur des charges utiles Association de sécurité (pas des champs SPI dans l'en-tête IKE). Les charges utiles TS sont omises dans le changement de clé d'une SA IKE. Le SKEYSEED pour la nouvelle SA IKE est calculé en utilisant 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 éphémère Diffie-Hellman de cet échange CREATE_CHILD_SA (représenté comme une chaîne d'octets en ordre gros boutien bourrée avec des zéros si nécessaire pour qu'elle ait la longueur du module) et Ni et Nr sont les deux noms occasionnels débarrassés de tout en-tête.

L'ancienne et la nouvelle SA IKE peuvent avoir choisi une PRF différente. Parce que l'échange de changement de clé appartient à l'ancienne SA IKE, c'est la PRF de l'ancienne SA IKE qui est utilisée pour générer le SKEYSEED.

La principale raison du changement de clé de SA IKE est d'assurer que la compromission de l'ancien matériel de chiffrement ne donne pas d'information sur les clés actuelles, ou vice versa. Donc, les mises en œuvre DOIVENT effectuer un nouvel échange Diffie-Hellman quand elles changent de clé la SA IKE. En d'autres termes, un initiateur NE DOIT PAS proposer la valeur "NONE" pour la transformation Diffie-Hellman, et un répondant NE DOIT PAS accepter une telle proposition. Cela signifie que un échange réussi de changement de clé de SA IKE inclut toujours les charges utiles KEi/KEr.

La nouvelle SA IKE DOIT réinitialiser 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 PRF de la nouvelle SA IKE.

2.19 Demande d'une adresse interne sur un réseau distant

Se produisant le plus couramment dans le scénario de point d'extrémité à passerelle de sécurité, un point d'extrémité peut avoir besoin d'une adresse IP dans le réseau protégé par la passerelle de sécurité et peut avoir besoin que cette adresse soit allouée de façon dynamique. Une demande d'une telle adresse temporaire peut être incluse dans toute demande de créer une SA fille (incluant la demande implicite du message 3) en incluant une charge utile CP. Noter cependant qu'il est usuel d'allouer seulement une 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 les autres informations concernant le réseau protégé) dans l'échange IKE_AUTH. L'IRAS peut procurer une adresse pour l'IRAC à partir de tout nombre de sources comme 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 (IPv4 ou IPv6) mais PEUT contenir tout nombre d'attributs supplémentaires que l'initiateur veut retourner dans la réponse.

Par exemple, dans un message de l'initiateur à celui qui répond :

```
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 de celui qui répond 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, le 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.

Celui qui répond 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 échoue à envoyer une CP(CFG_REQUEST), l'IRAS DOIT faire échouer la demande, et terminer la création de SA fille avec une erreur FAILED_CP_REQUIRED. FAILED_CP_REQUIRED n'est pas fatal pour la SA IKE ; elle cause simplement l'échec de la création de SA fille. L'initiateur peut corriger cela en commençant plus tard une nouvelle demande de charge utile Configuration. Il n'y a pas de données associées à 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 suivante. 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 donner les informations de version avant l'authentification ou même après l'authentification au cas où une mise en œuvre serait connue pour avoir des faiblesses de sécurité. Dans ce cas, elle DOIT soit retourner une chaîne vide, soit pas de charge utile CP si CP n'est pas pris en charge.

```
Initiateur                                                    Répondant
-------------------------------------------------------------------------------
HDR, SK {CP(CFG_REQUEST)}  -->
                                        <--  HDR, SK {CP(CFG_REPLY)}

CP(CFG_REQUEST) = APPLICATION_VERSION("")

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

2.21 Traitement des erreurs

De nombreuses sortes d'erreurs peuvent se produire durant le traitement IKE. La règle générale est que si une demande est reçue mal formatée, ou inacceptable pour des raisons de politique (comme pas d'algorithme de chiffrement correspondant) la réponse contient une charge utile Notify indiquant 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 du 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 la seule façon de renvoyer un message d'erreur). De tels paquets de réponse d'erreurs d'analyse ou de traitement devraient quand même amener le receveur à nettoyer l'état IKE (par exemple, en envoyant un Delete pour la mauvaise SA).

Seules les défaillances d'authentification (AUTHENTICATION_FAILED et défaillance EAP) et les messages mal formés (INVALID_SYNTAX) conduisent à la suppression de la SA IKE sans exiger un échange INFORMATIONAL explicite portant une charge utile Delete. D'autres conditions d'erreur PEUVENT requérir un tel échange si la politique l'impose. Si l'échange est terminé avec Échec EAP, une notification AUTHENTICATION_FAILED n'est pas envoyée.

2.21.1 Traitement d'erreur dans IKE_SA_INIT

Les erreurs qui se produisent avant qu'une SA IKE cryptographiquement protégée soit établie doivent être traitées très prudemment. 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 falsifiés.

Dans un échange IKE_SA_INIT, toute notification d'erreur cause l'échec de l'échange. Noter que certaines notifications d'erreur comme COOKIE, INVALID_KE_PAYLOAD ou INVALID_MAJOR_VERSION peuvent conduire à un échange suivant réussi. Parce que 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 une raison qiuelconque (secret partagé invalide, identifiant invalide, producteur de certificat pas de confiance, certificat révoqué ou expiré, etc.) DEVRAIT résulter en une notification AUTHENTICATION_FAILED. Si l'erreur s'est produite sur celui qui répond, la notification est retournée dans la réponse protégée, et est généralement 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 cette information 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 nouvel échange 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 lorsque tout le message est mal formé (plutôt que juste un mauvais contenu de charge utile) DOIVENT être rejetés en totalité, et DOIVENT seulement conduire à une notification UNSUPPORTED_CRITICAL_PAYLOAD ou INVALID_SYNTAX envoyée en réponse. Le receveur ne devrait pas vérifier les charges utiles relatives à l'authentification dans ce cas.

Si l'authentification a réussie dans l'échange IKE_AUTH, la SA IKE est établie ; cependant, établir la SA fille ou demander les 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, NON_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 plus tard une telle SA IKE.

Dans un échange IKE_AUTH, ou dans l'échange INFORMATIONAL qui le suit immédiatement (dans le cas où une erreur se produit 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 charge utile Delete. Des documents d'extension pourront définir de nouvelles notifications d'erreur avec cette

sémantique, mais NE DOIVENT PAS les utiliser sauf si l'homologue a montré qu'il les comprend, comme par l'utilisation de la charge utile Vendor ID.

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 notifiant l'erreur à l'autre extrémité.

Dans les situations ormales, 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, donc il ne devrait y avoir aucune raison pour qu'un homologue envoie des messages d'erreur à l'autre extrémité sauf une réponse. Parce que l'envoi de tels messages d'erreur comme un échange INFORMATIONAL pourrait conduire à d'autres erreurs qui pourraient causer des boucles, de telles erreurs NE DEVRAIENT PAS être envoyées. Si des erreurs sont vues qui indiquent que les homologues n'ont pas le même état, il serait bon de supprimer la SA IKE pour nettoyer l'état et recommencer.

Si un homologue en analysant une demande remarque qu'elle est mal formatée (après qu'elle a passé les vérifications de code d'authentification de message et les vérifications 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 nécessiter une charge utile Delete explicite.

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

Un nœud doit limiter le taux d'envoi des 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 démarrer une SA IKE) cela peut être le résultat d'une panne récente 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, la réponse DOIT être envoyée à l'adresse et accès IP d'où elle vient avec les mêmes SPI IKE et l'identifiant de message copié. La réponse NE DOIT PAS être protégée cryptographiquement et DOIT contenir une charge utile Notify INVALID_IKE_SPI. La notification INVALID_IKE_SPI indique qu'un message IKE a été reçu avec un SPI de destination non reconnu ; cela 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 falsification ou pourrait être une réponse qu'un correspondant authentique a été trompé pour la lui faire envoyer. Un nœud devrait traiter un tel message (et aussi un message du réseau comme un message ICMP destination injoignable) comme l'indication qu'il pourrait y avoir des problèmes avec les SA pour cette adresse IP et devrait initier une vérification de vie pour toute SA IKE de ce genre. Une mise en œuvre DEVRAIT limiter la fréquence de tels essais pour éviter d'être conduite à participer à une attaque de DoS.

Si une erreur se produit en dehors du contexte d'une demande IKE (par exemple, le nœud obtient 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 provenant d'une adresse IP (et accès, si une traversée de NAT est utilisée) avec laquelle il a une SA IKE DEVRAIT envoyer une charge utile IKE Notify dans un échange INFORMATIONAL IKE sur cette SA. Le receveur DOIT par suite de cela ne changer l'état d'aucune SA, mais peut souhaiter examiner l'événement pour aider à diagnostiquer des dysfonctionnements.

2.22 Compression IP

L'utilisation de la compression IP [RFC3173] peut être négociée au titre de l'établissement d'une SA fille. Alors 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 qui la contient. Les associations de compression disparaissent quand la SA ESP ou AH correspondante s'en va. Elle n'est pas explicitement mentionnée dans une charge utile Delete.

La négociation de la compression IP est séparée de la négociation des paramètres de chiffrement associée à une SA fille.

Un nœud qui demande une SA fille PEUT annoncer sa prise en charge d'un ou plusieurs algorithmes de compression par une ou plusieurs charges utiles Notify de type IPCOMP_SUPPORTED. Ce message Notify peut être inclus seulement dans un message contenant une charge utile SA négociant 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 de 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 que plusieurs algorithmes sont pris en charge. Un message acceptant une SA ne peut en contenir qu'une.

La liste des identifiants de transformation figure ci-dessous. Les valeurs du tableau suivant sont seulement les valeurs courantes 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. Le lecteurs devrait 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 en utilisant un algorithme autre que celui proposé et accepté à l'établissement de la SA fille.

Un effet collatéral de la séparation de la négociation de IPComp de celle des paramètres de chiffrement est qu'il n'est pas possible de proposer plusieurs suites cryptographiques et de proposer la compression IP avec certaines d'entre elles 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 pensent que les NAT sont le mal absolu et qu'on ne devrait pas concevoir de protocoles pour qu'elles fonctionnent mieux. IKEv2 spécifie bien des règles de traitement non intuitives pour que les NAT aient une meilleure probabilité de fonctionner.

Les NAT existent principalement à cause du manque d'adresses IPv4, bien qu'il y ait 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 certain espace qui est unique au sein du réseau derrière le NAT mais qui vont probablement être réutilisées par des nœuds derrière d'autres NAT. Généralement, des nœuds derrière des NAT peuvent communiquer avec les autres nœuds derrière le même NAT et avec des nœuds qui ont des adresses uniques au monde, mais pas avec des nœuds derrière d'autres NAT. Il y a des exceptions à cette règle. Quand ces nœuds font des connexions à des nœuds sur l'Internet réel, la passerelle de NAT "traduit" l'adresse IP de source en une adresse qui va être réacheminée à 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 incluent des adresses IP des points d'extrémité au sein de charges utiles de paquet vont échouer sauf si la passerelle de NAT comprend le protocole et modifie les références internes ainsi que celles des en-têtes. Une telle connaissance est non fiable par essence, est une violation de la couche réseau, et résulte souvent en problèmes subtils.

Ouvrir une connexion IPsec à travers un NAT introduit des problèmes particuliers. Si la connexion fonctionne en mode transport, changer les adresses IP sur les paquets va causer l'échec des sommes de contrôle et le NAT ne peut pas corriger les sommes de contrôle parce qu'elles sont cryptographiquement protégées. Même en mode tunnel, il y a des problèmes d'acheminement parce que traduire de façon transparente les adresses des paquets AH et ESP exige une logique particulière dans le NAT et cette logique est de 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 à passer le trafic IPsec encapsulé dans UDP mais pas en clair, en ESP/AH non encapsulé ou vice versa.

Il est une 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 obtenir un certain paquet. Pour cette raison, même si les paquets IKE DOIVENT être envoyés de et vers l'accès UDP 500 ou 4500, ils DOIVENT être acceptés venant de tout accès et les réponses DOIVENT être envoyées à l'accès d'où elles sont venues. C'est parce que les accès peuvent être modifiés lorsque les paquets passent à travers les NAT. De même, les adresses IP des points d'extrémité IKE ne sont généralement pas incluses dans les charges utiles IKE parce que les charges utiles sont cryptographiquement protégées et ne pourraient pas être modifiées de façon transparente par les NAT.

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

Un initiateur peut utiliser l'accès 4500 pour IKE et ESP, sans considérer si il y a ou non un NAT, même au début de IKE. Quand chaque côté utilise l'accès 4500, l'envoi de ESP avec encapsulation UDP n'est pas exigé, mais comprendre les paquets ESP reçus encapsulés dans ESP est exigé. 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 à tout moment être capables de recevoir et traiter les paquets ESP encapsulés dans UDP et non encapsulés dans UDP. L'un ou l'autre côté peut décider d'utiliser ou non l'encapsulation UDP pour ESP sans égard au choix fait par l'autre côté. Cependant, si un NAT est détecté, les deux appareils DOIVENT utiliser l'encapsulation UDP pour ESP.

Les exigences spécifique de la prise en charge de la traversée de NAT [RFC3715] sont énumérées ci-dessous. La prise en charge de la traversée de NAT est facultative. Dans ce paragraphe, seules les exigences mentionnées comme DOIT s'appliquent seulement aux mises en œuvre qui prennent en charge la traversée de NAT.

- L'initiateur et le répondant IKE DOIVENT inclure dans leurs paquets IKE_SA_INIT les 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 localisation des 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 à la notification NAT_DETECTION_SOURCE_IP sont un résumé SHA-1 des SPI (dans l'ordre dans lequel ils apparaissent dans l'en-tête) l'adresse IP, et l'accès d'où 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 de plusieurs 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 de leur apparition dans l'en-tête) l'adresse IP, et l'accès auquel ce paquet a été envoyé.

- Le receveur de la notification NAT_DETECTION_SOURCE_IP ou NAT_DETECTION_DESTINATION_IP PEUT comparer la valeur fournie à un hachage SHA-1 des SPI, à l'adresse IP de source ou de réception, et à l'accès (respectivement) 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 prise en charge. Dans le cas d'une discordance du hachage 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 d'envoyer des paquets de maintien en vie comme défini dans la [RFC3948] ; autrement, il PEUT rejeter la tentative de connexion si la traversée de NAT n'est pas supportée.

- Si aucune des charges utiles NAT_DETECTION_SOURCE_IP reçues ne correspond à la valeur attendue de la source

IP et l'accès 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, que quelqu'un sur le chemin a changé l'adresse de source du paquet original pour correspondre à l'adresse de la boîte 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 les charges utiles NAT_DETECTION_SOURCE_IP ou NAT_DETECTION_DESTINATION_IP si elles sont présentes, et si elles ne correspondent pas aux adresses dans le paquet externe, 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 devant et le résultat suit immédiatement l'en-tête UDP. Pour tunneler les paquets ESP sur l'accès UDP 4500, l'en-tête ESP suit immédiatement l'en-tête UDP. Comme les quatre premiers octets de l'en-tête ESP contiennent le SPI, et que le SPI ne peut pas validement être zéro, il est toujours possible de distinguer les messages ESP et IKE.

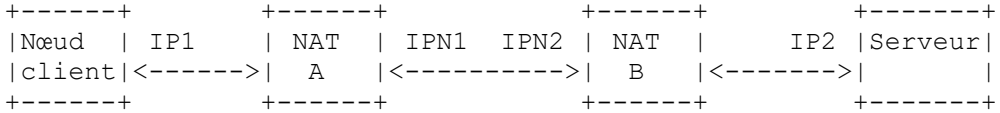
- Les mises en œuvre DOIVENT traiter les paquets ESP encapsulés dans UDP reçus même quand aucun NAT n'a été détecté.

- L'adresse IP originale de source et de destination requise pour la correction de la somme de contrôle de paquet TCP et UDP en mode transport (voir la [RFC3948]) est obtenue 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 traité plus en détails au paragraphe 2.23.1.

- Il y a des cas où une boîte de NAT décide de supprimer les transpositions qui sont encore actives (par exemple, l'intervalle de garde en vie est trop long, ou la boîte de NAT est réamorcée). Cela va être apparent à un hôte si il reçoit un paquet dont la protection d'intégrité se valide, mais a une adresse ou accès différents, ou les deux par rapport à ceux qui étaient associés à la SA dans le paquet validé. Quand on trouve un tel paquet validé, un hôte qui ne prend pas en charge d'autres méthodes de récupération comme le protocole de mobilité et multi rattachements IKEv2 (MOBIKE, *IKEv2 Mobility and Multihoming*) [RFC4555], et qui n'est pas derrière un NAT, DEVRAIT envoyer tous les paquets (y compris les paquets en retransmission) à l'adresse et accès IP du paquet validé, et DEVRAIT mémoriser cela comme la nouvelle combinaison d'adresse et accès pour la SA (c'est-à-dire, il DEVRAIT dynamiquement mettre à jour l'adresse). 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 qu'il ouvre la possibilité d'une 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 devrait seulement être faite 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 peuvent seulement être faites en toute sécurité si la protection contre la répétition est activée. Quand IKEv2 est utilisé avec MOBIKE, la mise à jour dynamique des adresses décrite ci-dessus interfère avec la façon dont MOBIKE récupère de la même situation. Voir plus d'informations au paragraphe 3.8 de la [RFC4555].

2.23.1 Traversée de NAT en mode transport

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



(D'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 les adresses : NAT A et NAT B. NAT A est un NAT dynamique qui transpose l'adresse de source du client IP1 en IPN1. NAT B est un NAT statique configuré à ce que les connexions venant de l'adresse IPN2 soient transposée en l'adresse IP2 de la passerelle, c'est-à-dire, 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 peut seulement le faire si il sait l'adresse externe du NAT B, c'est-à-dire, l'adresse IPN2. Si le NAT B est un NAT statique, alors son adresse peut être configurée à la configuration du client. Une autre option serait de trouver à utiliser un autre protocole (comme le DNS) mais c'est en dehors de la portée de IKEv2.

Dans ce scénario, le client et le serveur sont 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 envoyer du trafic au serveur, il peut avoir un paquet déclenchant avec l'adresse IP de source IP1, et une adresse de destination IP de IPN2. Sa base de données d'autorisation des homologues (PAD, *Peer Authorization Database*) et sa SPD doivent avoir une configuration correspondant à ces adresses (ou des entrées de caractère générique qui les couvrent). Parce que c'est le 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 de TSi doivent utiliser la gamme IP1-IP1 comme adresses IP, et toutes les entrées de 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 le protocole et les numéros d'accès, comme ceux provenant du paquet qui déclenché 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 qu'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 cherche dans la SPD sur la base des sélecteurs de trafic. Parce que IP1 ne signifie rien pour le serveur (c'est l'adresse que le client a derrière le NAT) il est inutile de faire une recherche fondée sur 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 de trouver 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 doit d'abord mémoriser les adresses IP du vieux sélecteur de trafic pour les utiliser plus tart pour la correction de la somme de contrôle incrémentaire (l'adresse IP dans le TSi peut être mémorisée comme adresse de source originale et l'adresse IP dans le TSr peut être mémorisée comme adresse de destination originale). 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 du serveur a été détectée comme étant derrière le 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, remplace IPN2 dans TSr par IP2).

Après cette substitution d'adresse, les sélecteurs de trafic et les adresses de source/destination IKE UDP semblent les mêmes, et le serveur fait une recherche de SPD sur la base de ces nouveaux sélecteurs de trafic. Si une entrée est trouvée et qu'elle permet le mode transport, cette entrée est alors utilisée. Si une entrée est trouvée mais ne permet pas le mode transport, alors le serveur PEUT défaire la substitution d'adresse et refaire la recherche SPD en utilisant les sélecteurs de trafic originaux. Si la seconde recherche réussit, le 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 recherche dans la SPD utilise les adresses qui vont être vues par l'hôte local. Cela va aussi assurer que les entrées de la base de données d'associations de sécurité (SAD, *Security Association Database*) pour la sortie du tunnel se vérifient et que les paquets en retour sont ajoutés 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 caractère générique correspondant à toutes les adresses, mais cela permet aussi de faire des entrées de SPD différentes, par exemple, pour différentes adresses externes de NAT connu.

Après la recherche de SPD, le serveur va faire le rétrécissement de sélecteur de trafic sur la base de l'entrée de SPD qu'il a trouvé. Il va à nouveau 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 encore rétrécir le nombre de protocoles ou les gammes d'accès utilisées par les sélecteurs de trafic. L'entrée de SAD créée pour la SA fille va avoir 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 les adresses originales de source et destination. Il va remplacer les adresses IP dans les sélecteurs de trafic par celles 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 vérifie la SA par rapport aux

sélecteurs de trafic envoyés, et quand il installe l'entrée de SAD.

Un résumé des règles de la traversée de NAT en mode transport est :

Pour le client qui propose le mode transport :
- les entrées de TSi DOIVENT avoir exactement une adresse IP, et elles DOIVENT correspondre à l'adresse de source de la SA IKE.
- Les entrées de TSr DOIVENT avoir exactement une adresse IP, et elles DOIVENT 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 le protocole et les numéros d'accès, comme ceux provenant du paquet qui déclenché la demande.
- Il PEUT y avoir plusieurs entrées de TSi et TSr.
- Si le mode transport a été choisi pour la SA (c'est-à-dire, si le serveur a inclus la notification USE_TRANSPORT_MODE dans sa réponse) : mémoriser les sélecteurs de trafic originaux comme adresses de source et destination reçues.
- Si le serveur est derrière un NAT, substituer à l'adresse IP dans les entrées de TSr l'adresse distante de la SA IKE.
- Si le client est derrière un NAT, substituer à l'adresse IP dans les entrées de TSi l'adresse locale de la SA IKE.
- Faire la substitution d'adresse avant d'utiliser ces sélecteurs de trafic pour autre chose que mémoriser leur contenu original. Cela inclut la vérification que les sélecteurs de trafic ont été réduits correctement par l'autre extrémité, la création de l'entrée de SAD, et ainsi de suite.

Pour celui qui répond, quand le mode transport est proposé par le client :
- Mémoriser l'adresse IP originale du sélecteur de trafic comme adresses de source et destination reçues, au cas où il serait nécessaire de défaire la substitution d'adresse, pour l'utiliser comme "adresse réelle de source et destination" spécifiée par la [RFC3948], et pour la correction de somme de contrôle TCP/UDP.
- Si le client est derrière un NAT, substituer à l'adresse IP dans les entrées de TSi l'adresse distante de la SA IKE.
- Si le serveur est derrière un NAT, substituer à l'adresse IP dans les entrées de TSr l'adresse locale de la SA IKE.
- Faire la recherche de PAD et SPD en utilisant l'identifiant et les sélecteurs de trafic substitués.
- Si aucune entrée de SPD n'est trouvée, ou (si une est trouvée) si l'entrée de SPD ne permet pas le mode transport, défaire les substitutions de sélecteur de trafic. Faire à nouveau une recherche de PAD et SPD en utilisant l'identifiant et les sélecteurs de trafic originaux, mais aussi en cherchant une entrée de SPD en mode tunnel (c'est-à-dire, revenir au mode tunnel).
- Cependant, si une entrée de SPD en mode transport est 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 spécifié à l'origine 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 de fonctionnement et de négociation (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. Précisément, les encapsuleurs et désencapsuleurs de tunnel pour toutes les SA en mode tunnel créées par IKEv2 DOIVENT prendre en charge l'option ECN de pleine fonctionnalité pour les tunnels spécifiés dans la [RFC3168] et DOIVENT mettre en œuvre le traitement d'encapsulation et désencapsulation de tunnel spécifié dans la [RFC4301] pour prévenir l'élimination des indications d'encombrement de ECN.

2.25. Collisions d'échanges

Parce que les échanges IKEv2 peuvent être initiés par l'un ou 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 de façon normale.

Évidemment, utiliser une taille de fenêtre supérieure à 1 conduit à des situations plus complexes, en particulier si des demandes sont traitées en désordre. Ce paragraphe se concentre sur les 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 comme 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 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 clé d'une SA fille qui n'existe pas. La SA dont l'initiateur a tenté de changer la clé est indiquée par le champ SPI dans la charge utile Notify, qui est copiée 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 encore) 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 d'un changement de clé ou de la clôture de SA filles

Si un homologue reçoit une demande de changement de clé d'une SA fille qu'il est en train d'essayer de clore, il DEVRAIT répondre avec une TEMPORARY_FAILURE. Si un homologue reçoit une demande de changement de clé d'une SA fille dont il est actuellement en train de changer la clé, il DEVRAIT répondre comme d'habitude, et DEVRAIT se préparer à clore plus tard les SA redondantes sur la base des noms occasionnels (voir le paragraphe 2.8.1). Si un homologue reçoit une demande de changement de clé d'une SA fille qui n'existe pas, il DEVRAIT répondre avec 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 Delete (voir le paragraphe 1.4.1). Si un homologue reçoit une demande de clore une SA fille dont il est actuellement en train de changer la clé, il DEVRAIT répondre comme d'habitude, avec une charge utile Delete. Si un homologue reçoit une demande de clore une SA fille qui n'existe pas, il DEVRAIT répondre sans charge utile Delete.

Si un homologue reçoit une demande de changement de clé de la SA IKE, et qu'il est actuellement en train de créer, de changer de clé, ou de clore une SA fille de cette SA IKE, il DEVRAIT répondre avec TEMPORARY_FAILURE.

2.25.2 Collisions lors d'un changement de clé ou de la clôture de SA IKE

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

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

Si un homologue reçoit une demande de créer ou changer la clé d'une SA fille quand il est actuellement en train de changer la clé de la SA IKE, il DEVRAIT répondre avec TEMPORARY_FAILURE. Si un homologue reçoit une demande de suppression d'une SA fille quand il est actuellement en train de changer la clé de la SA IKE, il DEVRAIT répondre comme d'habitude, avec une charge utile Delete.

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

Dans les tableaux de cette Section, certaines primitives cryptographiques et 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 pourra 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" dans les mises en œuvre qui sont destinées à être interopérables.

3.1 En-tête IKE

Les messages IKE utilisent les accès UDP 500 et/ou 4500, avec un message IKE par datagramme UDP. Les informations provenant du début du paquet jusqu'à l'en-tête UDP sont largement ignorées sauf que les adresses IP et les accès UDP provenant des en-têtes sont inversés et utilisés pour retourner les paquets. Quand ils sont envoyés sur l'accès UDP 500, les messages IKE commencent immédiatement à la suite de l'en-tête UDP. Quand ils sont envoyés sur l'accès UDP 4500, les messages IKE ont quatre octets de zéros ajoutés devant. Ces quatre octets de zéros ne font pas partie du message IKE et ne sont inclus dans aucun champ de longueur ou somme de contrôle définie 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 chacune identifiée par un champ Prochaine charge utile dans la charge utile précédente. Les charges utiles sont identifiées dans l'ordre dans lequel elles apparaissent dans le message IKE en regardant dans le champ Prochaine charge utile dans l'en-tête IKE, et ensuite conformément au champ Prochaine charge utile de la charge utile IKE elle-même jusqu'à ce qu'un champ Prochaine charge utile de zéro indique qu'aucune charge utile ne suit. Si une charge utile de type "Chiffrée" est trouvée, cette charge utile est déchiffrée et son contenu est analysé comme une charge utile supplémentaire. Une charge utile Chiffrée DOIT être la dernière charge utile dans un paquet et une charge utile Chiffrée NE DOIT PAS contenir une autre charge utile Chiffrée.

Le SPI de celui qui répond dans l'en-tête identifie 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 disposés en ordre gros boutien (aussi appelé "bit de poids fort en premier", ou "ordre des octets du réseau").

Le format de l'en-tête IKE est montré dans la Figure 4.

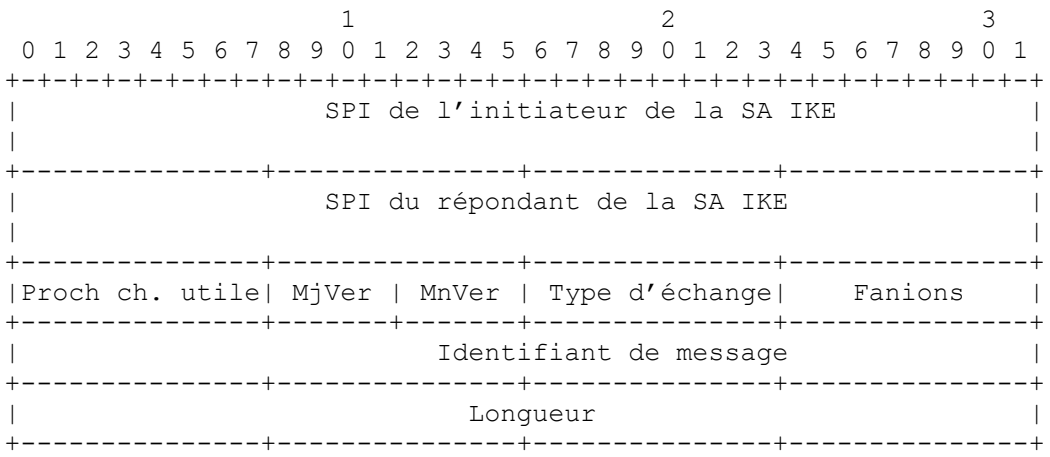


Figure 4 : Format d'en-tête IKE

o SPI de l'initiateur (8 octets) : valeur choisie par l'initiateur pour identifier une association de sécurité IKE unique. Cette valeur NE DOIT PAS être zéro.

o SPI du répondant (8 octets) : valeur choisie par celui qui répond pour identifier une association de sécurité IKE unique. Cette valeur DOIT être zéro dans le premier message d'un échange initial IKE (y compris les répétitions de ce message qui incluent un mouchard).

o 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.

o Version majeure (4 bits) : indique la version majeure du protocole IKE utilisé. Les mises en œuvre fondées sur cette version de IKE DOIVENT régler version majeure à 2. Les mises en œuvre fondées sur de précédentes versions de IKE et ISAKMP DOIT régler version majeure à 1. Les mises en œuvre fondées 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.

o Version mineure (4 bits) : indique la version mineure du protocole IKE utilisé. Les mises en œuvre fondées sur cette version de IKE DOIVENT régler Version mineure à 0. Elles DOIVENT ignorer le numéro de version mineure des messages reçus.

o Type d'échange (1 octet) : indique le type d'échange utilisé. Cela exerce une contrainte sur les charges utiles envoyées dans chaque message d'un échange. Les valeurs du tableau suivant sont seulement celles 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. Les lecteurs devraient 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

o Fanions (1 octet) : indique les options spécifiques établies pour le message. La présence d'options est indiquée par le bit approprié établi dans le champ Fanions. Ces bits sont :

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

Dans la description ci-dessous, un bit "établi" signifie que sa valeur est "1", tandis que "à zéro" signifie que sa valeur est "0". Les bits "X" DOIVENT être à zéro à l'envoi et DOIVENT être ignorés à réception.

* R (Réponse) : ce bit indique que ce message est une réponse à un message contenant le même identifiant de message. Ce bit DOIT être à zéro 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 marqué comme réponse (avec une exception : voir le paragraphe 2.21.2).

* V (Version) : ce bit indique que l'émetteur est capable de parler 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 mettre à zéro ce bit à l'envoi et DOIT 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 à zéro dans les messages envoyés par le répondant original. Il est utilisé par le receveur pour déterminer quels huits octets du SPI ont été générés par le receveur. Ce bit change pour refléter qui a initié le dernier changement de clé de la SA IKE.

o Identifiant de message (4 octets, entier non signé) : identifiant de message utilisé pour contrôler les retransmissions des paquets perdus et faire correspondre les demandes et les réponses. Il est essentiel pour la sécurité du protocole parce qu'il est utilisé pour prévenir les attaques de répétition de message. Voir les paragraphes 2.1 et 2.2.

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

3.2 En-tête générique de charge utile

Chaque charge utile IKE définie dans les 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 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. utile|C|   Réservé    | Longueur de charge utile      |
+---------------+-+-------------+---------------+---------------+
```

Figure 5 : En-tête générique de charge utile

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

o Prochaine charge utile (1 octet) : identifiant du type de charge utile de la prochaine charge utile dans le message. Si la charge utile en cours est la dernière du message, ce champ va être 0. Ce champ fournit une capacité de "chaînage" par laquelle des charges utiles supplémentaires peuvent être ajoutées à un message en incluant chacune à la fin du message et en établissant le champ Prochaine charge utile de la charge utile précédente pour 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 de charge utile supplémentaire. Dans l'en-tête de la 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. Les valeurs de type de charge utile sont mentionnées ci-dessous. Les valeurs du tableau suivant sont seulement les valeurs courantes 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. Les lecteurs devraient 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 clé	KE	34
Identification – Initiateur	IDi	35
Identification – Répondant	IDr	36
Certificat	CERT	37
Demande de certificat	CERTREQ	38
Authentification	AUTH	39
Nom occasionnel	Ni, Nr	40
Notify	N	41
Delete	D	42
ID 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 de sorte qu'il n'y a pas de chevauchement avec les allocations de code de IKEv1.)

o 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 précédente charge utile. 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. DOIT être ignoré par le receveur si le receveur comprend le code de type de charge utile. 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 courante plutôt qu'à la charge utile "suivante" dont le code de type apparaît dans le premier octet. Le raisonnement pour 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 le paragraphe 2.5 pour plus d'informations sur ce bit.

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

o Longueur de charge utile (2 octets, entier non signé) : longueur en octets de la charge utile courante, 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

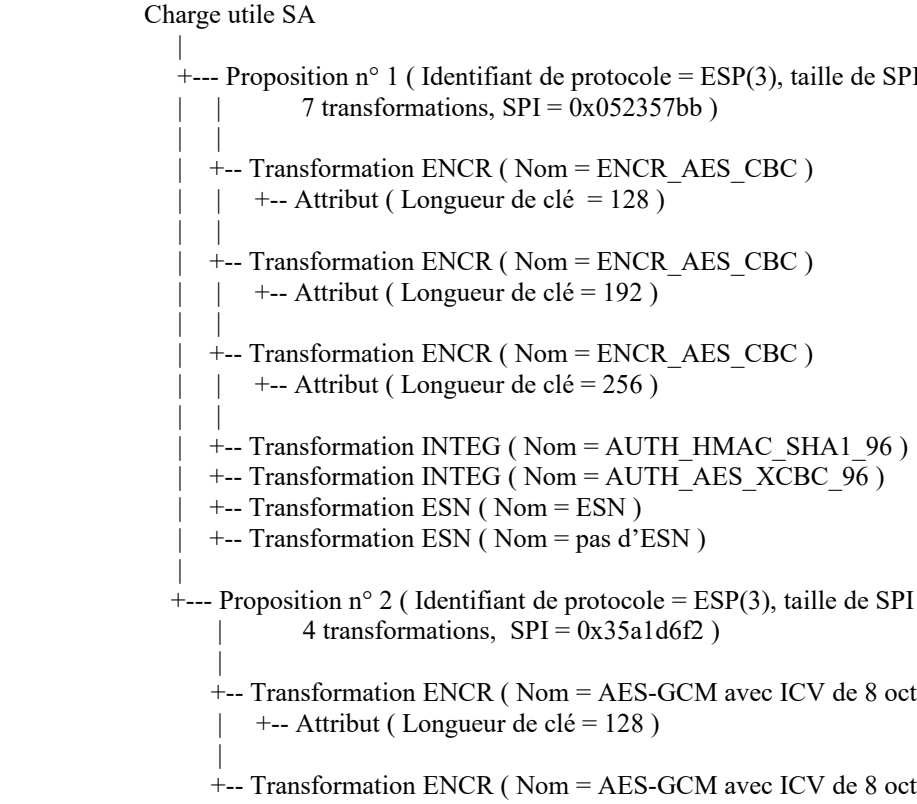
La charge utile Association de sécurité, notée SA dans le présent document, est utilisée pour négocier les attributs d'une association de sécurité. L'assemblage des charges utiles Association de sécurité exige beaucoup de calme. Une charge utile

SA PEUT contenir plusieurs propositions. Si il y en a plus d'une, elles DOIVENT être ordonnées de la plus préférée à la moins préférée. 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 de charge utile totale est cohérente avec les longueurs et comptes internes de la charge utile. Les propositions, les transformations, et les attributs ont chacun leur propre codage de longueur variable. Ils sont incorporés de telle façon que la longueur de charge utile d'une SA inclue le contenu combiné des informations de SA, proposition, transformation, et attribut. La longueur d'une proposition inclut les longueurs de toutes les transformations et de tous les attributs qu'elle contient. La longueur d'une transformation inclut les longueurs de tous les attributs qu'elle contient.

La syntaxe de Associations de sécurité, Propositions, Transformations, et Attributs se fonde sur ISAKMP ; cependant, leur sémantique est un peu différente. La raison de cette complexité et de la hiérarchie est de permettre que plusieurs combinaisons possibles d'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 d'algorithmes. Par exemple, un initiateur pourrait vouloir proposer d'utiliser ESP avec soit (3DES et HMAC_MD5) soit (AES et HMAC_SHA1).

Une des raisons pour laquelle la sémantique de la charge utile SA a changé par rapport à ISAKMP et IKEv1 est de rendre les codages plus compacts dans les cas courants.

La structure 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 de l'utilisation de plusieurs propositions est de proposer à la fois des chiffrements standard et des chiffrements en mode combiné. Les chiffrements en mode combiné incluent 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 "NONE", aucun algorithme d'intégrité étant la méthode RECOMMANDÉE. Si un initiateur veut proposer à la fois des chiffrements en mode combiné et des chiffrements normaux, il doit inclure deux propositions : une va avoir tous les chiffrements en mode combiné, et l'autre va avoir tous 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 bits en mode chaînage de bloc de chiffrement (CBC, *Cipher Block Chaining*) avec HMAC-SHA1-96 ou 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, mais n'exigent pas, l'utilisation de numéros de séquence étendus (ESN, *Extended Sequence Number*). Cela peut être illustré comme suit :



```
|    +-- Attribut ( Longueur de clé = 256 )
|
+-- Transformation ESN ( Nom = ESN )
+-- Transformation ESN ( Nom = pas d'ESN )
```

Chaque structure Proposition/Protocole est suivie par une ou plusieurs structures Transformation. Le nombre des différentes transformations est généralement déterminé par le protocole. AH a généralement deux transformations : ESN 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 est le nombre d'identifiants de transformation alloués, qui apparaît 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, en proposant ESP avec (3DES ou AES-CBC) et (HMAC_MD5 ou HMAC_SHA) la proposition ESP contiendrait deux transformation candidates de type 1 (une pour 3DES et une pour AEC-CBC) et deux transformations candidates de type 3 (une pour HMAC_MD5 et une pour HMAC_SHA). Cela propose effectivement quatre combinaisons d'algorithmes. Si l'initiateur voulait proposer seulement un sous ensemble d'elles, par exemple (3DES et HMAC_MD5) ou (IDEA et HMAC_SHA) il n'y aurait pas de moyen de coder cela comme plusieurs transformations dans une seule proposition. À la place, l'initiateur va devoir construire deux propositions différentes, chacune avec deux transformations.

Une 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 tailles de clé 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.

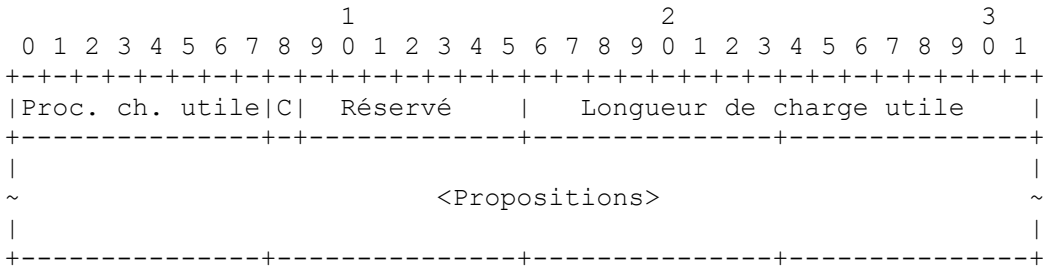
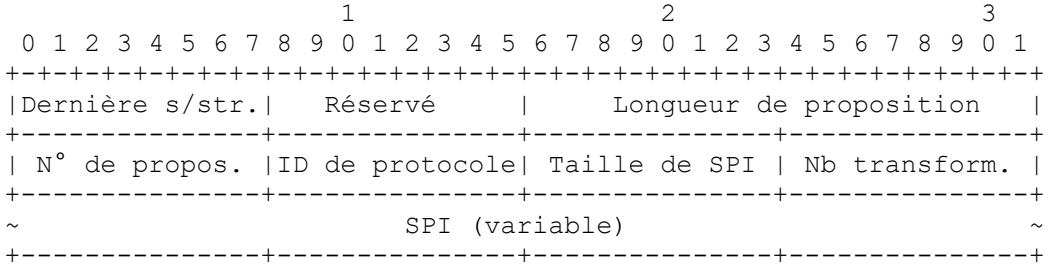


Figure 6 : Charge utile Association de sécurité

o Propositions (variable) - une ou plusieurs sous structures Proposition.

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

3.3.1 Sous structure Proposition



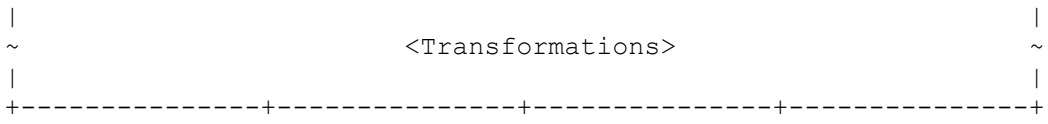


Figure 7 : Sous structure Proposition

o Dernière sous structure (1 octet) : spécifie si c'est ou non la dernière sous structure Proposition dans la SA. Ce champ a une valeur de 0 si c'est la dernière sous structure Proposition, et une valeur de 2 si il y a plus de sous structures Proposition. Cette syntaxe est héritée de ISAKMP, mais n'est pas nécessaire parce que la dernière proposition pourrait être identifiée à partir de la longueur de la SA. La valeur (2) correspond à un type de charge utile de Proposition dans IKEv1, et les quatre premiers octets de la structure Proposition la font ressembler à l'en-tête d'une charge utile.

o Réservé (1 octet) : DOIT être envoyé à zéro ; DOIT être ignoré à réception.

o Longueur de proposition (2 octets, entier non signé) : longueur de cette proposition, incluant toutes les transformations et attributs qui suivent.

o 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 précédente proposition (indiquant un OU des 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.

o Identifiant de protocole (1 octet) : spécifie l'identifiant de protocole IPsec pour la négociation en cours. Les valeurs du tableau suivant sont seulement celles en cours à la date de publication de la RFC 4306. D'autres valeurs peuvent avoir été ajoutées depuis ou vont être ajoutées après la publication du présent document. Les lecteurs devraient se référer à [IKEV2IANA] pour les dernières valeurs.

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

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

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

o SPI (variable) : SPI de l'entité envoyeuse. Même si la taille de SPI n'est pas un multiple de 4 octets,il n'y a pas de bourrage 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.

o Transformations (variable) : une ou plusieurs sous structures Transformation.

3.3.2 Sous structure Transformation

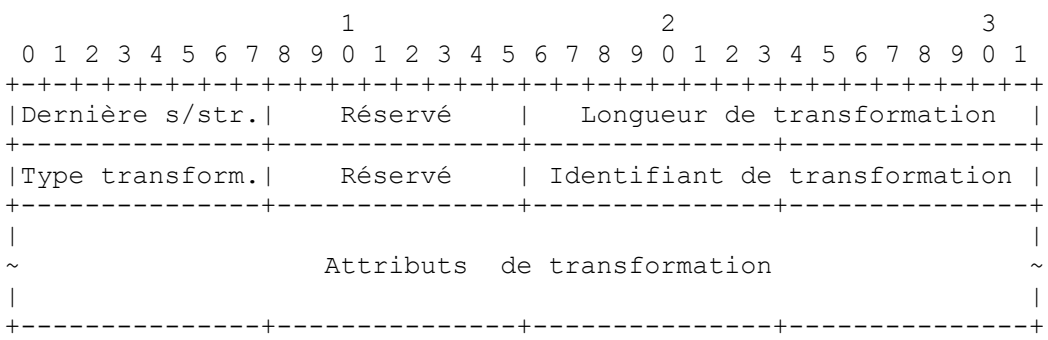


Figure 8 : Sous structure Transformation

o Dernière sous structure (1 octet) : spécifie si c'est ou non la dernière sous structure Transformation dans la proposition. Ce champ a la valeur 0 si c'est la dernière sous structure Transformation, et la valeur 3 si il y a plus de sous structures Transformation. Cette syntaxe est héritée de ISAKMP, mais n'est pas nécessaire parce que la dernière transformation pourrait être identifiée de la longueur de la proposition. La valeur (3) correspond à un type de charge utile de transformation de IKEv1, et les quatre premiers octets de la structure Transformation la font ressembler un peu à l'en-tête d'une charge utile.

o Réservé : DOIT être envoyé à zéro ; DOIT être ignoré à réception.

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

o Type de transformation (1 octet) : type de la transformation spécifiée dans cette transformation. Différents protocoles prennent en charge différents types de transformations. 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'utilisation de la transformation facultative pour celui qui répond, il inclut une sous structure Transformation avec l'identifiant de transformation = 0 comme une des options.

o Identifiant de transformation (2 octets) : instance spécifique du type de transformation proposé.

La liste des valeurs de type de transformation est donnée ci-dessous. Les valeurs dans le tableau suivant sont seulement celles en cours 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.

Description	**Type de transformation**	**Utilisé 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
Numéro de séquence étendu (ESN)	5	AH et ESP

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

Pour le type de transformation 1 (Algorithme de chiffrement) les identifiants de transformation sont mentionnées ci-dessous. Les valeurs dans le tableau suivant sont seulement celles en cours 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**
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) les identifiants de transformation sont mentionnés ci-dessous. Les valeurs dans le tableau suivant sont seulement celles en cours 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
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) les identifiant de transformation définis sont mentionnés ci-dessous. Les valeurs dans le tableau suivant sont seulement celles en cours 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
NONE	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) les identifiants de transformation définis sont mentionnés ci-dessous. Les valeurs dans le tableau suivant sont seulement celles en cours 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
NONE	0	
Groupe MODP à 768 bits	1	Appendice B
Groupe MODP à 1024 bits	2	Appendice B
Groupe MODP à1536 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 d'échange Diffie-Hellman, un groupe Diffie-Hellman PEUT être négocié pour la SA fille. Cela permet aux homologues d'employer Diffie-Hellman dans l'échange CREATE_CHILD_SA, fournissant le secret parfait de transmission pour les clés de SA fille générées.

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

Pour le type de transformation 5 (Numéro de séquence étendu) les identifiants de transformation définis sont mentionnés ci-dessous. Les valeurs dans le tableau suivant sont seulement celles en cours 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
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 au registre "Paramètres de l'échange de clé Internet version 2 (IKEv2)" de l'IANA pour les détails.

3.3.3. Types de transformation valides par protocole

Le nombre et le type de transformations qui accompagnent une charge utile SA dépendent du protocole dans la SA elle-même. Une charge utile SA proposant 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 pris en charge (bien qu'elle n'ait pas besoin d'accepter les propositions qui ont des suites inacceptables). Une proposition PEUT omettre les types facultatifs si la seule valeur qu'elle va accepter pour eux est NONE.

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 Chiffrée spécifié dans le présent document. Par exemple, la [RFC5282] spécifie des formats supplémentaires fondés sur le chiffrement authentifié, dans lequel un algorithme d'intégrité séparé n'est pas négocié.

3.3.4 Identifiant de transformation obligatoires

La spécification de suites qui DOIVENT et DEVRAIENT être prises en charge pour l'interopérabilité a été retirée du présent document parce que elles vont probablement 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'elle pourrait être mise à jour à l'avenir, et que d'autres RFC pourraient spécifier des ensembles de suites différents.

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

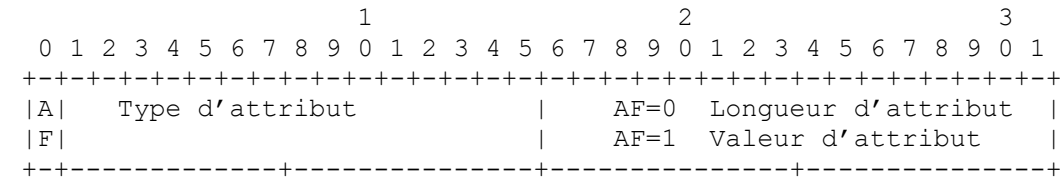
Il est probable que l'IANA va ajouter à l'avenir des transformations supplémentaires, et certains utilisateurs peuvent vouloir utiliser des suites privées, en particulier pour IKE où les mises en œuvre devraient être capables de prendre en charge des paramètres différents, jusqu'à certaines limites de taille. En vue de ce but, toutes les mises en œuvre de IKEv2 DEVRAIENT inclure une facilité de gestion qui permette la spécification (par un utilisateur ou l'administrateur du système) de paramètres Diffie-Hellman (le générateur, le module, 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 utilisateur ou administrateur de système) pour permettre de négocier de tels groupes.

Toutes les mises en œuvre de IKEv2 DOIVENT inclure une facilité de gestion qui permette à un utilisateur ou administrateur de système de spécifier les suites dont l'utilisation est acceptable avec 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 localement configurés 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 contrôle de suite IKE. Noter que les suites de chiffrement qui DOIVENT être mises en œuvre n'ont pas besoin d'être configurées comme acceptables pour la politique locale.

3.3.5 Attributs de transformation

Chaque transformation dans une charge utile 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-après).

Les attributs sont des paires type/valeur et sont définis ci-dessous. Les attributs peuvent avoir une valeur avec une longueur fixe de deux octets ou une valeur de longueur variable. Pour cette dernière, l'attribut est codé comme type/longueur/valeur.



```
|                   AF=0  Valeur d'attribut                     |
|                   AF=1  Non transmis                          |
+---------------+---------------+---------------+---------------+
```

Figure 9 : Attributs de données

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

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

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

Le seul type d'attribut actuellement défini (Longueur de clé) 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 étant de longueur fixe même si leur valeur peut tenir dans deux octets. Note : ceci change de IKEv1, où une souplesse accrue a pu simplifier la composition des messages mais a certainement compliqué la tâche de l'analyseur.

Les valeurs dans le tableau suivant sont seulement celles en cours 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.

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 aux valeurs correspondantes.

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

o 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 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.

o Certaines transformations spécifient que l'attribut Longueur de clé DOIT être toujours inclus (en omettant quand 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.

o 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 une meilleure 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 qui sont réputées 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 plus grande longueur de clé, la mise en œuvre DEVRAIT accepter l'offre si elle supporte l'utilisation d'une clé plus longue.

La prise en charge de cette capacité permet à celui qui répond 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 le paragraphe suivant) un initiateur qui veut accepter plusieurs longueurs de clé doit inclure plusieurs transformations avec le même type de transformation, chacun 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 choisi un seul ensemble complet de paramètres provenant des offres (ou rejeter toutes les offres si aucune n'est acceptable). Si il y a plusieurs propositions, celui qui répond DOIT choisir une seule proposition. Si la proposition choisie a plusieurs transformations avec le même type, celui qui répond DOIT en choisir une seule. Tous les attributs d'une transformation choisie DOIVENT être retournés sans modification. L'initiateur d'un échange DOIT vérifier que l'offre acceptée est cohérente avec une de ses propositions, et sinon DOIT terminer l'échange.

Si celui qui répond reçoit une proposition qui contient un type de transformation qu'il ne comprend pas, ou une proposition à qui il manque un type de transformation obligatoire, il DOIT considérer que cette proposition est inacceptable ; cependant, les autres propositions dans la même charge utile SA sont traitées comme d'habitude. De même, si celui qui répond reçoit une transformation qu'il ne comprend pas, ou une qui contient un attribut de transformation qu'il ne comprend pas, il DOIT considérer cette transformation comme 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.

La négociation des groupes Diffie-Hellman présente des défis particuliers. Les offres de SA incluent des attributs proposés et un numéro public Diffie-Hellman (KE) dans le même message. Si dans l'échange initial l'initiateur offre d'utiliser un de plusieurs groupes Diffie-Hellman, il DEVRAIT en prendre un que celui qui répond va le plus probablement accepter et inclure un KE correspondant à ce groupe. Si celui qui répond choisit une proposition utilisant un groupe Diffie-Hellman différent (autre que NONE) il va indiquer le groupe correct dans la réponse et l'initiateur DEVRAIT prendre un élément de ce groupe comme valeur de KE quand il réessaye le premier message. Il DEVRAIT cependant continuer de proposer son ensemble de groupes pleinement pris en charge afin de prévenir une attaque en dégradation par interposition. Si une des propositions offertes est pour le groupe Diffie-Hellman de NONE, et si celui qui répond choisit ce groupe Diffie-Hellman, il DOIT alors ignorer la charge utile KE de l'initiateur et omettre la charge utile KE de la réponse.

3.4 Charge utile d'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 de charge utile IKE générique suivi par la valeur publique Diffie-Hellman elle-même.

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

Une charge utile Échange de clé est construite en copiant une valeur publique Diffie-Hellman dans la portion "Données d'échange de clé" de la charge utile. La longueur de la valeur publique Diffie-Hellman pour les groupes MODP DOIT être égale à la longueur du module de nombre premier sur lequel l'exponentiation a été effectuée, en ajoutant des bits de 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 le paragraphe 3.3.2). Ce numéro de groupe Diffie-Hellman DOIT correspondre à un groupe Diffie-Hellman spécifié dans une proposition de la charge utile SA qui a été envoyée dans le même message, et DEVRAIT correspondre au groupe Diffie-Hellman dans le premier groupe de la première proposition, si elle existe. Si aucune des propositions dans cette charge utile SA ne spécifie un groupe Diffie-Hellman, la charge utile KE NE DOIT PAS être présente. Si la proposition choisie utilise un groupe Diffie-Hellman différent (autre que NONE) 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 IDi et IDr dans le présent document, permettent aux homologues d'affirmer l'un l'autre une identité. Cette identité peut être utilisée pour une recherche de politique, mais ne doit pas nécessairement correspondre à quelque chose 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 des 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 dans les charges utiles TSi/TSr. Les contenus de IDi/IDr sont utilisés uniquement pour aller chercher les données de politique et d'authentification relatives à l'autre partie.

Note : dans IKEv1, deux charges utiles ID étaient utilisées dans chaque direction pour contenir les informations de sélecteur de trafic (TS) pour les données passant sur la SA. Dans IKEv2, cette information est portée dans les charges utiles TS (voir le paragraphe 3.13).

La base de données d'autorisation des homologues (PAD, *Peer Authorization Database*) décrite dans la [RFC4301] décrit l'utilisation de la charge utile ID dans IKEv2 et fournit un modèle formel pour le lien d'une identité à la politique en plus de fournir des services qui traitent plus spécifiquement des détails de l'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 RFC 4301.

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

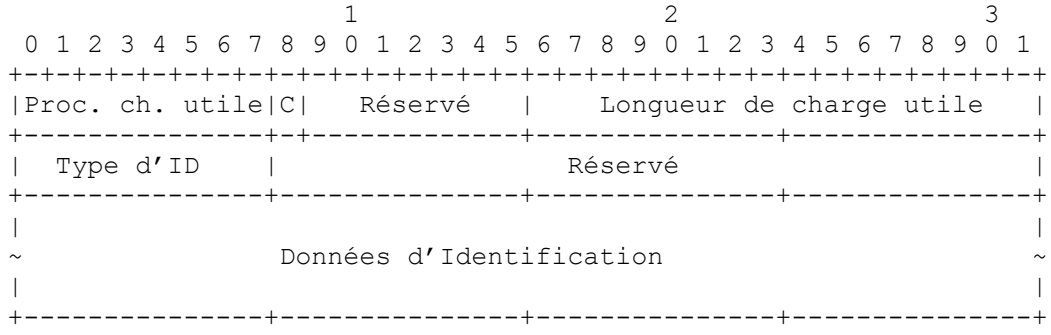


Figure 11 : Format de charge utile Identification

o Type d'ID (1 octet) : spécifie le type d'identification utilisé.

o Réservé : DOIT être envoyé à zéro ; DOIT être ignoré à réception.

o Données d'identification (longueur variable) : valeur, indiquée par le type d'identification. La longueur des données d'identification est calculée à partir de la taille dans l'en-tête de charge utile ID.

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

Le tableau suivant donne la liste des significations allouées au champ Type d'identification. Les valeurs dans le tableau suivant sont seulement celles en cours 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.

Type d'identifiant	Valeur	Signification
ID_IPV4_ADDR	1	Une seule adresse IPva de quatre (4) octets.
ID_FQDN	2	Chaîne de nom de domaine pleinement qualifié. Un exemple de ID_FQDN est "exemple.com". La chaîne NE DOIT PAS contenir de terminaison (par exemple, NULL, CR, etc.). Tous les caractères dans le 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 de messagerie pleinement qualifiée de la RFC 822. Un exemple de ID_RFC822_ADDR est "jsmith@exemple.com". La chaîne NE DOIT PAS contenir de terminaison. À cause de la [RFC6532], les mises en œuvre devraient traiter ce champ comme du texte codé en UTF-8, pas comme du pur ASCII.
ID_IPV6_ADDR	5	Adresse IPv6 de seize (16) octets.
ID_DER_ASN1_DN	9	Codage binaire en DER d'un nom distinctif ASN.1 X.500 [RFC5280].
ID_DER_ASN1_GN	10	Codage binaire en DER d'un nom général ASN.1 X.509 [RFC5280].
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 propriétaires.

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 capables de IPv6 DOIVENT de plus être configurables à accepter ID_IPV6_ADDR. Les mises en œuvre seulement IPv6 PEUVENT être configurables à envoyer seulement ID_IPV6_ADDR au lieu de ID_IPV4_ADDR pour les adresses IPv6.

EAP [RFC3748] ne rend pas obligatoire l'utilisation d'un type particulier d'identifiant, mais souvent EAP est 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 (par exemple, "joe@exemple.com") la syntaxe n'est pas exactement la même que celle d'une adresse de messagerie dans 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 à la syntaxe exacte de la [RFC5322], mais devraient plutôt accepter tout NAI d'apparence raisonnable. Pour les NAI qui n'incluent pas de composant de domaine, le type d'identification ID_KEY_ID DEVRAIT être utilisé.

Voir le "Profil Internet de PKI de sécurité IP de IKEv1/ISAKMP, IKEv2, et PKIX" ([RFC4945]) pour plus d'informations sur la confrontation des charges utiles Identification et du contenu des certificats PKIX.

3.6 Charge utile Certificat

La charge utile Certificat, notée CERT dans le présent document, fournit un moyen de transport des certificats ou autres informations relatives à l'authentification via IKE. Les charges utiles Certificat DEVRAIENT être incluses dans un échange si des certificats sont disponibles chez l'envoyeur. Les formats Hachage et URL des charges utiles Certificat devraient être utilisés quand l'homologue a indiqué la capacité de restituer ces informations 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 :

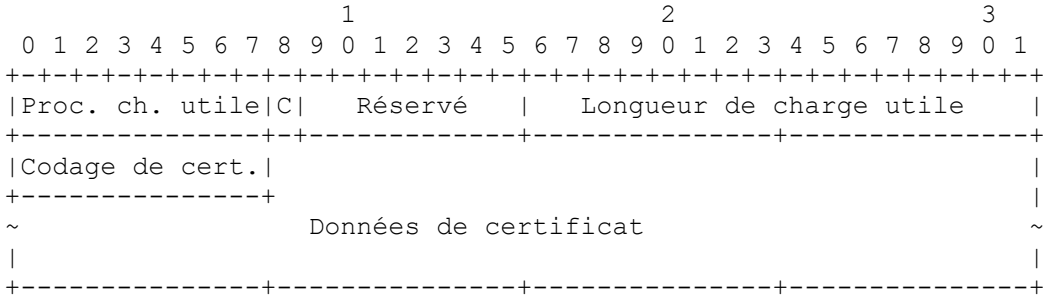


Figure 12 : Format de charge utile Certificat

o Codage de certificat (1 octet) : ce champ indique le type de certificat ou les informations relatives au certificat contenues dans le champ Données de certificat. Les valeurs dans le tableau suivant sont seulement celles en cours 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.

Codage de certificat	Valeur	Syntaxe
Certificat X.509 enveloppé dans PKCS n° 7	1	NON SPÉCIFIÉ
Certificat PGP	2	NON SPÉCIFIÉ
Clé signée DNS	3	NON SPÉCIFIÉ
Certificat X.509 – Signature	4	
Jeton Kerberos	6	NON SPÉCIFIÉ
Liste de révocation de certificat (CRL)	7	
Liste de révocation d'autorité (ARL)	8	NON SPÉCIFIÉ
Certificat SPKI	9	NON SPÉCIFIÉ
Certificat X.509 – Attribut	10	NON SPÉCIFIÉ
déconseillé (clé RSA brute)	11	DÉCONSEILLÉ
Hachage et URL de certificat X.509	12	
Hachage et URL de faisceau X.509	13	

o Données de certificat (longueur variable) : codage réel des données du 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 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 a besoin d'être envoyée, plusieurs charges utiles CERT sont utilisées, dont seulement la première contient la clé publique utilisée pour valider la charge utile AUTH de l'envoyeur.

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

o Les codages de hachage et d'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 codée en DER elle-même. Cela améliore l'efficacité quand les points d'extrémité ont les données de certificat en antémémoire et rend IKE moins sujet à des attaques de DoS qui deviennent plus faciles à monter quand les messages IKE sont assez grands pour exiger la fragmentation IP [DOSUDPPROT].

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

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

ÉTIQUETTES EXPLICITES DE DÉFINITIONS  ::=
DÉBUT

IMPORTE
Certificate, CertificateList
  DE PKIX1Explicit88
    { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(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 à

l'appui de l'authentification, et DOIVENT aussi être capables d'être configurées à envoyer et accepter les deux formats 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 pas spécifié actuellement, et de tels schémas NE DEVRAIENT PAS être utilisés en l'absence d'un document qui les spécifient.

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_SA_INIT 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 :

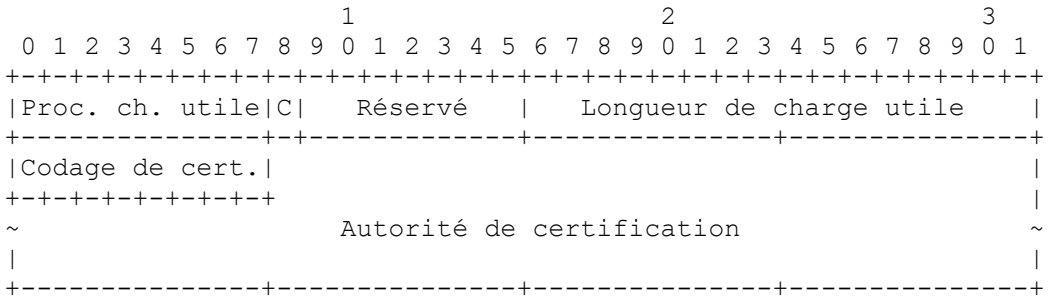


Figure 13 : Format de charge utile Demande de certificat

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

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

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

Le champ Codage de certificat a les mêmes valeurs que celles définies au paragraphe 3.6. Le champ Autorité de certification contient un indicateur des autorités de confiance pour ce type de certificat. La valeur de l'autorité de certification est une liste enchaînée de hachages SHA-1 des clés publiques des autorités de certification (CA, *Certification Authority*) de confiance. Chacune est codée comme le hachage SHA-1 de l'élément Informations de clé publique de sujet (voir le 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 avant que soient publiées des spécifications sur la voie de la normalisation qui spécifient leur usage.

Noter que le terme "demande de certificat" est un peu trompeur parce que des valeurs autres que de certificats sont définies dans une charge utile "Certificat" et les 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 en inspectant le 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. Cela peut être une chaîne de certificats.

Si un certificat d'entité d'extrémité existe satisfaisant les critères spécifiés dans la 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 a la permission d'envoyer une charge utile CERT,
- o a une politique correspondante de CA de confiance qui gouverne la négociation en cours, et
- o a au moins un chaînage de certificat d'entité d'extrémité à usage unique approprié à l'usage à une CA fournie dans la CERTREQ.

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

L'intention est de ne pas empêcher la communication par le strict respect du choix d'un certificat fondé sur la CERTREQ, quand un certificat de remplacement pourrait être choisi par l'envoyeur qui permettrait quand même au receveur de le valider et lui faire confiance grâce à la confiance portée par la certification croisée, les CRL, ou autres moyens configurés hors bande. Donc, le traitement d'une CERTREQ devrait être vu comme une suggestion d'un certificat à choisir, non obligatoire. Si aucun certificat n'existe, la CERTREQ est ignorée. Ce 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 pourrait être acceptable (peut-être sur l'invite d'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 donc préférerait probablement 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 :

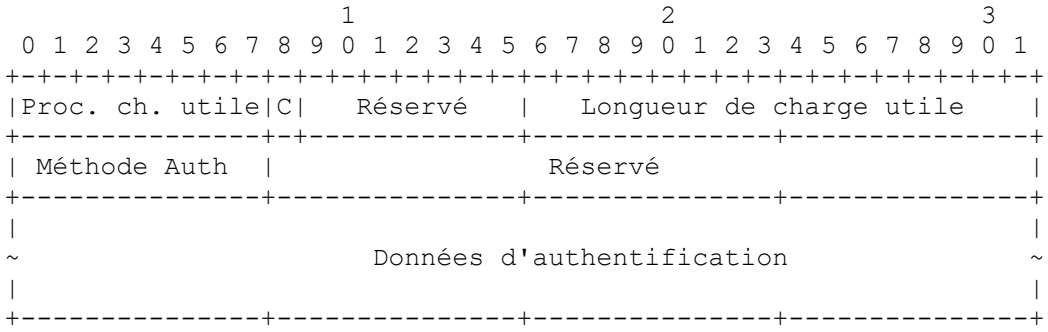


Figure 14 : Format de charge utile Authentification

- o Méthode Auth (1 octet) : spécifie la méthode d'authentification utilisée. Les types de signatures sont mentionnés ici. Les valeurs dans le tableau suivant sont seulement celles en cours 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.

Mécanisme	Valeur	Signification
Signature numérique RSA	1	Calculé 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] (on notera 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 accepter les signatures qui utilisent SHA-1 comme fonction de hachage et DEVRAIENT utiliser SHA-1 comme fonction de hachage par défaut en générant des signatures. Les mises en œuvre peuvent utiliser les certificats reçus d'un homologue comme indication 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 tout algorithme de hachage utilisé dans le ou les certificats.
Code d'intégrité de message de clé partagée	2	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.
Signature numérique DSS	3	Calculé comme spécifié au paragraphe 2.15 en utilisant une clé privée DSS (voir [DSS]) sur un hachage SHA-1.

o Réservé : DOIT être envoyé à zéro ; DOIT être ignoré à réception.

o Données d'authentification (longueur variable) : voir le 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 de celui qui répond, contient des données aléatoires utilisées pour garantir la vie durant un échange et protéger contre les attaques en répétition.

La charge utile Nom occasionnel est définie comme suit :

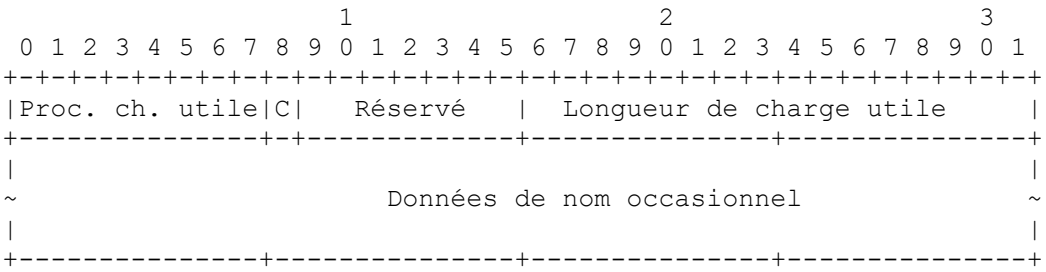


Figure 15 : Format de charge utile Nom occasionnel

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

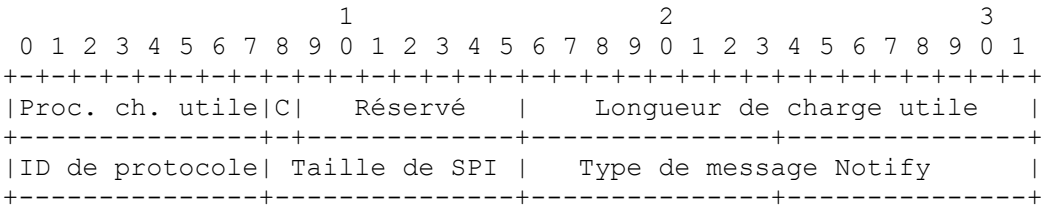
Le type de charge utile pour la charge utile Nom occasionnel est quarante (40).

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 Notify

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



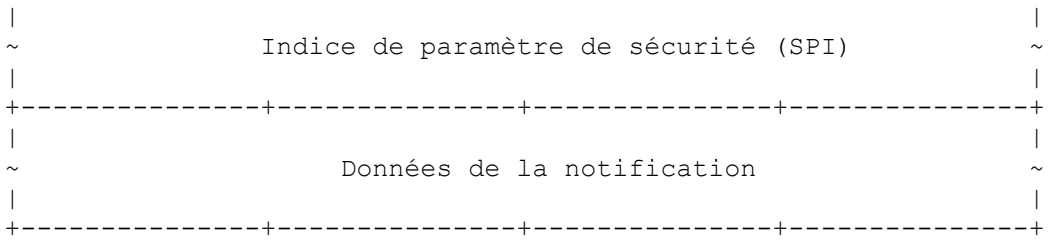


Figure 16 : Format de charge utile Notify

o 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 est inclus seulement avec INVALID_SELECTORS, REKEY_SA, et CHILD_SA_NOT_FOUND. Si le champ SPI est vide, ce champ DOIT être envoyé à zéro pour indiquer NONE et DOIT être ignoré à réception.

o 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, la taille de SPI DOIT être zéro et le champ doit être vide.

o Type de message Notify (2 octets) : spécifie le type du message de notification.

o SPI (longueur variable) : indice de paramètre de sécurité.

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

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

3.10.1 Types de message Notify

Les informations de notification peuvent être des messages d'erreur qui spécifient pourquoi une SA ne pourrait pas être établie. Ce peut aussi être des données d'état qu'un processus gérant une base de données de SA souhaite communiquer à un processus homologue.

Le tableau ci-dessous fait la liste des messages de notification et de leurs valeurs correspondantes. Le nombre des différents états d'erreur a été largement réduit par rapport à IKEv1 à la fois pour simplifier et pour éviter de donner des informations de configuration 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 Notify avec un de ces types qu'elle 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.

Les charges utiles Notify avec des types d'état PEUVENT être ajoutées à tout message et DOIVENT être ignorées si elles ne sont pas reconnues. Elles sont destinées à indiquer les capacités, et au titre d'une négociation de SA, sont utilisées pour négocier des paramètres non cryptographiques.

Plus d'informations sur le traitement d'erreur se trouvent au paragraphe 2.21.

Les valeurs dans le tableau suivant sont seulement 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 seront ajoutées après la publication du présent document. Les lecteurs devraient se référer à [IKEV2IANA] pour les dernières valeurs.

Messages NOTIFY : types d'erreur	**Valeur**	
UNSUPPORTED_CRITICAL_PAYLOAD	1	Voir le paragraphe 2.5.
INVALID_IKE_SPI	4	Voir le paragraphe 2.21.
INVALID_MAJOR_VERSION	5	Voir le paragraphe 2.5.
INVALID_SYNTAX	7	Indique que le message IKE reçu est invalide parce que un type,

		longueur, ou valeur était hors gamme ou parce que la demande a été rejetée pour des raisons de politique. Pour éviter une attaque de DoS utilisant des messages falsifiés, cet état peut seulement être retourné pour et dans un paquet chiffré si l'identifiant de message et la somme de contrôle cryptographique sont valides. Pour éviter des fuites d'informations 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 enregistrées.
INVALID_MESSAGE_ID	9	Voir le paragraphe 2.3.
INVALID_SPI	11	Voir le paragraphe 1.5.
NON_PROPOSAL_CHOSEN	14	Aucune des suites de chiffrement proposées n'est acceptable. Cela peut être envoyé dans tous les cas où les propositions offertes (incluant mais sans se limiter aux valeurs de charge utile SA, USE_TRANSPORT_MODE, IPCOMP_SUPPORTED) ne sont pas acceptables pour celui qui répond. Cela peut aussi être utilisé comme erreur "génériquc" de SA fille quand la SA fille ne peut pas être créée pour une 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 échoue. Il n'y a pas de données associées. Voir aussi le paragraphe 2.21.2.
SINGLE_PAIR_REQUIRED	34	Voir le paragraphe 2.9.
NON_ADDITIONAL_SAS	35	Voir le paragraphe 1.3.
INTERNAL_ADDRESS_FAILURE	36	Voir le paragraphe 3.15.4.
FAILED_CP_REQUIRED	37	Voir le paragraphe 2.19.
TS_UNACCEPTABLE	38	Voir le 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 qui 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 le paragraphe 2.25.
CHILD_SA_NOT_FOUND	44	Voir le paragraphe 2.25.

Messages NOTIFY : types d'état	**Valeur**	
INITIAL_CONTACT	16384	Voir le paragraphe 2.4.
SET_WINDOW_SIZE	16385	Voir le paragraphe 2.3.
ADDITIONAL_TS_POSSIBLE	16386	Voir le paragraphe 2.9.
IPCOMP_SUPPORTED	16387	Voir le paragraphe 2.22.
NAT_DETECTION_SOURCE_IP	16388	Voir le paragraphe 2.23.
NAT_DETECTION_DESTINATION_IP	16389	Voir le paragraphe 2.23.
COOKIE	16390	Voir le paragraphe 2.6.
USE_TRANSPORT_MODE	16391	Voir le paragraphe 1.3.1.
HTTP_CERT_LOOKUP_SUPPORTED	16392	Voir le paragraphe 3.6.
REKEY_SA	16393	Voir le paragraphe 1.3.3.
ESP_TFC_PADDING_NOT_SUPPORTED	16394	Voir le paragraphe 1.3.1.
NON_FIRST_FRAGMENTS_ALSO	16395	Voir le paragraphe 1.3.1.

3.11 Charge utile Delete

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

La suppression de la 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 attend dans les paquets ESP ou AH entrants.

La charge utile Delete est définie comme suit :

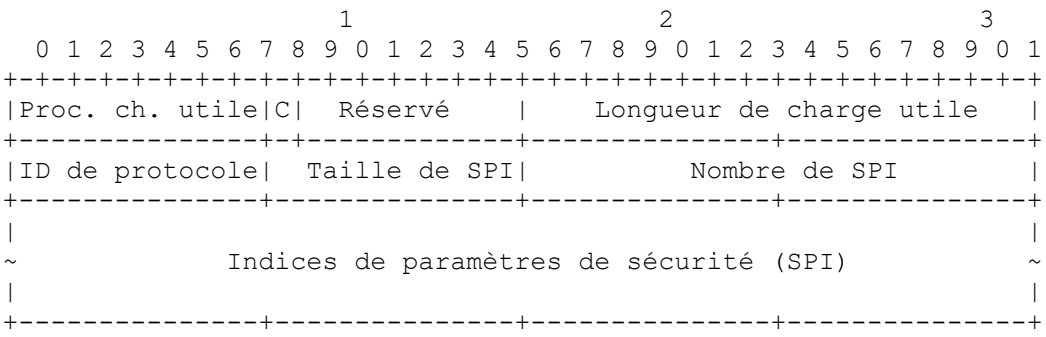


Figure 17 : Format de charge utile Delete

o Identifiant de protocole (1 octet) : DOIT être 1 pour une SA IKE, 2 pour AH, ou 3 pour ESP.

o Taille de SPI (1 octet) : longueur en octets du SPI comme défini par l'identifiant de protocole. DOIT être zéro pour IKE (SPI est un en-tête de message) ou quatre pour AH et ESP.

o Nombre de SPI (2 octets, entier non signé) : nombre de SPI contenus dans la charge utile Delete. La taille de chaque SPI est définie par le champ Taille de SPI.

o Indices de paramètre de sécurité (longueur variable) : identifie la ou les associations de sécurité à supprimer. La longueur de ce champ est déterminée par les champs Taille de SPI et Nombre de SPI.

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

3.12 Charge utiIe 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 maintenant la rétro compatibilité.

Une charge utile Identifiant de fabricant PEUT annoncer que l'envoyeur est capable d'accepter certaines extensions au protocole, ou PEUT simplement identifier la mise en œuvre pour aider au débogage. Une charge utile Identifiant de fabricant NE DOIT PAS changer l'interprétation d'informations définies dans la présente 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 d'utiliser des numéros à usage privé décrits dans le présent document, comme des charges utiles privées, des échanges privés, des notifications privées, etc. Les identifiants de fabricant non familiers DOIVENT être ignorés.

Les auteurs 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. On s'attend à ce que les documents qui sont acceptés et sont normalisés vont donner des "numéros magiques" dans la gamme utilisation future de l'IANA, et l'exigence d'utiliser un Identifiant de fabricant va disparaître.

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

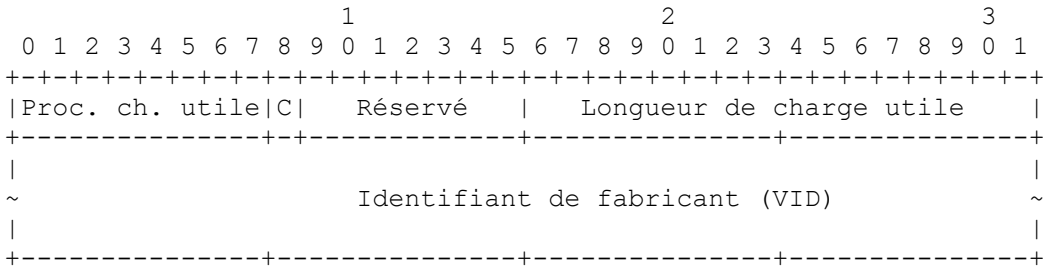


Figure 18 : Format de charge utile Identifiant de fabricant

o Identifiant de fabricant (longueur variable) : il est de la responsabilité de la personne qui choisit l'identifiant de fabricant d'assurer son unicité en dépit de l'absence d'un registre central des identifiants. Une bonne pratique est d'inclure un nom d'entreprise, un nom de personne, ou des informations de ce genre. Si on veut s'appliquer, on pourrait inclure la latitude et longitude et l'heure du choix de l'identifiant et des entrées 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 de charge utile IKE générique suivi par des sélecteurs de trafic individuels comme suit :

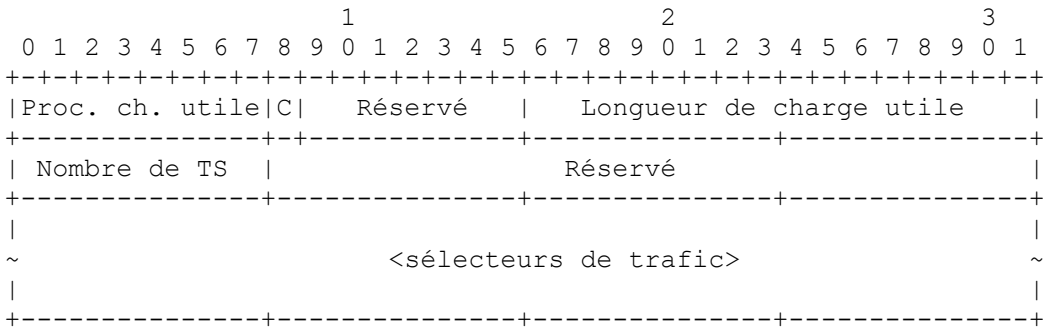


Figure 19 : Format de charge utile Sélecteurs de trafic

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 à l'extrémité de l'initiateur de la SA et quarante cinq (45) pour les adresses à l'extrémité de celui qui répond.

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 quelconque 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

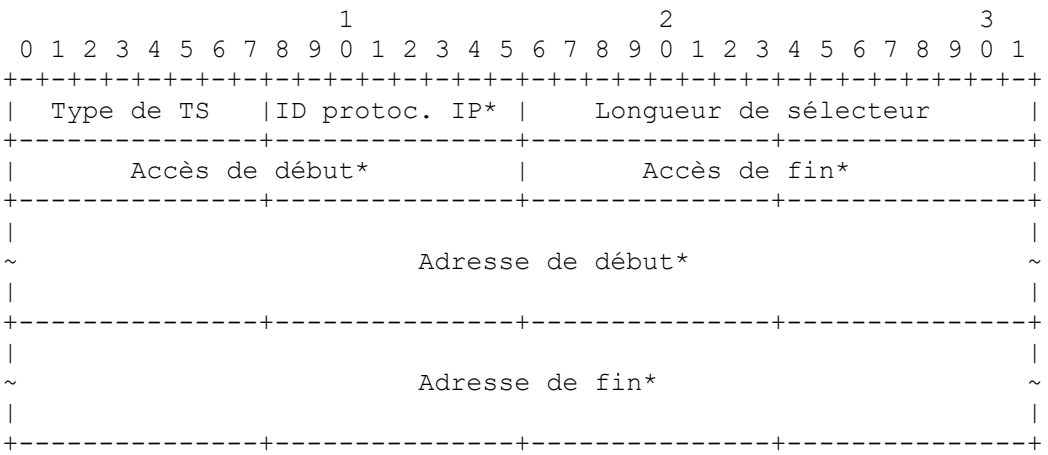


Figure 20 : Sélecteur de trafic

*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.

o Type de TS (un octet) : spécifie le type du sélecteur de trafic.

o Identifiant 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.

o Longueur de sélecteur (2 octets, entier non signé) : spécifie la longueur de cette sous structure de sélecteur de trafic incluant l'en-tête.

o Accès de début (2 octets, entier non signé) : valeur qui spécifie le plus petit numéro d'accès permis par ce sélecteur de trafic. Pour les protocoles où 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 code ICMP et ICMPv6, ainsi que les valeurs de type d'en-tête de mobilité 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 plus fort poids et Code dans les huit bits de moindre poids. Les valeurs de type d'en-tête de mobilité MIPv6 sont traitées comme un seul numéro d'accès entier de 16 bits, avec Type dans les huit bits de poids fort et les huit bits de moindre poids réglés à zéro.

o Accès de fin (2 octets, entier non signé) : valeur spécifiant le plus grand numéro d'accès permis par ce sélecteur de trafic. Pour les protocoles où 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 d'en-tête de mobilité 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 plus fort poids et Code dans les huit bits de moindre poids. Les valeurs de type d'en-tête de mobilité MIPv6 sont traitées comme un seul numéro d'accès entier de 16 bits, avec Type dans les huit bits de poids fort et les huit bits de moindre poids réglés à zéro.

o Adresse de début : plus petite adresse incluse dans ce sélecteur de trafic (longueur déterminée par le type de TS).

o Adresse de fin : plus grande adresse incluse dans ce sélecteur de trafic (longueur déterminée par le type de TS).

Les systèmes conformes à la [RFC4301] qui souhaitent indiquer "ANY" accès DOIVENT régler l'accès de début à 0 et l'accès de fin à 65535 ; noter que selon la [RFC4301], "ANY" inclut "OPAQUE". Les systèmes qui fonctionnent avec la [RFC4301] qui souhaitent indiquer des accès "OPAQUE", mais pas des accès "ANY", DOIVENT régler l'accès de débur 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 Type MH pour l'en-tête de mobilité IPv6 [RFC6275]. Noter, cependant, que ni les paquets ICMP ni MIPv6 n'ont de champs source et destination séparés. 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].

Le tableau suivant donne les valeurs pour le champ Type de sélecteur de trafic et les données correspondantes de sélecteur d'adresse. Les valeurs dans le tableau suivant sont seulement celles en cours 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.

Type de TS	**Valeur**	
TS_IPV4_ADDR_RANGE	7	Gamme d'adresses IPv4, représentée par deux valeurs de quatre octets. La première valeur est l'adresse IPv4 de début (incluse) et la seconde valeur est l'adresse IPv4 de fin (incluse). Toutes les adresses 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 l'adresse IPv6 de début (incluse) et la seconde valeur est l'adresse IPv6 de fin (incluse). Toutes les adresses entre les deux adresses spécifiées sont considérées être dans la liste.

3.14 Charge utile Chiffrée

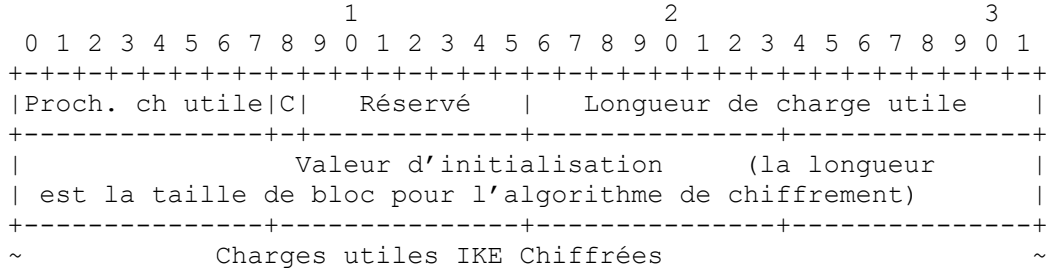
La change utile Chiffrée, notée SK {...} dans le présent document, contient d'autres charges utiles en forme chiffrée. La change utile Chiffrée, si elle est présente dans un message, DOIT être la dernière charge utile du message. Souvent, c'est la seule charge utile du message. Cette charge utile est 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 Chiffrées 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. Le concept est modélisé d'après les algorithmes ESP décrits dans les RFC [RFC2104], [RFC4303], et [RFC2451]. Le présent document spécifie complètement le traitement cryptographique des données IKE, mais ces documents devraient être consultés sur les raisons du concept. De futurs documents pourront spécifier le traitement des charges utiles Chiffrées pour d'autres types de transformations, comme le chiffrement en mode compteur et les algorithmes de chiffrement authentifié. Les homologues NE DOIVENT PAS négocier des transformations pour lesquelles il n'existe pas une telle spécification.

Quand un algorithme de chiffrement authentifié est utilisé pour protéger la SA IKE, la construction de la charge utile Chiffrée 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 Chiffrée est quarante six (46). La charge utile Chiffrée consiste en l'en-tête de charge utile IKE générique suivi par des champs individuels comme suit :



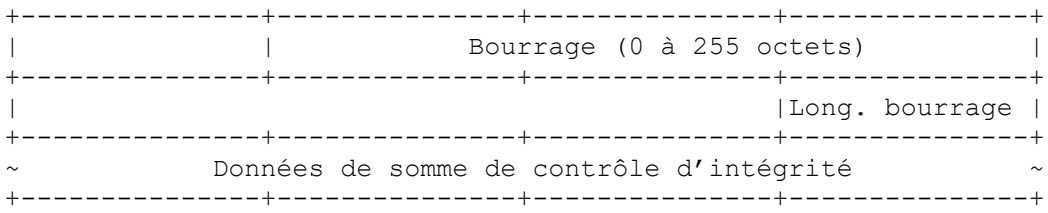


Figure 21 : Format de charge utile Chiffrée

o Prochaine charge utile : type de charge utile de la première charge utile incorporée. Noter que ceci est une exception au format d'en-tête standard, car la charge utile Chiffrée est la dernière charge utile dans le message et donc le champ Prochaine charge utile va normalement être zéro. Mais parce que le contenu de cette charge utile est des charges utiles incorporées et qu'il n'y avait pas d'endroit naturel où placer le type de la première, ce type est placé ici.

o 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.

o 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 une nouvelle IV imprévisible pour chaque message ; les receveurs DOIVENT accepter toute valeur. Le lecteur est encouragé à consulter [MODES] pour un avis sur la génération d'une IV. En particulier, utiliser le bloc final de texte chiffré du message précédent n'est pas considéré comme imprévisible. Pour les modes autres que CBC, le format et le traitement de l'IV est spécifié dans le document qui spécifie l'algorithme et le mode de chiffrement.

o Les charges utiles IKE sont comme spécifié plus haut dans ce paragraphe. Ce champ est chiffré avec le chiffrement négocié.

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

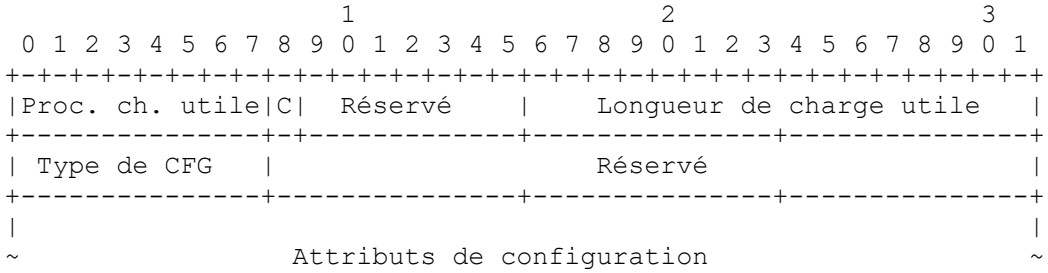
o Longueur de bourrage est la longueur du champ Bourrage. L'envoyeur DEVRAIT régler la longueur de bourrage à la valeur minimum qui rend la combinaison des charges utiles, du bourrage, et de la longueur du bourrage un multiple de la taille de bloc, mais le receveur DOIT accepter toute longueur qui résulte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

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

3.15 Charge utile 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 pour un IRAC de demander une adresse IP interne d'un IRAS et d'échanger d'autres informations de cette sorte qu'on acquerrait avec le protocole dynamique de configuration d'hôte (DHCP, *Dynamic Host Configuration Protocol*) si l'IRAC était directement connecté à un LAN.

La charge utile Configuration est définie comme suit :



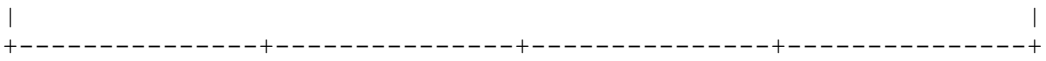


Figure 22 : Format de charge utile Configuration

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

o Type de CFG (1 octet) : type de l'échange représenté par les attributs de configuration. Les valeurs dans le tableau suivant sont seulement celles en cours 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.

Type 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

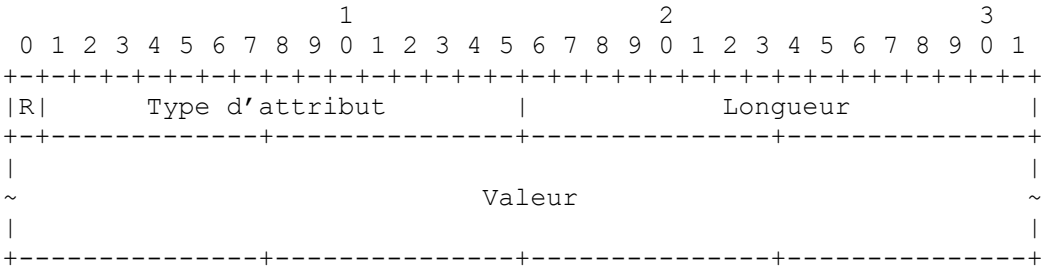


Figure 23 : Format d'attribut Configuration

o Réservé (1 bit) : ce bit DOIT être réglé à zéro et DOIT être ignoré à réception.

o Type d'attribut (15 bits) : identifiant unique pour chacun des types 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. Voici la liste des types d'attributs.

Les valeurs dans le 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 supprimées par la RFC 5996). 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.

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 plus

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 plusieurs valeurs étaient demandées.

o INTERNAL_IP4_ADDRESS, INTERNAL_IP6_ADDRESS : adresse sur le réseau interne, parfois appelée 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, elle indique probablement qu'une connexion précédente existait avec cette adresse et que le demandeur aimerait 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. Celui qui répond PEUT seulement envoyer le nombre d'adresses demandé. 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 que la SA IKE (ou ses successeurs après changement de clé) qui demande l'adresse est valide. Ceci est décrit plus en détails au paragraphe 3.15.3.

o INTERNAL_IP4_NETMASK : gabarit de réseau du réseau interne. Un seul gabarit de réseau est permis dans les messages de demande et réponse (par exemple, 255.255.255.0) et il DOIT être utilisé seulement 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 pour ces adresses à travers moi") mais implique aussi une limite de liaison. Par exemple, le client pourrait utiliser sa propre adresse et le gabarit de 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 cette information (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.

o INTERNAL_IP4_DNS, INTERNAL_IP6_DNS : spécifie une adresse d'un serveur DNS dans le réseau. Plusieurs serveurs DNS PEUVENT être demandés. Celui qui répond PEUT répondre avec zéro, un ou plusieurs attributs de serveur DNS.

o INTERNAL_IP4_NBNS : spécifie une adresse d'un serveur de noms NetBios (NBNS) dans le réseau. Plusieurs serveurs NBNS PEUVENT être demandés. Celui qui répond PEUT répondre avec zéro, un ou plusieurs attributs serveur NBNS.

o 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. Celui qui répond PEUT répondre avec zéro, un ou plusieurs attributs Serveur DHCP.

o APPLICATION_VERSION : informations de version ou application de l'hôte IPsec. C'est une chaîne de caractères ASCII imprimables c'est-à-dire NON terminée par un NUL.

o INTERNAL_IP4_SUBNET : sous réseaux protégés par cet appareil de bordure. Cet attribut est constitué de deux champs dont le premier est une adresse IP et le second un gabarit de réseau. Plusieurs sous réseaux PEUVENT être demandés. Celui qui répond PEUT répondre avec zéro, un ou plusieurs attributs Sous réseau. Ceci est discuté plus en détails au paragraphe 3.15.2.

o SUPPORTED_ATTRIBUTES : quand il est utilisé dans une demande, cet attribut DOIT être de longueur zéro et spécifie une interrogation de celui qui répond pour qu'il renvoie tous les attributs qu'il prend en charge. La réponse contient un attribut qui porte un ensemble d'identifiants d'attribut chacun sur 2 octets. La longueur divisée par 2 (octets) va déclarer le nombre d'attributs pris en charge contenus dans la réponse.

o INTERNAL_IP6_SUBNET : sous réseaux protégés que cet appareil de bordure protège. Cet attribut 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]. Plusieurs sous réseaux PEUVENT être demandés. Celui qui répond PEUT répondre avec zéro, un ou plusieurs attributs 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 comment une mise en œuvre représente en fait les informations envoyées dans une réponse. C'est-à-dire, on ne recommande aucune méthode spécifique pour qu'un IRAS détermine quel serveur DNS devrait être retourné à un IRAC demandeur.

La paire CFG_REQUEST et CFG_REPLY permet à un point d'extrémité IKE de demander l'information à son homologue. Si un attribut dans la CFG_REQUEST de la charge utile Configuration n'est pas de longueur zéro, il est pris comme une suggestion pour cet attribut. La CFG_REPLY de la charge utile Configuration PEUT retourner cette valeur, ou une nouvelle. Elle PEUT aussi ajouter de nouveaux attributs et ne pas inclure certains demandés. Les attributs non reconnus ou non pris en charge DOIVENT être ignorés dans les demandes et les réponses.

La paire CFG_SET et CFG_ACK permet à un point d'extrémité IKE de pousser les données de configuration à son homologue. Dans ce cas, le CFG_SET de la charge utile Configuration contient les attributs que l'initiateur veut que son homologue altère. Celui qui répond DOIT retourner une charge utile Configuration si il accepte des données de configuration, et la charge utile Configuration DOIT contenir les attributs que celui qui répond a accepté avec des données de longueur zéro. Les attributs qu'il n'a pas accepté NE DOIVENT PAS être dans le CFG_ACK de la charge utile Configuration. Si aucun attribut n'a été accepté, celui qui répond DOIT retourner une charge utile CFG_ACK vide ou un message de réponse sans charge utile CFG_ACK. Il n'y a pas d'utilisation actuellement définie pour l'échange CFG_SET/CFG_ACK, bien qu'il puisse être utilisé en connexion avec les 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, ceux qui ont besoin d'une ou plusieurs SA séparées, qui peuvent être atteints à travers la passerelle qui annonce les attributs. Les attributs INTERNAL_IP4/6_SUBNET peuvent aussi exprimer la politique de la passerelle sur le trafic qui devrait être envoyé à travers elle ; le client peut choisir si d'autre trafic (couverts par TSr, mais pas dans INTERNAL_IP4/6_SUBNET) est envoyé à travers la passerelle ou directement à la destination. Donc, le trafic pour les adresses mentionnées dans les attributs INTERNAL_IP4/6_SUBNET devrait être envoyé à travers la passerelle qui annonce les attributs. Si il n'y a pas de SA fille existante dont les sélecteurs de trafic couvrent l'adresse en question, de nouvelles SA doivent 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 :

```
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 ce cas, le INTERNAL_IP4_SUBNET ne porte pas réellement d'information utile.

Une réponse différente possible aurait été :

```
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 ne se soucie pas de si le client envoie du trafic non inclus par INTERNAL_IP4_SUBNET directement à la destination (sans passer par la passerelle).

Une situation différente apparaît 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 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, 192.0.2.155-192.0.2.155)

alors 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)

Parce que la signification de INTERNAL_IP4_SUBNET/INTERNAL_IP6_SUBNET dans les demandes de configuration n'est pas claire, elles ne peuvent pas être utilisées fiablement dans les CFG_REQUEST.

3.15.3 Charges utiles Configuration pour IPv6

Les charges utiles Configuration pour IPv6 sont fondées 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'un document suplémentaire discute de la configuration IPv6 dans IKEv2, la [RFC5739]. Pour l'instant, c'est un document expérimental, mais il y a un espoir qu'avec plus d'expérience de mise en œuvre, il obtiendra le même traitement normalisé 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'est pas acceptable, la passerelle tente d'utiliser l'identifiant d'interface avec un autre préfixe ; si même cela échoue, la passerelle choisit un autre identifiant d'interface.

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

Bien que cette approche de la configuration d'adresses IPv6 soit raisonnablement simple, elle a quelques limitations. Les tunnels IPsec configurés en utilisant IKEv2 ne sont pas des "interfaces" pleinement caractérisées au sens de l'architecture d'adressage IPv6 [RFC4291]. En particulier, ils 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 celui qui répond rencontre une erreur en tentant d'allouer une adresse IP à l'initiateur durant le traitement d'une charge utile Configuration, il répond avec une notification INTERNAL_ADDRESS_FAILURE. La SA IKE est créée même si la SA fille initiale ne peut pas être créée à cause de cet échec. Si cette erreur est générée dans un échange IKE_AUTH, aucune SA fille ne va être créée. Cependant, il y a des cas d'erreur plus complexes.

Si celui qui répond ne prend pas du tout en charge les charges utiles Configuration, il peut simplement les ignorer toutes. Ce type de mise en œuvre n'envoie jamais de notification 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 d'adresse particulier (IPv4 ou IPv6) que celui qui répond ne prend pas en charge, même si celui qui répond prend en charge les charges utiles Configuration. Dans ce cas, celui qui répond ignore simplement le type d'adresses 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 celui qui répond prend en charge, et si certaines des demandes (mais pas toutes) échouent, celui qui répond retourne seulement les adresses qui réussissent. Celui qui répond n'envoie 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 comme un échange INFORMATIONAL séparé après une temporisation convenable, ou il PEUT supprimer la SA IKE en envoyant une charge utile Delete dans un échange INFORMATIONAL séparé et réessayer plus tard la SA IKE depuis le début après une certaine temporisation. Une telle temporisation ne devrait pas être trop courte (en particulier si la SA IKE est démarrée depuis le début) parce que ces situations d'erreur peuvent ne pas pouvoir être corrigées rapidement ; la temporisation devrait probablement être de plusieurs minutes. Par exemple, un problème de manque d'adresses chez celui qui répond va probablement seulement être corrigé quand plus d'entrées sont retournées au réservoir d'adresses lorsque d'autres clients se déconnectent ou quand celui qui répond est reconfiguré avec un plus grand réservoir d'adresses.

3.16 Charge utile 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 les extensions suivantes à ce protocole. Quand on utilise EAP, une méthode EAP appropriée doit être choisie. Beaucoup de ces méthodes ont été définies, en spécifiant l'utilisation du protocole avec divers mécanismes d'authentification. La liste des types de méthode EAP figure dans [EAP-IANA]. Un bref résumé du format EAP est inclus ici.

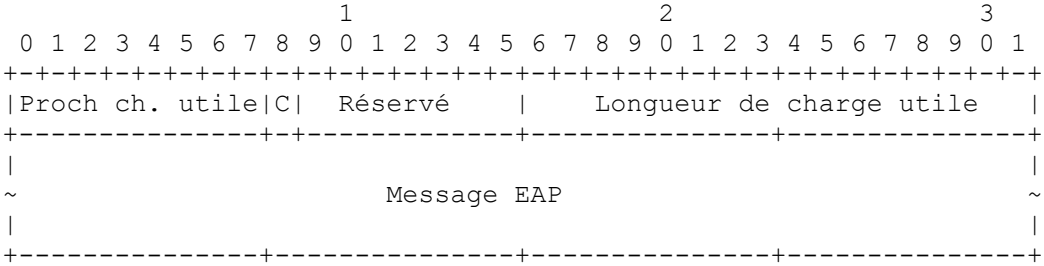


Figure 24 : Format de charge utile EAP

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

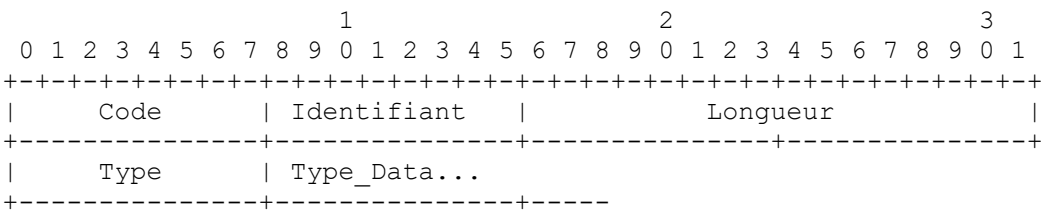


Figure 25 : Format de message EAP

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

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

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

o Type (1 octet) : présent seulement si le champ Code est demande (1) ou réponse (2). Pour les autres codes, la longueur de message EAP DOIT être de quatre octets et les champs Type et Type_Data NE DOIVENT PAS être présents. Dans un message de demande (1) Type indique les données demandées. Dans un message de réponse (2) Type DOIT être NAK ou correspondre au type des données demandé. Noter que comme IKE passe une indication de l'identité de l'initiateur dans le premier message d'échange IKE_AUTH, celui qui répond NE DEVRAIT PAS envoyer de demandes Identité EAP (type 1). L'initiateur PEUT, cependant, répondre à de telles demandes si il en reçoit.

o Type_Data (longueur variable) : varie avec le type de demande et la réponse associée. Pour la documentation des méthodes EAP, voir la [RFC3748].

4. Exigences de conformité

Afin d'assurer que toutes les mises en œuvre de IKEv2 puissent interopérer, il y a des exigences de "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 à établir complètement les SA. Ainsi une mise en œuvre particulière peut être configurée avec un nombre quelconque de restrictions concernant les algorithmes et les autorités de confiance qui vont empêcher une interopérabilité universelle.

IKEv2 est conçu pour permettre que des mises en œuvre minimales puissent interopérer avec toutes les mises en œuvre conformes. Les caractéristiques suivantes peuvent être omises dans une mise en œuvre minimale :
- o Capacité de négocier des SA à travers un NAT et tunneler la SA ESP résultante sur UDP.
- o Capacité de demander (et répondre à une demande) d'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 (si seulement de 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 pris en charge, toutes les mises en œuvre DOIVENT rejeter les messages contenant ces charges utiles.

Chaque mise en œuvre DOIT être capable de faire des échanges de quatre messages IKE_SA_INIT et IKE_AUTH établissant deux SA (une pour IKE, une pour ESP ou AH). Les mises en œuvre PEUVENT être d'initier seulement ou répondre seulement 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 Chiffrée sans charge utile dedans). Une mise en œuvre minimale PEUT prendre en charge l'échange CREATE_CHILD_SA seulement pour autant qu'elle reconnaît les demandes et les rejette avec une charge utile Notify de type NON_ADDITIONAL_SAS. Une mise en œuvre minimale n'a pas besoin d'être capable d'initier des échanges CREATE_CHILD_SA ou INFORMATIONAL. Quand une SA arrive à expiration (sur la base des valeurs configurées en local de durée de vie ou d'octets passés) une mise en œuvre PEUT soit essayer de la renouveler avec un échange CREATE_CHILD_SA, soit PEUT supprimer (clore) la vieille SA et en créer une nouvelle. Si celui qui répond rejette la demande CREATE_CHILD_SA avec une notification NON_ADDITIONAL_SAS, la mise en œuvre DOIT 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'adresses IP temporaires ou de répondre 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'une adresse du type approprié, elle DOIT retourner une charge utile CP de type CFG_REPLY contenant une adresse du type demandé. Celui qui répond peut inclure tous autres attributs en rapport.

Pour qu'une mise en œuvre soit dite conforme à la présente spécification, il DOIT être possible de la configurer à accepter ce qui suit :

- L'infrastructure de clé publique utilisant des certificats X.509 (PKIX) contenant, et signés par, des clés RSA de 1024 ou 2048 bits, où l'identifiant passé est un de ID_KEY_ID, ID_FQDN, ID_RFC822_ADDR, ou ID_DER_ASN1_DN.
- L'authentification de clés partagées où l'identifiant passé est un de ID_KEY_ID, ID_FQDN, ou ID_RFC822_ADDR.
- L'authentification où celui qui répond est authentifié en utilisant des certificats PKIX et l'initiateur est authentifié en utilisant l'authentification de clé partagée.

5. Considérations sur la sécurité

Alors que ce protocole est conçu pour minimiser la divulgation des informations de configuration aux homologues non authentifiés, une telle divulgation est partiellement inévitable. Un des homologues doit s'identifier d'abord et prouver d'abord son identité. Pour éviter les sondages, l'initiateur d'un échange doit d'abord s'identifier, et est généralement obligé de s'authentifier en premier. L'initiateur peut cependant apprendre que celui qui répond prend en charge IKE et quels protocoles de chiffrement il prend en charge. Celui qui répond (ou quelqu'un qui se fait passer pour celui qui répond) non seulement peut sonder l'initiateur sur son identité mais peut, en utilisant des 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, celui qui répond prouve son identité avant que l'initiateur le fasse, donc un initiateur qui connaît le nom d'un initiateur valide pourrait sonder celui qui répond sur son nom et ses certificats.

Répéter le changement de clé en utilisant CREATE_CHILD_SA sans échanges Diffie-Hellman supplémentaires laisse toutes les SA vulnérables à la cryptanalyse d'une seule clé. Les mises en œuvre devraient noter ce fait et établir une limite aux échanges CREATE_CHILD_SA entre les exponentiations. Le présent document ne prescrit pas une telle limite.

La force d'une clé déduite d'un échange Diffie-Hellman en 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 de pas moins de 200 bits, est d'utilisation courante avec 3DES. Le groupe cinq fournit une plus grande sécurité que le groupe deux. Le groupe un est seulement pour des raisons historiques et ne fournit pas une force suffisante sauf pour être utilisé avec DES, qui est aussi seulement pour une utilisation historique. Les mises en œuvre devraient noter ces estimations quand elles établissent une politique et négocient les 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 plus forts groupes (limitée par la force des autres algorithmes négociés y compris de la PRF). En fait, le cadre extensible de IKE encourage à la définition de plus de groupes ; l'utilisation de groupes de courbes elliptiques peut largement augmenter la force en utilisant des nombres beaucoup plus petits.

On suppose que tous les exposants Diffie-Hellman sont supprimés de la mémoire après usage.

Les échanges IKE_SA_INIT et IKE_AUTH se produisent avant que l'initiateur soit authentifié. Par suite, une mise en œuvre de ce protocole doit être complètement robuste quand elle est déployée sur un réseau non sûr. Les vulnérabilités de mise en œuvre, en particulier aux attaques de DoS, peuvent être exploitées par des homologues non authentifiés. Ce problème est particulièrement préoccupant 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 dépend de façon critique de l'aléa des paramètres choisis au hasard. Ils devraient être générés par une source d'aléa forte ou avec un germe pseudo aléatoire approprié (voir la [RFC4086]). Les mises en œuvre devraient veiller à assurer que l'utilisation de nombres aléatoires pour les clés et les noms occasionnels est construite d'une façon qui ne sape pas la sécurité des clés.

Pour des informations sur les raisons de nombreux choix de conception cryptographique dans ce protocole, voir [SIGMA] et [SKEME]. Bien que la sécurité des SA filles négociées ne dépendent pas de la force du chiffrement et de la protection de l'intégrité négociées dans la SA IKE, les mises en œuvre NE DOIVENT PAS négocier NONE comme algorithme de protection de l'intégrité IKE ou ENCR_NULL comme algorithme de chiffrement IKE.

Quand on utilise des clés pré-partagées, une considération critique est comment assurer l'aléa de ces secrets. La plus forte pratique 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, nom, ou autre source à faible entropie n'est pas sûr. Ces sources sont sujettes à des 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 de IPv4 est seulement de 32 bits, et qu'il est généralement très dispersé, il serait possible à un attaquant de trouver l'adresse interne utilisée derrière la boîte de NAT en essayant toutes les adresses IP possibles et d'essayer de trouver le hachage correspondant. Les numéros d'accès sont normalement fixés à 500, et les SPI peuvent être extraits du paquet. Cela réduit le nombre de calculs de hachage à 2^32. Avec une supposition éclairée de l'utilisation de l'espace d'adresses privées, le nombre de calculs de hachage est beaucoup plus petit. Les concepteurs ne devraient donc pas supposer que l'utilisation de IKE ne va pas laisser fuir des informations d'adresse interne.

Quand on utilise une méthode EAP d'authentification qui ne génère pas de clé partagée pour protéger une charge utile AUTH suivante, certaines attaques par interposition et usurpation d'identité de serveur sont possibles [EAPMITM]. Ces vulnérabilités se produisent quand EAP est aussi utilisé dans les protocoles qui ne sont pas protégés avec un tunnel sûr. Comme EAP est un protocole d'authentification générique, qui est souvent utilisé pour fournir des facilités d'une seule signature, une solution IPsec déployée qui s'appuie sur une méthode EAP d'authentification qui ne génère pas de clé partagée (aussi appelée une méthode EAP non génératrice de clés) peut être compromise à cause du déploiement d'une application sans aucun rapport qui se trouve aussi utiliser la même méthode EAP non génératrice de clés, mais de façon non protégée. Noter que cette vulnérabilité n'est pas limitée à EAP, mais peut se produire dans d'autres scénarios où une infrastructure d'authentification est réutilisée. Par exemple, si le mécanisme EAP utilisé par IKEv2 utilise un jeton authentifiant, une attaquant interposé pourrait se faire passer pour le serveur de la Toile, intercepter l'échange de jeton d'authentification, et l'utiliser pour initier une connexion IKEv2. Pour cette raison, l'utilisation de méthodes EAP non génératrices de clés DEVRAIT être évitée lorsque possible. Lorsque elles sont utilisées, il est extrêmement important que tous les usages de ces méthodes EAP DEVRAIENT utiliser un tunnel protégé, où l'initiateur valide le certificat de celui qui répond avant d'initier l'authentification EAP. Les mises en œuvre devraient décrire les vulnérabilités de l'utilisation des méthodes EAP non génératrices de clés dans la documentation de leur application afin que les administrateurs déployant des solutions IPsec soient conscients de ces dangers.

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

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 mémoires tampon de réassemblage. Les chances que cela réussisse peuvent être minimisées en utilisant les codages de hachage et d'URL au lieu d'envoyer les certificats (voir le paragraphe 3.6). Des atténuations supplémentaires sont discutées dans [DOSUDPPROT].

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 fournisse une grande liberté d'action pour définir la politique de sécurité pour l'identité d'un homologue de confiance, les accréditifs, 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 des homologues (PAD, *Peer Authorization Database*) quand on détermine quelles sortes de SA filles il est permis à un homologue de 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" liant l'identité authentifiée par IKEv2 et les adresses permises pour les sélecteurs de trafic.

Par exemple, la PAD pourrait être configurée de façon que l'identité authentifiée "sgw23.exemple.com" ait la permission de créer des SA filles pour 192.0.2.0/24, signifiant que cette passerelle de sécurité est un"représentant" valide pour ces adresses. IPsec d'hôte à hôte exige des entrées similaires, reliant, par exemple, "fooserver4.exemple.com" à 198.51.100.66/32, ce qui signifie que cette identité est un "possesseur" ou "représentant" valide de l'adresse en question.

Comme noté dans la [RFC4301], "Il est nécessaire d'imposer ces contraintes à la création de SA filles pour empêcher un homologue authentifié de se faire passer pour une identité associée à d'autres, homologues légitimes". Dans l'exemple donné ci-dessus, une configuration correcte de la PAD empêche sgw23 de créer des SA filles avec l'adresse 198.51.100.66, et empêche fooserver4 de créer des SA filles avec des adresses à partir de 192.0.2.0/24.

Il est important de noter que le simple envoi de paquets IKEv2 en utilisant une adresse particulière n'implique pas une permission de créer des SA filles avec cette adresse dans les sélecteurs de trafic. Par exemple, même si sgw23 serait capable de faire passer son adresse IP pour 198.51.100.66, il ne pourrait pas créer des SA filles correspondant au trafic de fooserver4.

La spécification IKEv2 ne spécifie pas exactement comment une allocation d'adresse IP 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 entrée de PAD temporaire reliant l'identité authentifiée de l'homologue et la nouvelle adresse interne 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 d'assurer que la PAD spécifie le "possesseur" correct de chaque adresse IP. Cela exigerait un mécanisme pour porter en toute sécurité les allocations d'adresse depuis le serveur DHCP, et de les reliier entre elles pour les identités authentifiées en utilisant IKEv2.

Du fait de cette limitation, certains fabricants ont appris à configurer leurs PAD à permettre à un homologue authentifié de créer des SA filles avec des sélecteurs de trafic contenant la même adresse que celle utilisée pour les paquets IKEv2. Dans les 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 tous sélecteurs de trafic. Ce n'est pas une configuration appropriée ou sûre dans la plupart des circonstances. Voir dans [H2HIPSEC] une discussion étendue sur cette question, et les limitations de IPsec d'hôte à hôte en général.

6. Considérations relatives à l'IANA

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

Un élément a été déconseillé dans le registre "Codages de certificat IKEv2" : "Clé RSA brute".

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. *(MàJ par* RFC8174)

[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. *(P.S. ; MàJ par* RFC8820)

[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 la* RFC2401)

[RFC4307] J. Schiller, "Algorithmes cryptographiques à utiliser avec la version 2 de l'échange de clés sur Internet (IKEv2)", décembre 2005. *(P.S. ; rendue obsolète par* 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. and M. Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory, V.IT-22 n. 6, juin 1977.

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

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

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

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

[EXCHANGE] Perlman, R. and 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, U.S. Department of Commerce, "Secure Hash Standard (SHS)", FIPS 180-4, mars 2012, <http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>.

[H2HIPSEC] Aura, T., Roe, M., and 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, NIST Special Publication 800-38A, édition 2001, décembre 2001.

[REUSE] Menezes, A. and 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 RFC*4306)

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

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

[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. *(Information)*

[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) ; *remplacée par* RFC7296 *(STD79))*

[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 tout le protocole IKE 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 distante ;

2. De simplifier IKE en remplaçant les huit différents échanges initiaux 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 entier) voir [EXCHANGE] ;

3. De supprimer 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 2 allers-retours (4 messages), et en permettant la capacité de porter l'établissement d'une SA fille sur cet échange;

5. De remplacer la syntaxe de chiffrement pour protéger les messages IKE eux-mêmes avec une syntaxe fondée étroitement sur ESP pour simplifier la mise en œuvre et l'analyse de la sécurité ;

6. De réduire le nombre d'états d'erreur possibles en rendant le protocole fiable (tous les messages sont acquittés) et séquencés. Cela permet de raccourcir les échanges CREATE_CHILD_SA de 3 messages à 2 ;

7. D'augmenter la robustesse en permettant à celui qui répond de ne pas faire de traitement significatif jusqu'à ce qu'il reçoive un message prouvant que l'initiateur peut recevoir des messages à l'adresse IP qu'il revendique ;

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

9. De spécifier des sélecteurs de trafic dans leurs propres types de charges utiles plutôt que de surcharger les charges utiles ID, et de rendre plus souples 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 rendre plus facile de futures révisions sans casser la rétro compatibilité ;

11. De simplifier et préciser comment l'état partagé est maintenu en présence de défaillances du réseau et d'attaques de DoS ;

12. De maintenir 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 le minimum d'efforts.

Appendice B. Groupes Diffie-Hellman

Deux groupes Diffie-Hellman sont définis ici pour être utilisés dans IKE. Ces groupes ont été générés par Richard Schroeppel à 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 il est donné ici pour des raisons historiques.

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

B.1 Groupe 1 -t MODP de 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 80DC1CD129024E08 8A67CC74 020BBEA6 3B139B22
514A0879 8E3404DDEF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245E485B576 625E7EC6
F44C42E9 A63A3620 FFFFFFFF FFFFFFFF

Le générateur est 2.

B.2 Groupe 2 – MODP de 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 80DC1CD129024E08 8A67CC74 020BBEA6 3B139B22
514A0879 8E3404DDEF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245E485B576 625E7EC6
F44C42E9 A637ED6B 0BFF5CB6 F406B7EDEE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651
ECE65381FFFFFFFF FFFFFFFF

Le générateur est 2.

Appendice C. Échanges et charges utiles

Cet appendice contient un bref résumé des échanges IKEv2, et de quelles charges utiles peuvent apparaître dans quels messages. Cet appendice est purement informatif ; si il est en désaccord avec le corps du présent document, le texte du corps est considéré correct.

Les charges utiles Identifiant de fabricant (V) peuvent être incluses en tout endroit de tout message. La séquence présentée ici montre quels sont les endroits les plus logiques 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, [N(NAT_DETECTION_SOURCE_IP),
(pas de mouchard) N(NAT_DETECTION_DESTINATION_IP),]
[[N(HTTP_CERT_LOOKUP_SUPPORTED),] CERTREQ+,] [V+][N+]

réponse avec mouchard <-- N(COOKIE), [V+][N+]

groupe Diffie-Hellman différent voulu <-- 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 dans 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 à N fois | \ <-- EAP

Dernière demande --> AUTH

Dernière réponse <-- 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 la clé 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)

groupe Diffie-Hellman différent voulu <-- N(INVALID_KE_PAYLOAD), [V+][N+]

C.5 Échange CREATE_CHILD_SA pour le changement de clé de la 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 beaucoup aidé en contribuant par des idées et du texte au présent document, ainsi qu'en revoyant les précisions suggérées par d'autres.

Les remerciements pour le document IKEv2 étaient :
Le présent document est un effort collaboratif du groupe de travail IPsec entier. Si il n'y avait pas de limite au nombre des auteurs qui peuvent apparaître sur une RFC, les personnes suivantes, par ordre alphabétique, auraient été mentionnées : 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 peronnes 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 a préciser la conception de la négociation de sélecteur de trafic.

Adresse des auteurs

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

Paul Hoffman
VPN Consortium
127 Segre Place
Santa Cruz, CA 95060
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

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

Pasi Eronen
Independent
mél : pe@iki.fi