IPsec: Présentation et Configuration

version 1.7 - 22 avril 2003
Olivier Courtay

Table des matières

Introduction

Le but de ce document est de décrire IPsec, protocole de sécurisation des communications informatiques et d'expliquer les raisons qui ont poussé à la création de ce protocole.
Les principes d'IPsec et le vocabulaire nécessaire à la compréhension seront abordés ainsi que la description des utilisations possibles d'IPsec.
Seuls les éléments indispensables sont décrits, ce document n'est donc pas un document de référence (voir la bibliographie pour plus d'informations).
L'utilisation d'IPsec sera aussi décrites via la configuration et l'utilisation des outils nécessaires au déploiement d'IPsec.
Plusieurs cas d'utilisation seront couverts qu'ils soient manuels ou automatiques.
Seule la configuration pour IPv6 sera abordée mais la partie théorique est indépendante de la version du protocole IP.

Les remarques sont les bienvenues et peuvent être envoyées à l'auteur : olivier.courtay@enst-bretagne.fr.

1ère Partie - Présentation d'IPsec

1.1 Les faiblesses d'IP

Originellement IP est le principal protocole d'interconnexion des machines informatiques.
Les spécifications d'IP étaient axées essentiellement sur la robustesse (routage dynamique, par exemple) et non sur la sécurité (ni la fiabilité).
Ainsi actuellement le protocole IP est sujet à de nombreux problèmes de sécurité dont nous décrirons quelques exemples.
IP étant utilisé aussi bien pour les communications par l'Internet qu'au sein des réseaux d'entreprises, ces problèmes peuvent avoir de larges répercussions.

1.1.1 L'écoute des paquets

Sur IP, les données transportées sont contenues en clair dans les paquets, il est donc possible si on peut accéder aux paquets IP qu'échangent deux machines de connaître toutes les informations échangées entre ces deux machines.

Il existe des outils pour accéder aux paquets en cours de transmission ("sniffer") comme, par exemple :
- Ethereal qui permet de visualiser les paquets et leur contenu. Il est capable de réassembler le message TCP d'origine à partir des paquets IP.
- Dsniff qui peut récuperer des mots de passe sur le réseau, est capable de clore n'importe quelle connexion TCP. Dsniff implémente aussi une attaque dit "de l'homme au milieu" (man-in-the-middle) sur SSH qui permet notamment de modifier les paquets lors du transit sans que les deux machines communicantes puissent le savoir.

Lorsque deux ordinateurs de deux réseaux différents communiquent les paquets peuvent par exemple transiter successivement sur le réseau de l'entreprise puis sur celui du fournisseur d'accès Internet (FAI) puis sur la dorsale de l'Internet (backbone) puis sur le FAI de l'autre réseau puis sur le réseaux de la machine destinatrice.
Chaque routeur, ou chaque brin intermédaire, sont autant de points où il est possible, pour quelqu'un mal-intentionné, d'écouter les communications.

1.1.2 L'usurpation d'adresse

Un autre problème d'IP est le contrôle des adresses sources. Les routeurs ne vérifient pas l'adresse source des paquets qui transitent. Une machine pirate peut donc émettre des paquets IP ayant comme source l'adresse d'une autre machine, le paquet arrivera bien à destination et le destinataire ne pourra pas identifier la véritable source.
Dans le cadre d'UDP ou d'ICMP cela peut poser des problèmes importants car des notifications peuvent être faites (ICMP Redirect par exemple).
Beaucoup des causes de déni de service (DoS) sont dues à ce problème d'usurpation d'adresse, une ou des personnes malintentionnées bombardent une cible avec des paquets dont l'adresse source est fausse. La machine cible ne peut pas déceler qui attaque et donc remédier à l'attaque.

1.2 Nouveaux besoins

Aux vues de ces faiblesses, on peut définir de nouveaux besoins auxquels doit répondre IP :

Confidentialité :
La capture des paquets ne doit pas permettre de savoir quelles sont les informations échangées.
Seules les machines réceptrices et émettrices doivent pouvoir accéder à l'information.

Authentification:
Le récepteur doit être capable de verifier si les données reçues proviennent bien de l'émetteur supposé.

Intégrité :
Le récepteur doit être capable de verifier si les données n'ont pas été modifiées lors de la transmission.

Il y a aussi des propriétés moins évidentes à saisir mais néanmoins importantes :
Protection contre le rejeu :
Une personne qui intercepte un message d'une communication sécurisée entre deux machines ne pourra pas retransmettre ce message sans que cela soit détecté.

1.3 Base cryptographique

Les éléments que l'on vient de décrire (Confidentialité, Authentification, Intégrité et Protection contre le rejeu) n'ont pas fait leur apparition dans le monde Internet. Ces problèmes ont été étudiés auparavant dans d'autres secteurs (transmission militaire par exemple), la cryptographie possède une longue histoire.

Nous allons présenter dans ce paragraphe les principes de base des solutions cryptographiques existantes qui vont nous être utiles dans IPsec.

La cryptographie couvre, entre autres, les moyens de transmettre un message de manière confidentielle. Pour cela le message va subir une opération de **chiffrement** qui va créer un **message chiffré** , ce nouveau message va respecter la condition de confidentialité que l'on recherche et c'est ce message chiffré qui va être envoyé au destinataire.
Ce message n'aurait pas d'utilité si le destinataire était incapable de retrouver facilement le message d'origine. L'opération qui consiste à retrouver le message original est appellée **déchiffrement** .

Ces deux opérations, chiffrement et déchiffrement, s'appuient chacune sur un algorithme et sur une clé.
L'algorithme est en général connu de tous, la confidentialité du message ne vient pas de la méthode de chiffrement en elle-même, mais de l'ajout d'une donnée secrête qui est la clé et qui masque les données du message.
En effet, l'algorithme prend deux objets en entrée : le message et une clé, et il produit en sortie un message chiffré.
Sans connaissance de la clé de déchiffrement, l'obtention du message original est considérée comme extrèmement difficile.
L'information du message original n'est pas accessible en pratique depuis le message chiffré, bien qu'elle y soit présente.

Il existe deux grandes classes d'algorithmes (utilisant deux types de clés), le système de chiffrement à clés asymétriques et le système de chiffrement à clés symétriques.

1.3.1 Le système de chiffrement à clés symétriques

Ces systèmes sont appelés symétriques car la même clé sert à chiffrer et à déchiffrer. Il faut donc que l'expéditeur et le destinataire, et eux seuls, possèdent cette clé pour pouvoir s'échanger des messages de façon confidentielle.

Il existe plusieurs algorithmes de ce type couramment utilisés dans le monde :

DES (Data Encryption Standard)

C'était le standard du gouvernement américain depuis 1977. Il chiffre par blocs de 64 bits avec une clé d'une longueur de 56 bits . On parle de chiffrement par bloc quand l'algorithme prend en entrée un message de longueur constante (si le message est plus long, on le découpe en plusieurs blocs de la longueur voulue).
Aujourd'hui cet algorithme est considéré comme trop faible, car une recherche systématique par essai de clés permet de déchiffrer le message en un temps raisonnable suite à l'augmentation de la puissance des ordinateurs.

Le "Triple DES" a été introduit pour renforcer la sécurité du DES. Il consiste, pour simplifier, à chiffrer trois fois le message par DES (par deux ou trois clés de 56 bits).

Le standard (américain) est devenu en 2001 l'algorithme AES (Advanced Encryption Standard), créé par des chercheurs belges de l'université catholique de Louvain : Vincent Rijmen et Joan Daemen sous le nom de "Rijndael".
L'AES est reconnu comme très sûr, il bénificie aussi de l'atout de pouvoir être utilisé avec des clés de différentes tailles (128, 192 et 256 bits) et d'être portable sur beaucoup d'architectures.

Il existe bien d'autres algorithmes à clé symétrique : IDEA (International Data Encryption Algorithm), blowfish, RC4, RC5, cast128, tiger, twofish ...

1.3.2 Le système de chiffrement à clés asymétriques

Dans les systèmes à clés asymétriques, les clés de chiffrement et de déchiffrement sont distinctes cependant elles sont liées par une relation mathématique mais ne peuvent pas être déduites l'une de l'autre (dans la pratique).
La clé de chiffrement est rendue publique (appelée **clé publique**) tandis que celle de déchiffrement doit être gardée totalement secrète (et appelée **clé privée**).

Ainsi tout le monde peut chiffrer un message avec la clé publique (ou de chiffrement) mais seul le détenteur de la clé privée (ou de déchiffrement) peut déchiffrer le message chiffré.

Un autre intérêt de ce système, c'est qu'il permet la vérification d'identité. En effet la clé privée peut aussi servir de clé de chiffrement pour l'algorithme considéré et tout le monde pourra déchiffrer le message chiffré avec la clé publique.
Une personne A envoie un message à B en clair, le destinataire B pour prouver son identité chiffre le message avec sa clé privée, et renvoie le message généré à la personne A. A peut alors vérifier que le message a bien été chiffré avec la clé privée de B en déchiffrant le message avec la clé publique de B. En effet le message ainsi déchiffré doit être identique au premier message envoyé par A. Ce procédé est connu sous le nom de signature numérique.

Les algorithmes à clé publique sont basés sur des problèmes mathématiques du type factorisation de grand nombres. Connaisant deux nombres, il est facile d'en connaître le produit, mais connaissant un nombre, il n'est pas aisé de le factoriser.

1.3.3 L'algorithme Diffie-Hellman

Le problème lié aux algorithmes à clés asymétriques, c'est qu'ils sont beaucoup moins rapides que les algorithmes à clé symétrique.
C'est pourquoi, en général, pour établir une communication entre deux entités, on commence par utiliser des algorithmes à clés asymétriques, pour s'échanger un secret partagé.
Ce secret partagé pouvant être utilisé comme clé symétrique, on peut l'utiliser pour communiquer. Les entités chiffrent alors les messages avec la clé secrète partagée.

Ce sont Diffie et Hellman qui en 1976 explicitent une méthode pour effectuer cet échange :

Alice et Bob veulent partager un secret mais toutes les communications qu'ils échangent peuvent être écoutées. Ils appliquent donc l'algorithme de Diffie-Hellman :

1. Alice et Bob choisissent un nombre **n** tel que **n** et (**n**-1)/2 soient premiers et un nombre **g** tel que pgcd(g,n) = 1.
2. Alice choisit au hasard un nombre **a** (qui sera sa clé privée) et calcule **A** = **g**^**a** mod(**n**).
3. Bob choisit un nombre **b** et calcule **B**= **g**^b mod(**n**).
4. Bob et Alice s'échangent **A** et **B**.
5. Alice calcule **KA** = **B**^**a** mod(**n**).
6. Bob calcule **KB** =**A**^**b** mod(**n**)

Comme **KA** = **B**^**a** mod(**n**) = (**g**^**b** mod(n))^**a** mod(**n**) = **g**^**b**^**a** mod(**n**) = **g**^**a**^**b** mod(**n**) = **A**^**b** mod(**n**) = **KB**

Ainsi Alice et Bob partagent un secret commun.

1.4 Le protocole IPsec

Pour implémenter les fonctionnalités de confidentialité, d'authentification et d'intégrité, le protocole IPsec ("IP security protocol") a été spécifié.
L'IETF (*Internet Engineering Task Force*) travaille depuis 1992 à la standardisation d'IPsec et des RFC ont été produites en ce sens depuis 1995.
Il a été décidé que IPsec serait obligatoire dans IPv6 et facultatif dans IPv4. Les mécanismes sont les mêmes pour les deux versions d'IP.

IPsec va s'appuyer sur les techniques de cryptoraghie pour assurer ces nouvelles fonctionnalités.

IPsec regroupe trois mécanismes indépendants (au niveau réseau) :
- AH (Authentication Header) qui sert à valider l'intégrité des messages et à prouver l'identité de l'expéditeur, ainsi que d'eviter les rejeux.
- ESP (Encapsulation Security Payload) qui sert à assurer la confidentialité des messages.
- IPcomp (IP compression) qui compresse les données qui transitent.

Il existe d'autres solutions pour assurer ces fonctions mais elles sont particulières à chaque protocole ou application.
On peut citer SSH qui permet à un utilisateur distant d'avoir un interpréteur de commande à distance securisé (chiffrement et authentification).
Il y a aussi SSL (Secure Socket Layer) appelé aussi TLS qui offre la possibilité d'ouvrir des sockets TCP sécurisées, mais les applications doivent alors explicitement faire appel à cette bibliothèque de programmation.

Un des grands apports d'IPsec est de sécuriser toutes les applications ou communications au-dessus d'IP de façon transparente.

1.4.1 La "Security Policy Database" (SPD)

La "Security Policy Database" (SPD) regroupe l'ensemble des comportements que le système doit présenter par rapport à des communications externes.
Cette base sert donc à décider, lors de l'ouverture d'une communication, si elle doit ou non utiliser IPsec.
Elle est en général configurée par l'administrateur de la machine.
Dans cette base, les communications qui peuvent ou qui doivent utiliser le protocole IPsec sont définies par les informations disponibles dans l'en-tête IP et dans le niveau transport (TCP, UDP, ICMP) des paquets de la communication.
En général, les adresses source et destination, le protocole et les ports source et destination (pour TCP ou UDP) suffisent pour définir une politique de sécurité ("Security Policy").
Il est à noter qu'une "Security Policy" est unidirectionnelle, si la communication veut s'établir dans les deux sens il faudra deux "Security Policy" ou entrée dans la SPD.

Il faut ensuite associer à ces communications les fonctionnalités IPsec que l'on veut leur appliquer.
Si on désire de la confidentialité, on va spécifier qu'il faut utiliser la transformation ESP.
On peut aussi spécifier que l'on désire de l'authentification, il s'agit alors de la transformation AH.
Il existe aussi la transformation IPcomp qui compresse les paquets.

1.4.2 La "Security Association Database" (SADB)

Une "Security Policy" peut par exemple demander à ce que les communications entre une adresse source et une adresse destination soient chiffrées (via ESP).
Mais plusieurs communications peuvent avoir lieu en même temps, il faut donc maintenir un contexte pour chaque connexion IPsec. Ce contexte est appellé "Security Association" (SA).

On a vu que IPsec proposait plusieurs fonctionnalités appliquées ou non, et que plusieurs algorithmes pouvaient être utilisés.
Une "Security Association" regroupe l'ensemble de ces informations :
- le numéro de la connexion ou "Security Parameter Index" (SPI)
- l'adresse de destination des paquets
- le mécanisme IPsec utilisé (ESP ou AH)
- les algorithmes utilisés pour ces mécanismes.

Si AH et ESP sont utilisés conjoîtement, on aura plusieurs SA liées à cette connexion (une pour AH, une pour ESP), et de même si la connexion est bidirectionelle.

1.4.3 Les modes Transports et Tunnels

Il existe deux modes dans IPsec, le mode Transport et le mode Tunnel.
Ces deux modes diffèrent par la méthode de construction des paquets IP des messages.

Dans le mode Transport, seules les données contenues dans la couche Transport sont protégées par IPsec, l'en-tête IP reste inchangé.

exemple :

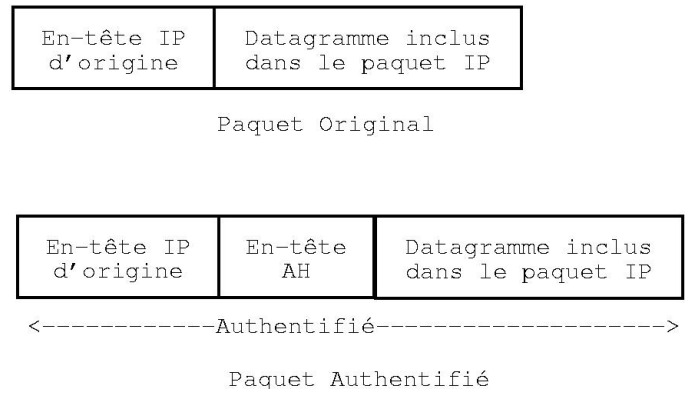

fig 1. en-tête AH - mode Transport.

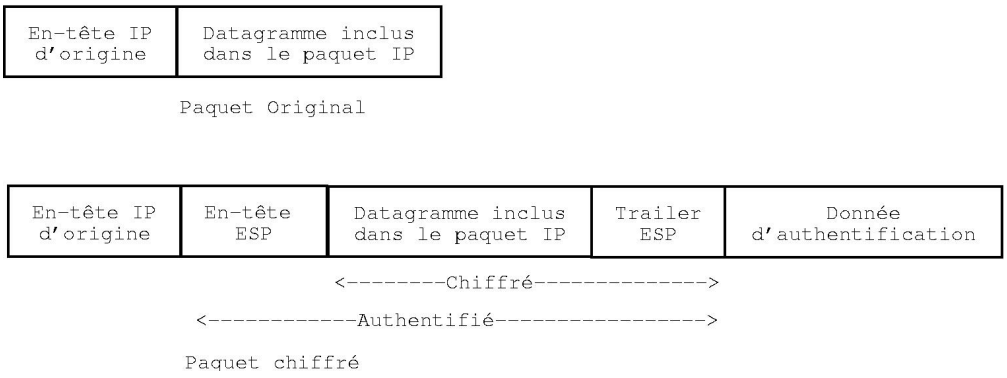

fig 2. en-tête ESP - mode Transport.

Dans le mode Tunnel, tout le paquet IP est protégé. Pour cela , il est consideré comme un simple message, et un nouvel en-tête IP est créé.

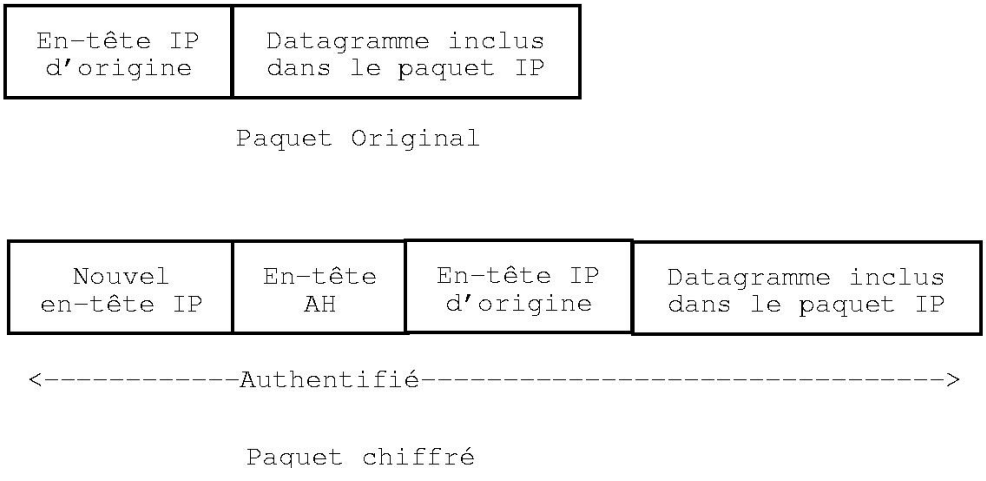

fig 3. en-tête AH - mode Tunnel.

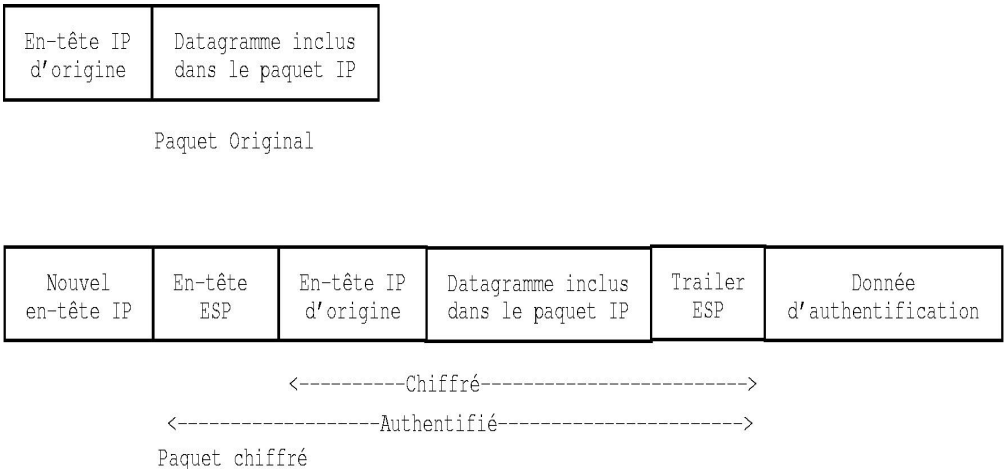

fig 4. en-tête ESP - mode Tunnel.

Le mode Tunnel possède comme grand intérêt de pouvoir créer des tunnels sécurisés. Deux machines ou passerelles de deux réseaux voulant sécuriser leurs communications qui passent par une zone non protégée, vont créer un tunnel IPsec entre ces deux passerelles. Toute communication d'une machine d'un réseau vers l'autre sera encapsulée dans un nouvel en-tête IP. Une personne écoutant la communication ne verrait que des paquets allant d'une passerelle à l'autre sans pouvoir comprendre le contenu de ces paquets. Ce mode correspond à un réseau privé virtuel ou VPN (Virtual Private Network).

Voici un schéma des différentes possibilités :

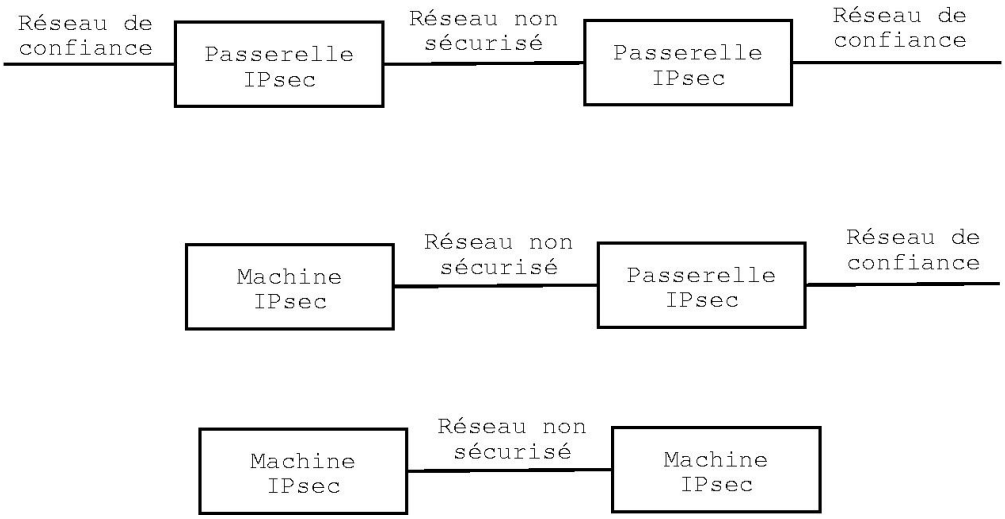

fig 5. modes d'utilisations d'IPsec.

Le premier correspond au mode VPN, où un tunnel IPsec est mis en place entre deux passerelles pour sécuriser l'ensemble des communications entre ces deux réseaux.
Le deuxième correspond à une utilisation de poste nomade, où un ordinateur portable affilié au réseau, mais distant veut se connecter sur le réseau (appellé aussi Road-Warrior).
Le troisième est l'utilisation classique du mode Transport, où deux machines veulent communiquer ensemble de façon sécurisée.

1.4.4 Gestion des clés (IKE)

On a décrit précédemment une Security Association, elle peut être configurée manuellement par un administrateur dans la base SADB, mais en général elle est positionnée de façon automatique. En effet dans le cadre d'un VPN, le tunnel sera stable dans la durée, l'administrateur peut positionner une fois pour toute une SA. Mais dans le cadre d'une utilisation d'un nomade, par exemple, il faut que cette tâche soit automatisée.

Il existe donc un démon appelé démon IKE (Internet Key Exchange) qui se charge de la négociation et de la mise en place des SA.

IKE se décompose en plusieurs protocoles :
ISAKMP (Internet Security Association and Key Management Protocol) : il négocie les "Security Associations" (établissement, supression, paramètres...).
ISAKMP n'est pas spécifique à IPsec, il existe donc un DOI (Domain Of Interpretation) qui permet d'utiliser ce qui est produit par ISAKMP dans IPsec.
Oakley et Scheme sont des protocoles d'échange de clé et d'authentifications.

Le principe est que si un paquet est envoyé par la machine, le système vérifie dans la SPD quelle est la politique IPsec à lui associer. Si IPsec doit ou peut se charger de ce paquet, le système va rechercher une SA qui correspond à cette SP et à ce paquet. Si le système trouve une SA correpondant, le paquet se voit appliquer les régles définies dans la SA, sinon le système va appeler le démon IKE pour créer si possible une SA.

Il existe une exception à ce principe : les paquets IKE ne sont jamais soumis à IPsec par un ajout d'une régle précisant de ne pas utiliser IPsec dans ce cas.
Cela permet de casser le problème de l'oeuf ou de la poule (pour mettre en place IPsec, on ne peut pas utiliser IPsec).
A noter qu'IKE effectue ses échanges au-dessus du niveau transport (UDP, port 500, en général). Cela permet bien de découpler la négociation IPsec des fonctionnalités d'IPsec.

Voici le schéma global d'IPsec :

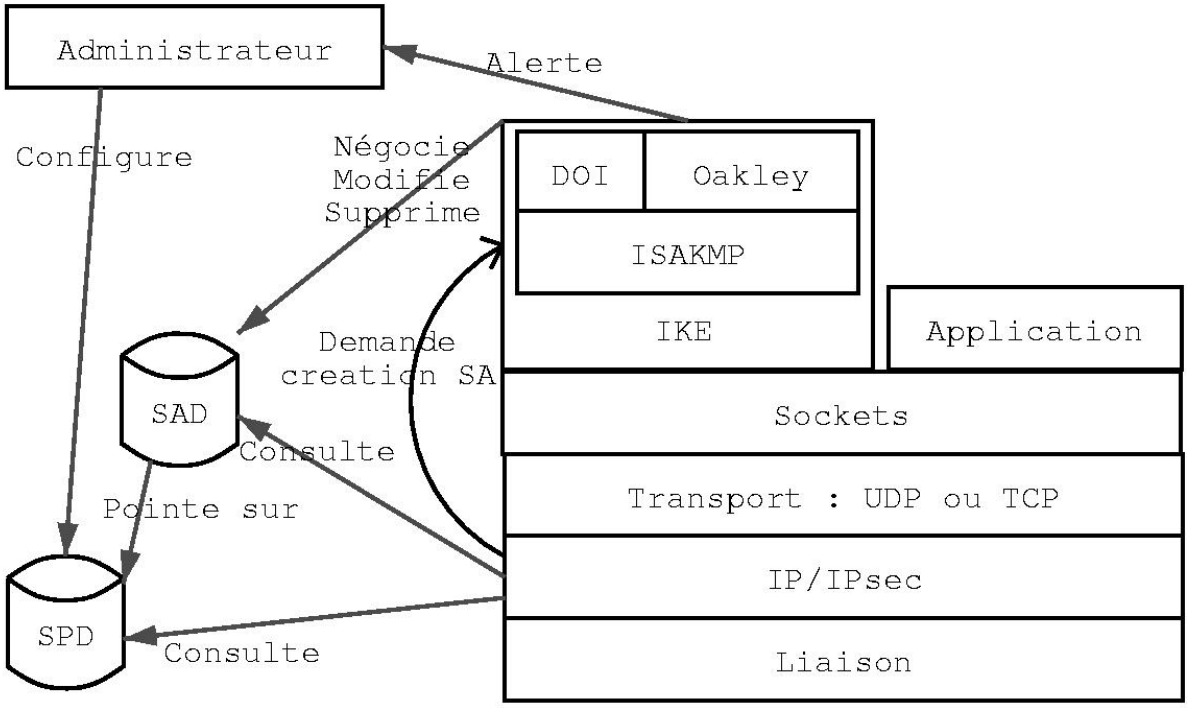

fig 6. Schéma global d'IPsec

Le protocole ISAKMP a été conçu pour être souple et pourrait être utilisé par d'autres applications

qu'IPsec.
La structure des paquets ISAKMP est constituée de chaînage de blocs.

Il existe 13 types de blocs possibles :

Nom	Sigle
Security Association	SA
Proposal	P
Transform	T
Key Exchange	KE
Identification	ID
Certificate	CERT
Certificate Request	CR
Hash	HASH
Signature	SIG
Nonce	NONCE
Notification	N
Delete	D
Vendor ID	VID

tab 1. bloc de ISAKMP

Le bloc **Security Association** indique le contexte et la situation de l'échange.
Il comporte notamment le DOI qui précise si on est dans un échange de type ISAKMP (première phase) ou dans l'étape plus spécifique d'IPsec (deuxième phase).
Ce bloc est composé d'un ou plusieurs blocs Proposal.

Le bloc **Proposal** contient des propositions sur les mécanismes de sécurité à fournir tel que les ESP avec les spi correspondants.
Ce bloc est composé d'un ou plusieurs blocs Transform.

Le bloc **Transform** contient les algorithmes et les attributs associés pour les mécanismes de sécurité contenus dans les blocs Proposal.

Le bloc **Key Exchange** contient des données nécessaires à la génération de clés.
Ces données dépendent du contexte et du protocole d'échange de clés.

Le bloc **Identification** contient des données nécessaires à l'identification des entités présentes.
Ces données dépendent du contexte et du procédé d'identification.
Plusieurs types sont possibles : adresses IP(v4 ou v6), voire plage d'adresses IP, mais aussi FQDN (Fully Qualified Domain Name) (exemple : xbox.ipv6.rennes.enst-bretagne.fr) ou USER FQDN (exemple : toto@toto.org).
Son interprétation est différente en phase 1 ou en phase 2. En phase 1 ce bloc sert à l'authentification, tandis que en phase 2 cela sert de traffic selector.

Le bloc **Certificate** contient un certificat ainsi que le type de certficat contenu.
Ce certificat peut être de différents types: PKCS#7, DNS signed KEY, X509 (signature key exchange ou attribute) ou SPKI certificate.

Le bloc **Certificate Request** permet de réclamer un certificat à l'entité tiers (en précisant le type de certificat voulu).

Le bloc **Hash** contient le résultat d'une fonction de hachage (cf bloc Transform) sur le message ou sur un attribut.

Le bloc **Signature** contient le résultat d'un mécanisme de signature sur le message ou sur un attribut. Cela est souvent utilisé pour faire de l'authentification.

Le bloc **Nonce** contient des valeurs aléatoires choisies par une des entités.

Le bloc **Notification** contient des messages d'erreur ou d'information. Son interprétation dépend du contexte.

Le bloc **Delete** permet à une entité de préciser à son homologue qu'elle vient de supprimer une SA.

Le bloc **Vendor ID** comme il est spécifié dans les RFC peut-être utilisé à des fins spécifiques à une implémentation.
Mais en général, il sert a negocier des extentions.

1.4.5 Les phases d'IKE

ISAKMP se décompose en deux phases:
-La première phase permet de vérifier l'identité des entités en présence. On décide des algorithmes de cryptographie utilisés pour les futures négociations ISAKMP.
Á la fin de cette phase, chaque entité doit disposer d'une clé de chiffrement, d'une clé d'authentification et d'un secret partagé qui servira de "graine" dans la phase suivante (on produit une clé en fonction de valeurs déja calculées).

- La deuxième phase permet de négocier les attributs plus spécifiques à IPsec (utilisation d'AH ou d'ESP par exemple), ces échanges sont chiffrés et authentifiés grâce aux éléments décidés lors de le première phase.

Il y a un interêt à ce découpage. La première phase fait appel à de la cryptographie asymétrique qui est lente, elle est de plus utilisée qu'une seule fois pour définir les paramètres qui vont permettre de sécuriser les échanges de phase 2.
La phase 2 est en revanche appelée plusieurs fois, en effet les clés qui servent à chiffrer deviennent vulnérables avec le temps ou quand on s'en sert beaucoup (un cryptanalyste a plus de matériel (ou messages) pour essayer de casser la clé). Cette phase est donc régulièrement réeffectuée pour changer certaines clés de sessions.

Il existe en effet trois grands types d'utilisation de clé :
- les clés de chiffrement de clés : elles sont souvent de très longue durée de vie et peuvent être contenues dans des certificats.
- les clés maîtresses : elles servent à générer d'autre clés, par exemple une pour l'authentification et une autre pour le chiffrement (le secret partagé à la fin de Diffie-Hellman rentre dans cette catégorie).
- les clés de session : ce sont elles qui sont utilisés pour chiffrer par exemple, elles ont en général une durée de vie assez faible (ça peut être quelques minutes).

Il existe plusieurs schémas possibles pour la phase 1. On décide de celui que l'on veut utiliser suivant le degré de rapidité ou de sécurité que l'on veut atteindre. Il faut aussi que le démon IKE de l'autre système accepte le schéma que l'on propose. En général l'administrateur précise un ordre de préférence.

On va utiliser les notations standards définies dans la RFC 2409 :

Notation	Signification
HDR	Header ISAKMPD

HDR*	tous les payloads sont chiffrés
SA	Payload de négociation, contient des propositions pour l'initiateur
Nonce	Payload Nonce
KE	Payload d'échange de clé
IDii	Payload d'identité Phase 1 initiateur
IDci	Payload d'identité Phase 2 initiateur
IDir	Payload d'identité Phase 1 répondeur
IDcr	Payload d'identité Phase 2 répondeur
Auth	Mécanisme d'authentification
HASH	Payload du Hash

tab 2. notation standard des échanges

1.4.5.1 Première Phase

Deux modes sont possibles pour la première phase, le mode "main"et mode "aggressive".

Le mode Main

Dans le premier échange, les deux entités se mettent d'accord sur les paramètres cryptographiques à utiliser.
Le deuxième échange est un échange Diffie-Hellman.
Après le deuxième échange, les deux entités ont donc un secret partagé qui permet de chiffrer la suite des échanges.
Dans le troisième échange, l'identité des entités est vérifiée (elles ne sont accessibles que par elles).

Initiator Responder

HDR, SA
---------->

HDR, SA
<----------

HDR, KE, Nonce
---------->

HDR, KE, Nonce
<----------

HDR*, IDii, Auth
---------->

HDR*, IDir, Auth
<----------

fig 7. le Main Mode

Le mode Aggressive

Le premier message (de l'Initiator) envoie les propositions des mécanismes cryptographiques mais aussi le matériel pour l'échange de Diffie-Hellman.
Le deuxième échange (du Responder) finalise le choix des mécanismes de sécurité, termine l'échange Diffie-Hellman et envoie l'information pour prouver son identité.
Le dernier message permet à l'Initator de prouver son identité.
L'intérêt de ce mode est d'être très rapide (3 messages au lieu de 6), mais assure un niveau de sécurité plus faible (notamment les identités ne sont pas protégés).

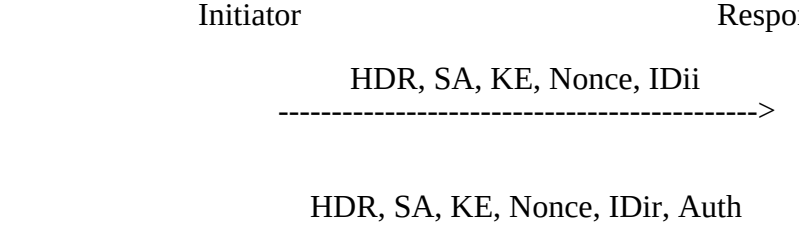

fig 8. l'Agressive Mode

1.4.5.2 Deuxième Phase

Tous les échanges sont maintenant chiffrés et authentifiés par les éléments négociés lors de la phase 1.
Cette phase est aussi appelée Quick Mode.
Les attributs spécifiques à IPsec sont décidés dans cette phase.
De plus il est possible d'effectuer un nouvel échange Diffie-Hellman pour obtenir la propriété de **Perfect Forward Secrecy** (PFS).
Cela permet que si une clé est cassée on ne peut en déduire les suivantes ou les précedentes et ainsi seul un petit bout de la communication a pu être déchiffrée par un tiers.

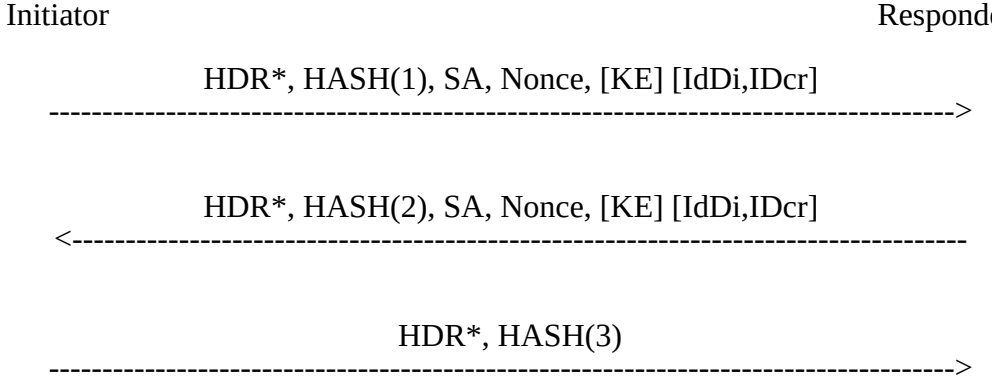

fig 9. Le Quick Mode

1.5 Cas concret d'échange IKE entre deux machines

Deux machines essayent de mettre en place IPsec entre elles.
Il s'agit des machines d'adresse IPv6 2001:660:284:100:2c0:4fff:fea9:8128 (machine A) et 2001:660:284:100:200:c0ff:fe4a:d8c5 (machine B).

Ces traces sont tirées d'une option du démon Isakmpd qui enregistre les paquets des échanges après les avoir déchiffrés si nécessaire.
On peut noter que l'autre démon utilisé est racoon, il y a bien une certaine intéropérabilité entre les divers démons existants (mais pas totale).

Ces traces ne sont que des illustrations des schémas d'échange ci-dessus.

Début de la première phase.

A ------ HDR + SA ----------------> B

```
14:24:26.639104 2001:660:284:100:2c0:4fff:fea9:8128.500 >
2001:660:284:100:200:c0ff:fe4a:d8c5.500: isakmp: phase 1 I ident:
    (sa: doi=ipsec situation=identity
        (p: #1 protoid=isakmp transform=1
            (t: #0 id=ike (type=enc value=3des)(type=hash value=md5)(type=auth
value=preshared)(type=group desc value=modp1024)(type=lifetype value=sec)(type=lifedu
ration value=0e10))))
```

On remarque que A propose de chiffrer avec 3DES, de hacher avec MD5 et que la méthode d'authentification est le secret partagé (preshared).

B ---------- HDR, SA ---------------> A

```
14:24:33.714317 2001:660:284:100:200:c0ff:fe4a:d8c5.500 >
2001:660:284:100:2c0:4fff:fea9:8128.500: isakmp: phase 1 R ident:
    (sa: doi=ipsec situation=identity
        (p: #1 protoid=isakmp transform=1
            (t: #0 id=ike (type=enc value=3des)(type=hash value=md5)(type=auth
value=preshared)(type=group desc value=modp1024)(type=lifetype value=sec)(type=lifedu
ration value=0e10))))
    (vid: len=16)
```

Même proposition du coté de B, il y a donc concordance.
Un bloc *vid* pour Vendor Id est aussi fourni.

A ---------- HDR, KE, Nonce ------> B

```
14:24:33.807258 2001:660:284:100:2c0:4fff:fea9:8128.500 >
2001:660:284:100:200:c0ff:fe4a:d8c5.500: isakmp: phase 1 I ident:
    (ke: key len=128)
    (nonce: n len=16)
```

Début de l'échange de Diffie-Hellman.

B ---------- HDR, KE, Nonce ------> A

```
14:24:33.917484 2001:660:284:100:200:c0ff:fe4a:d8c5.500 >
2001:660:284:100:2c0:4fff:fea9:8128.500: isakmp: phase 1 R ident:
    (ke: key len=128)
    (nonce: n len=16)
    (vid: len=16)
```

Continuation de Diffie-Hellman

A ---------HDR, ID; Hash ----------> B

```
14:24:34.042996 2001:660:284:100:2c0:4fff:fea9:8128.500 >
2001:660:284:100:200:c0ff:fe4a:d8c5.500: isakmp: phase 1 I ident:
    (id: idtype=IPv6 protoid=0 port=0 len=16 2001:660:284:100:2c0:4fff:fea9:8128)
    (hash: len=16)
    (n: doi=ipsec proto=isakmp type=INITIAL-CONTACT spi=92ae255eba1e5c1672cc477ad2370f71)
```

A décline son identité sous forme d'adresse IPv6.
Le INITIAL-CONTACT est un signal du démon qui fait savoir que pour lui c'est la première fois que le communication s'établit et qu'il faut donc oublier les anciennes SA.
Le bloc *Hash* sert à l'authentification ici (idem dans la suite de l'échange).

B ---------HDR, ID; Hash ----------> A

```
14:24:34.177205 2001:660:284:100:200:c0ff:fe4a:d8c5.500 >
2001:660:284:100:2c0:4fff:fea9:8128.500: isakmp: phase 1 R ident:
    (id: idtype=IPv6 protoid=udp port=500 len=16 2001:660:284:100:200:c0ff:fe4a:d8c5)
    (hash: len=16)
```

De même B décline son identité.

Début de la deuxième phase.

A ----------HDR, SA,KE,Nonce, ID, Hash --> B

```
14:24:34.312872 2001:660:284:100:2c0:4fff:fea9:8128.500 >
2001:660:284:100:200:c0ff:fe4a:d8c5.500: isakmp: phase 2/others I oakley-quick:
    (hash: len=16)
    (sa: doi=ipsec situation=identity
        (p: #1 protoid=ipsec-esp transform=1 spi=1ee4aa74
            (t: #1 id=3des (type=lifetype value=sec)(type=life value=04b0)(type=enc mode
value=transport)(type=auth value=hmac-md5)(type=group desc value=modp1024))
))
    (nonce: n len=16)
    (ke: key len=128)
    (id: idtype=IPv6 protoid=0 port=0 len=16 2001:660:284:100:2c0:4fff:fea9:8128)
    (id: idtype=IPv6 protoid=0 port=0 len=16 2001:660:284:100:200:c0ff:fe4a:d8c5)
```

On peut remarquer que dans le bloc Proposal le *protoid* est maintenant *ipsec-esp* au lieu de *isakmp* auparavant.
Un *spi* est proposé, ansi que l'algorithme de chiffrement (3DES), l'algorithme d'authentification utilisera hmac-md5 comme fonction de hachage.

Ici la réponse au initial contact:

```
14:24:34.341761 2001:660:284:100:200:c0ff:fe4a:d8c5.500 >
2001:660:284:100:2c0:4fff:fea9:8128.500: isakmp: phase 2/others R inf:
    (hash: len=16)
    (n: doi=ipsec proto=isakmp type=INITIAL-CONTACT spi=92ae255eba1e5c1672cc477ad2370f71
orig=([|isakmp]))
```

B ----------HDR, SA,KE,Nonce, ID, Hash --> A

```
14:24:34.598281 2001:660:284:100:200:c0ff:fe4a:d8c5.500 >
2001:660:284:100:2c0:4fff:fea9:8128.500: isakmp: phase 2/others R oakley-quick:
    (hash: len=16)
    (sa: doi=ipsec situation=identity
        (p: #1 protoid=ipsec-esp transform=1 spi=0d37f6e5
            (t: #1 id=3des (type=lifetype value=sec)(type=life value=04b0)(type=enc mode
value=transport)(type=auth value=hmac-md5)(type=group desc value=mod
p1024))))
    (nonce: n len=16)
    (ke: key len=128)
    (id: idtype=IPv6 protoid=0 port=0 len=16 2001:660:284:100:2c0:4fff:fea9:8128)
    (id: idtype=IPv6 protoid=0 port=0 len=16 2001:660:284:100:200:c0ff:fe4a:d8c5)
```

La réponse propose des caractéristiques identiques à celles proposées, les deux entités sont donc d'accord sur celles-ci.

A ---------- Hash ------------------------------> B

```
14:24:34.638155 2001:660:284:100:2c0:4fff:fea9:8128.500 >
2001:660:284:100:200:c0ff:fe4a:d8c5.500: isakmp: phase 2/others I oakley-quick:
    (hash: len=16)
```

2ème Partie - Configuration de Racoon

2.1 Introduction

Nous proposons ici, d'expliquer la mise en oeuvre pratique d'IPsec grâce au programme Racoon.
Trois méthodes sont etudiées :
- le mode du secret partagé (Pre Shared Key)
- le mode du certificat X509.
- le mode (appelé abusivement) DNSsec et X509, où le certificat X509 est recupéré via le DNS.

Nous nous limiterons au cas du mode Transport d'IPsec
Les tests ont été effectués avec un racoon 20021216 sur FreeBsd 4.5 et FreeBSD 4.6.2.

2.2 Architecture IPsec

Dans la partie théorique, nous avons vu la figure (6) représentant le schéma global d'IPsec.
Voici une nouvelle version axée sur son implémentation et non sur les protocoles comme dans le précédent schéma.

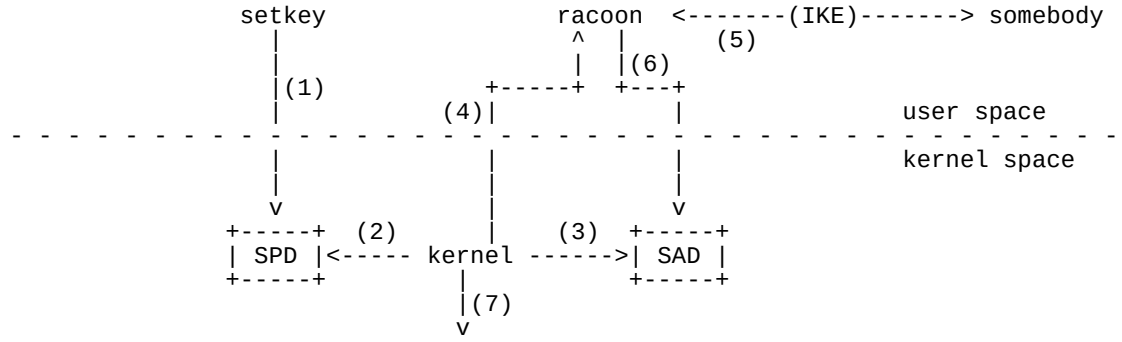

fig 10. Implémentation de IPsec

Le noyau comporte deux bases de données, la SPD(Security Policy Database) et la SADB (Security Association Database).
La SPD contient les règles (appelées SP pour Security Policy) qui permettent au noyau de décider à quel paquet doit être appliqué IPsec.
La SADB contient des SA (Security Association), les attributs qui permettent au noyau d'effectuer les fonctions d'IPsec (par exemple les clés de chiffrement).

Setkey est un outil qui permet à l'administrateur de poser des entrés dans la SPD et la SADB.
Racoon est un démon qui permet de négocier les attributs IPsec nécessaires à la création d'une paire de SA.

Le scénario classique est :
1) l'administrateur de la machine configure la SPD grâce à la commande setkey.
2) lorsque un paquet passe par la machine (depuis ou vers le réseau) le noyau vérifie s'il correspond ou non à une entrée de la SPD.
3) si IPsec est demandé pour ce paquet le noyau cherche les attributs IPsec correspondants dans la SAD.
4) si ces attributs n'existent pas alors le noyau demande au démon IKE (racoon) de les négocier avec la machine concernée.
5) les attributs sont négociés entre les deux démons IKE.
6) racoon les positionne dans une nouvelle entrée de la SADB et le paquet peut enfin être chiffré ou authentifié grâce à IPsec.

2.3 Configuration de test

Deux machines :
- FreeBSD 4.5 avec patch Kame
- FreeBSD 4.6.2 sans patch Kame

Les deux machines sont configurées en IPv6.

Le fichier /etc/ipsec.conf

Le premier fichier à configurer est le fichier /etc/ipsec.conf.
Il décrit les flux de données qui seront assujettis à IPsec.
Ce fichier va ensuite servir pour créer la SPD du noyau grâce à un appel à la commande setkey.
Ce fichier est commun aux trois méthodes étudiées ici.

exemple de fichier /etc/ipsec.conf

Pour une machine A ayant pour adresse 2001:660:305::1 et une machine B ayant pour adresse 2001:660:305::2.
Le ficher /etc/ipsec.conf pour la machine A peut-être:

```
flush;
spdflush;
spdadd  2001:660:305::2/128 2001:660:305::1/128  any -P in  ipsec esp/transport//use;
spdadd  2001:660:305::1/128 2001:660:305::2/128  any -P out ipsec esp/transport//use;
```

Les deux premières lignes suppriment la configuration courante.
La troisième ligne ajoute une entrée qui spécifie que tout paquet rentrant (flag *in*) de *2001:660:305::2* et allant vers *2001:660:305::1* devrait utiliser IPsec et ceci quel que soit le protocole utilisé (*any*).
La partie *esp/transport//use* précise le mode IPsec :
- *esp* : (Encapsulating Security Payload) permet de chiffrer les communications.
  la tranformation *ah* est possible aussi pour faire de l'authentification
- *transport* : il existe deux modes pour IPsec, *transport* ou *tunnel*
- *use* : utiliser IPsec si c'est possible, sinon on utilise le mode normal, (contrairement à *require*).

Pour mettre en place ces règles, il faut utiliser la commande "setkey".

```
#setkey -f /etc/ipsec.conf
```

Pour vérifier que la configuration a bien fonctionné, essayez ceci:

```
chronos# setkey -DP
```

vous devez avoir alors une sortie du type :

```
2001:660:305::1[any] 2001:660:305::2[any] any
        in ipsec
        esp/transport//use
        created: Oct 23 16:37:51 2002  lastused: Oct 24 10:37:26 2002
        lifetime: 0(s) validtime: 0(s)
        spid=8 seq=1 pid=39905
        refcnt=1
2001:660:305::2[any] 2001:660:305::1[any] any
        out ipsec
        esp/transport//use
        created: Oct 23 16:37:51 2002  lastused: Oct 24 10:37:26 2002
        lifetime: 0(s) validtime: 0(s)
        spid=7 seq=0 pid=39905
        refcnt=1
```

Pour résumer, pour connecter la machine A d'adresse IP_A à B d'adresse IP_B, il faut :
- créer le fichier /etc/ipsec.conf sur la machine A:

```
flush;
spdflush;
spdadd  IP_B/128 IP_A/128  any -P in  ipsec esp/transport//use;
spdadd  IP_A/128 IP_B/128  any -P out ipsec esp/transport//use;
```

- créer le fichier /etc/ipsec.conf sur la machine B:

```
flush;
spdflush;
spdadd  IP_A/128 IP_B/128  any -P in  ipsec esp/transport//use;
spdadd  IP_B/128 IP_A/128  any -P out ipsec esp/transport//use;
```

- lancer la commande *setkey -f /etc/ipsec.conf* sur les deux machines.

2.3.1 La méthode du secret partagé

Pour mettre en œuvre cette méthode, il faut que chaque machine possède un secret connu uniquement par elles.
Ce secret est généralement un mot. Ce mot doit être sans espace et s'il commence par 0x il sera intérpreté comme un nombre héxadécimal.

Ce secret est placé dans le fichier psk.txt.
Il doit obligatoirement appartenir à l'utilisateur de racoon (root) et n'être pas accessible aux autres utilisateurs (chmod 600 psk.txt).
Attention, par défaut la distribution de FreeBSD est mal configurée par rapport aux droits des fichiers.

Exemple:

```
# IPv4 address
10.160.94.3     mekmitasdigoat
# IPv4 addresse in hexadecimal notation
172.16.1.133    0x12345678
#IPv6 address
3ffe:501:410:ffff:200:86ff:fe05:80fa    mekmitasdigoat
# FQDN
gaia.ipv6.rennes.enst-bretagne.fr       foobar
# USER_FQDN
foo@kame.net    mekmitasdigoat
```

Le fichier associe des identifiants (nom FQDN ou adresse IP) à un secret. Ces identifiants permettent l'authentification lors de la connexion IPsec.
Par exemple, si la machine possédant ce fichier de configuration reçoit de la machine son adresse IP comme identifiant (*3ffe:501:410:ffff:200:86ff:fe05:80fa*), elle utilisera le secret partagé *mekmitasdigoat* pour établir la communication.

Le ficher racoon.conf :

```
# définition du répertoire où se situe les fichiers de configuration
path include "/usr/local/v6/etc" ;


# fichier où sont stockés les secrets partagés avec les autres entités
path pre_shared_key
"/usr/local/v6/etc/psk.txt"


#niveau de verbosité des messages : aucun (ligne commentée), debug ou debug2
log debug2;

remote anonymous # Dans cette partie sont définis les paramètres à utiliser pour la phase 1 par
défaut (anonymous)
```

```
{
        exchange_mode main,aggressive,base;   # type des modes d'échange de la phase 1 acceptable
par ordre de préférence
        lifetime time 24 hour ; # sec,min,heure

        # phase 1 proposal (pour la SA d'ISAKMP)
        proposal {
                encryption_algorithm 3des;
                hash_algorithm sha1;
                authentication_method  pre_shared_key; # C'est ici que l'on définit la méthode
d'authentification
                dh_group 2 ;
        }
        # si le démon est en repondeur, il suit les régles définis par l'initiateur (lifetime,
PFS...)
        proposal_check obey;
}


# phase 2 proposal (pour la SA d'IPsec)
# permet de fixer les paramètres pouvant être utilisés pour la phase 2
sainfo anonymous
{
        #   pfs_group 2;   # Permet d'avoir la propriété PFS
        lifetime time 12 hour ;
        encryption_algorithm 3des, cast128, blowfish 448, des, rijndael ;
        authentication_algorithm hmac_sha1, hmac_md5 ;
        compression_algorithm deflate ;
}
```

Lancer le démon racoon

Une fois les fichiers configurés, il faut lancer le démon racoon (sous l'identité root).

```
#racoon  -F  -f /usr/local/v6/etc/racoon.conf
```

Ici racoon est lancé en mode foreground (-F) (et non pas en mode démon).
Il faut préciser l'emplacement du fichier de configuration grâce à l'option -f.

Ultérieurement après les phases de tests, il suffit de lancer en mode démon :

```
#racoon -f /usr/local/v6/etc/racoon.conf.
```

Pour vérifier que cela fonctionne correctement, il faut lancer, par exemple, un tcpdump et utiliser un outil tel que ping pour initier la connexion.

Lancer sur une console:

```
#tcpdump host 2001:660:305::2
```

Lancer un ping sur une autre console:

```
#ping6 2001:660:305::2
```

On doit alors voir apparaître des logs du type suivant :

```
10:31:56.718729 2001:660:305::1.500 > 2001:660:305::2.500: isakmp: phase 1 I ident: [|sa]
10:31:56.784319 2001:660:305::2.500 > 2001:660:305::1.500: isakmp: phase 1 I ident: [|sa]
10:31:56.788862 2001:660:305::1.500 > 2001:660:305::2.500: isakmp: phase 1 R ident: [|sa]
10:31:56.813376 2001:660:305::2.500 > 2001:660:305::1.500: isakmp: phase 1 R ident: [|sa]
10:31:56.835713 2001:660:305::1.500 > 2001:660:305::2.500: isakmp: phase 1 I ident: [|ke]
10:31:56.897398 2001:660:305::2.500 > 2001:660:305::1.500: isakmp: phase 1 I ident: [|ke]
10:31:56.917716 2001:660:305::1.500 > 2001:660:305::2.500: isakmp: phase 1 R ident: [|ke]
10:31:56.968630 2001:660:305::2.500 > 2001:660:305::1.500: isakmp: phase 1 R ident: [|ke]
10:31:56.994983 2001:660:305::1.500 > 2001:660:305::2.500: isakmp: phase 1 I ident[E]: [encrypted
id]
10:31:57.211360 2001:660:305::2.500 > 2001:660:305::1.500: isakmp: phase 1 I ident[E]: [encrypted
id]
10:31:57.216676 2001:660:305::1.500 > 2001:660:305::2.500: isakmp: phase 1 R ident[E]: [encrypted
id]
10:31:57.226592 2001:660:305::1.500 > 2001:660:305::2.500: isakmp: phase 2/others R inf[E]:
[encrypted hash]
```

```
10:31:57.229601 2001:660:305::2.500 > 2001:660:305::1.500: isakmp: phase 1 R ident[E]: [encrypted
id]
10:31:57.236092 2001:660:305::2.500 > 2001:660:305::1.500: isakmp: phase 2/others R inf[E]:
[encrypted hash]
10:31:57.256860 2001:660:305::1.500 > 2001:660:305::2.500: isakmp: phase 2/others R
oakley-quick[E]: [encrypted hash]
10:31:57.396353 2001:660:305::2.500 > 2001:660:305::1.500: isakmp: phase 2/others I
oakley-quick[E]: [encrypted hash]
10:31:57.411474 2001:660:305::1.500 > 2001:660:305::2.500: isakmp: phase 2/others R
oakley-quick[E]: [encrypted hash]
10:31:57.521524 2001:660:305::2.500 > 2001:660:305::1.500: isakmp: phase 2/others R
oakley-quick[E]: [encrypted hash]
10:31:57.565673 2001:660:305::1.500 > 2001:660:305::2.500: isakmp: phase 2/others I
oakley-quick[E]: [encrypted hash]
10:31:57.704695 2001:660:305::1 > 2001:660:305::2: ESP(spi=0x0052ef02,seq=0x1)
10:31:57.825908 2001:660:305::2.500 > 2001:660:305::1.500: isakmp: phase 2/others R
oakley-quick[E]: [encrypted hash]
10:31:58.704539 2001:660:305::1 > 2001:660:305::2: ESP(spi=0x0b91e340,seq=0x1)
10:31:58.705409 2001:660:305::2 > 2001:660:305::1: ESP(spi=0x008e29b7,seq=0x1)
10:31:59.704376 2001:660:305::1 > 2001:660:305::2: ESP(spi=0x0b91e340,seq=0x2)
10:31:59.705201 2001:660:305::2 > 2001:660:305::1: ESP(spi=0x008e29b7,seq=0x2)
```

Lorsqu'on voit les paquets du type ESP(spi=...,seq=...), la connexion s'est bien établie.

2.3.2 La méthode par certificat X509

Une autre méthode est d'utiliser des certificats au format X509.
Ces certificats sont alors utilisés pour l'authentification des entités.

Création d'une autorité de certification :

`#openssl genrsa -out ca.key 512`
permet de créer un coule de clés publique et privée pour la CA (Certification Authority)

`#openssl req -new -x509 -key ca.key -out ca.crt`
Génération d'une demande de certification

Des questions sont alors posées à l'utilisateur:

```
#openssl req -new -x509 -key ca.key -out ca.crt
Using configuration from /usr/share/ssl/openssl.cnf
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:FR
State or Province Name (full name) [Berkshire]:.
Locality Name (eg, city) [Newbury]:Rennes
Organization Name (eg, company) [My Company Ltd]:.
Organizational Unit Name (eg, section) []:.
Common Name (eg, your name or your server's hostname) []:courtay
Email Address []:.

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Les champs en **gras** sont des exemples de réponses possibles (faire ' **.**' pour laisser l'entrée vide).

On peut visualiser le certificat en utilisant la commande :
`#openssl x509 -text < ca.crt`

Cette autorité de certification sera placée sur une machine. Les autres machines devront passer par elle pour certifier leur certificats.
Les machines possédant un certificat issu de cette autorité de certification devront posséder son certificat.

Génération d'un certificat :

```
#openssl genrsa -out local.key 1024

#openssl req -new -key local.key  -out local.csr
Using configuration from /usr/share/ssl/openssl.cnf
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:FR
State or Province Name (full name) [Berkshire]:.
Locality Name (eg, city) [Newbury]:Rennes
Organization Name (eg, company) [My Company Ltd]:.
Organizational Unit Name (eg, section) []:.
Common Name (eg, your name or your server's hostname) []:xbox.dnssec.rennes.enst-bretagne.fr
Email Address []:.

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:


#openssl x509 -req -days 365 -in local.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out
local.crt
```

Ici notre CA certifie le certificat de la machine xbox.dnssec.rennes.enst-bretagne.fr pour 365 jours.

Placer les certificats :

Une fois les certificats générés, il faut les placer dans une arborescence.
Créer un repertoire /usr/local/openssl/certs (accessible uniquement par root).

Placer les fichiers : ca.crt xbox.crt xbox.key (renommer local.key en xbox.key) dans ce répertoire.

Lancer la commande :

```
# openssl x509 -noout -hash -in ca.crt
```

Cela génère un hash spécifique au certificat qui permet au programme de retrouver le bon certificat de la CA.

exemple :

```
xbox# openssl x509 -noout -hash -in /usr/local/openssl/certs/ca.crt
f18a7346
```

Il faut lier le certificat avec un fichier ayant comme nom ce hash suivi de .0

exemple :

```
ln -s /usr/local/openssl/certs/ca.crt /usr/local/openssl/certs/f18a7346.0
```

Configurer le ficher racoon.conf:
En **gras** les éléments nouveaux par rapport à la méthode pre-shared-key.

```
path certificate "/usr/local/openssl/certs" ;
log debug2;

remote anonymous
{
        exchange_mode main,aggressive,base;

        my_identifier asn1dn;
        certificate_type x509 "xbox.crt" "xbox.key";

        lifetime time 24 hour ;
        passive off;
```

```
        proposal {
                encryption_algorithm 3des;
                hash_algorithm sha1;
                authentication_method rsasig;
                dh_group 2 ;
        }
        proposal_check obey;
}

sainfo anonymous
{
        pfs_group 2;
        lifetime time 12 hour ;
        encryption_algorithm 3des, cast128, blowfish 448, des, rijndael ;
        authentication_algorithm hmac_sha1, hmac_md5 ;
        compression_algorithm deflate ;
}
```

Le fichier /etc/ipsec.conf peut rester inchangé.
La procédure de test est aussi inchangée.

2.3.3 Certificat X509 via DNS

Dans le cas précédent les certificats sont envoyés lors de la transaction, une autre méthode consiste à récupérer ces certificats à partir du DNS.
Cela apporte une securité supplémentaire dans le cas des certificats auto-certifiés, la présence du certificat dans le DNS lui confère une certaine légitimité au niveau de la zone distante.
Par exemple, dans une entreprise, le fait de récupérer les certificats sur le DNS empêche un ordinateur portable intrus de se faire passer pour un ordinateur local à l'entreprise. Cette sécurité n'est que toute relative.

Evidemment, un simple DNS n'étant pas sécurisé, cette méthode n'est pas sécurisée.
Il faut donc utiliser DNSsec plutôt que le DNS classique et que le démon racoon vérifie la validité DNSsec des éléments qui lui sont fournis...
La validitation peut toutefois s'effectuer au niveau de la PKI X509.

Pour mettre en place un certificat X509 dans un DNS, il faut utiliser le bon format.
Pour cela on peut utiliser un utilitaire perl appelé **makecertrrs.pl** qui prend en argument un certificat X509 et qui donne en sortie un RR de type CERT à positionner dans le ficher de la zone.

exemple:

```
# perl makecertrrs.pl < /usr/local/openssl/certs/xbox.crt
  olivier.courtay.irisa.fr. 86400 IN CERT PKIX 7229 RSAMD5 (
        MIIBczCCAR0CAQIwDQYJKoZIhvcNAQEEBQAwMDELMAkGA1UEBhMCRlIxDzANBgNVBAcTBlJlbm5l
        czEQMA4GA1UEAxMHY291cnRheTAeFw0wMzAxMDkwODQ5MDBaFw0wNDAxMDkwODQ5MDBaMFkxCzAJ
        BgNVBAYTAkZSMQ8wDQYDVQQHEwZSZW5uZXMxEDAOBgNVBAMTB2NvdXJ0YXkxJzAlBgkqhkiG9w0B
        CQEWGG9saXZpZXIuY291cnRheUBpcmlzYS5mcjBcMA0GCSqGSIb3DQEBAQUAA0sAMEgCQQC8Yvbs
        9ACOfxseamDgkVdDQRgwsc7+ryVFTXe1EHecccfP2yHGnRUj2fBsGaiS24cToIGu+dq2ER3f+TYB
        DDiFAgMBAAEwDQYJKoZIhvcNAQEEBQADQQAQgc1GTvXw73ZKZhnhr1HoXHc6QTx2kZ/NDW7KXdl4
        SlgBNqhqQkZ1VoJd3nz90HrtWwFTY4PZ7daYZiFhHD2H
        )
```

Il faut modifier le premier champ et mettre à la place par exemple *xbox.dnssec.rennes.enst-bretagne.fr*.

Puis il faut rajouter dans le fichier de configuration racoon.conf une ligne précisant à racoon d'aller chercher le certificat sur le DNS et non pas d'utiliser le payload des paquets échangés (dont on peut désactiver l'envoi en mettant la ligne " **send_cert no;**").

```
path certificate "/usr/local/openssl/certs" ;

remote anonymous
{
        exchange_mode main,aggressive,base;

        my_identifier fqdn "chronos.dnssec.rennes.enst-bretagne.fr";
```

```
        certificate_type x509   "chronos.crt" "chronos.key";
        peers_certfile dnssec;

        lifetime time 24 hour ; # sec,min,hour
        support_mip6 on;

        proposal {
                encryption_algorithm 3des;
                hash_algorithm sha1;
                authentication_method rsasig ;
                dh_group 2 ;
        }
        proposal_check obey;
}
```

2.4 Etude de cas pour setkey

Exemple de fichier de configuration /etc/ipsec.conf.
voir le manuel de setkey pour plus d'info et d'exemple.

ESP en mode tunnel

Ici le fichier qui correspond à un tunnel qui sécurise deux réseaux 2001:688:1f98:1d5a:200:c0ff::/96 et 2001:688:1f98:1d5a:2c0:4fff::/96.
Il y a deux passerelles entre ces deux réseaux qui sont 2001:660:284:100:200:c0ff:fe4a:d8c5 et 2001:660:284:100:2c0:4fff:fea9:8128.
Et on décide de chiffrer toutes les données TCP qui transitent entre ces deux réseaux dans un tunnel établi entre ces deux passerelles.
Ce fichier correpond au fichier de la machine 2001:660:284:100:200:c0ff:fe4a:d8c5

```
flush;
spdflush;
spdadd  2001:688:1f98:1d5a:200:c0ff::/96  2001:688:1f98:1d5a:2c0:4fff::/96  tcp -P in   ipsec \
esp/tunnel/2001:660:284:100:200:c0ff:fe4a:d8c5-2001:660:284:100:2c0:4fff:fea9:8128/require;
spdadd  2001:688:1f98:1d5a:2c0:4fff::/96  2001:688:1f98:1d5a:200:c0ff::/96  tcp -P out ipsec \
esp/tunnel/2001:660:284:100:2c0:4fff:fea9:8128-2001:660:284:100:200:c0ff:fe4a:d8c5/require;
```

Ici, on commence par supprimer toutes les entrées SA (*flush*) puis toutes les entrées SP (*spdflush*).
Puis on rajoute une entrée à la SPD (grâce à *spdadd*).
L'analyse de la troisième ligne donne :
Tout ce qui arrive d'une machine du réseau *2001:688:1f98:1d5a:200:c0ff::/96* en direction d'une machine du réseau 2001:688:1f98:1d5a:2c0:4fff::/96 et qui est en TCP (*tcp*) (les paquets doivent être rentrant (*-P in*)) doivent se voir appliquer IPsec.
Les fonctions IPsec appliquées sont le chiffrement (*esp*) et ont un caractère obligatoire (*require*). Les paquets IP seront chiffrés et seront mis dans un tunnel (*tunnel*) entre les deux passerelles (*2001:660:284:100:200:c0ff:fe4a:d8c5-2001:660:284:100:2c0:4fff:fea9:8128*).

Remarques :
- Avec Kame, en mode tunnel, il n'est pas possible d'utiliser l'Authentification (AH) car AH fait aussi de l'intégrité de paquet, or une fois encapsulé, on ne sait pas quel paquet IP est couvert par la protection. Bien que cela soit spécifié clairement dans les RFC (cf fig 3) cela est rarement implémenté à cause de cette ambiguité.
- En mode tunnel on ne peut pas préciser les ports source et destination (cf plus bas).

AH+ESP en mode transport pour tcp

Exemple de fichier où AH et ESP sont mis en place tout les deux.

```
flush;
spdflush;
spdadd  2001:688:1f98:1d5a:200:c0ff:fe4a:d8c5/128[any]
2001:688:1f98:1d5a:2c0:4fff:fea9:8128/128[53]   tcp  -P in   ipsec esp/transport//require
```

```
ah/transport//require;
spdadd  2001:688:1f98:1d5a:2c0:4fff:fea9:8128/128[53]
 2001:688:1f98:1d5a:200:c0ff:fe4a:d8c5/128[any] tcp  -P out ipsec esp/transport//require
ah/transport//require;
```

Ici, on commence par supprimer toutes les entrées SA (*flush*) puis toutes les entrées SP (*spdflush*).
Puis on rajoute une entrée à SPD (grâce à *spdadd*).
L'analyse de la troisième ligne donne :
Tout ce qui vient de la machine *2001:688:1f98:1d5a:200:c0ff:fe4a:d8c5* en direction de la machine *2001:688:1f98:1d5a:2c0:4fff:fea9:8128* et qui arrive sur la machine (*-P in*) se voit appliquer IPsec (*ipsec*).
Cela ne concerne que le trafic TCP (*tcp*) venant de n'importe quel port (*[any]*) vers le port 53 (*[53]*).
Les règles IPsec appliquées sont *esp/transport//require* et *ah/transport//require*. On veut obligatoirement (*require*) chiffrer (*esp*) les données en mode *transport* et on veut aussi authentifier (*ah)* les données en mode *transport*.

Mise en place non-automatique

La commande setkey permet non seulement de configurer la SPD mais aussi la SAD ce qui permet de créer des liaisons IPsec sans avoir recours à un démon IKE.
On peut donc mettre grâce à la commande setkey les SA à la main. Cette méthode est relativement bien documentée, et peut être réalisée pour sécuriser rapidement une liaison entre deux machines.

Exemple de contenu du fichier /etc/ipsec.conf :

```
flush;
spdflush;
spdadd 2001:660:284:100:200:c0ff:fe4a:d8c5/128  2001:660:284:100:2c0:4fff:fea9:8128/128  tcp -P
in ipsec esp/transport//require;
spdadd 2001:660:284:100:2c0:4fff:fea9:8128/128  2001:660:284:100:200:c0ff:fe4a:d8c5/128  tcp -P
out ipsec esp/transport//require;
add 2001:660:284:100:200:c0ff:fe4a:d8c5  2001:660:284:100:2c0:4fff:fea9:8128  esp 0x10002
-m transport
-E blowfish-cbc "kamekame";
add 2001:660:284:100:2c0:4fff:fea9:8128 2001:660:284:100:200:c0ff:fe4a:d8c5 esp 0x10001
-m transport
-E blowfish-cbc "kamekame";
```

Les premières lignes sont connues.
La commande add permet de rajouter des SA.
On précise l'adresse source et destination que le paquet doit correspondre (*2001:660:284:100:2c0:4fff:fea9:8128 2001:660:284:100:200:c0ff:fe4a:d8c5*)
Puis le protocole IPsec associé à cette SA, ici ESP (*esp*).
Puis le numéro de cette règle appelée SPI (*0x10001*), on peut choisir ce numéro comme on veut mais il doit être unique sur la machine et être le même que sur la machine distante.
Puis on précise le mode (*-m transport*).
Pour finir, l'algorithme utilisé pour chiffrer et la clé associé (*-E blowfish-cbc "kamekame"*).

Il y a deux SA, une pour chaque sens de la communication.

Sur la machine en face, il faut garder presque le même fichier, seules les lignes pour la SPD changent car les paquets sortant pour une machine seront rentrants pour l'autre, on a donc :

```
spdadd 2001:660:284:100:200:c0ff:fe4a:d8c5/128  2001:660:284:100:2c0:4fff:fea9:8128/128  tcp -P
out ipsec esp/transport//require;
spdadd 2001:660:284:100:2c0:4fff:fea9:8128/128  2001:660:284:100:200:c0ff:fe4a:d8c5/128  tcp -P
in  ipsec esp/transport//require;
```

Les règles pour la SADB ne changent pas.

2.5 Choix de configuration d'IPsec

IPsec possède de nombreuses possibilités de mise en œuvre. Cela a pour but de couvrir tous les cas d'utilisations possibles de chiffrement et d'authentification sur un réseau IP.

La liste des options est en effet longue :
- chiffrer et/ou authentifier
- au moyen d'un tunnel ou non
- en manuel ou en automatique
- avec diverses méthodes d'authentification : secret partagé, certificats X509, via le DNS
- grâce à de nombreux algorithmes de chiffrement et de hachage.

Le choix de chiffrer et/ou authentifier dépend essentiellement de l'utilisation que l'on fait. Si on prend l'exemple de transactions bancaires via l'Internet, les deux fonctions d'IPsec seront nécessaires. Mais la mise à jour de fichiers entre deux serveurs DNS publics pourra n'être que authentifiée car les données ne sont pas confidentielles. D'une manière générale, chiffrer sans authentifier n'est pas sûr.

Comme on l'a vu précédemment le choix du mode Tunnel ou Transport dépend de l'architecture à protéger. Deux réseaux voulant sécuriser leurs communications peuvent mettre en place un tunnel sécurisé entre deux passerelles. Mais deux machines seules pourront avantageusement choisir le mode Transport, les en-têtes supplémentaires du mode Tunnel seront économisées.

Le mode de mise en place, manuel ou automatique, dépend de la permanence du lien sécurisé. Dans le cas par exemple de deux filiales d'une entreprise, un lien permanent mis en place manuellement est plus simple à configurer.
Mais si l'on désire avoir une gestion souple des clés avec renouvellement fréquent (plus sécurisé) ou avec des machines avec lesquelles il n'y pas eu de contact préalable, une méthode basée sur IKE devient obligatoire.

Concernant la méthode d'authentification, la méthode du secret partagé a l'avantage d'être simple a mettre en oeuvre. Mais il faut avoir un contact préalable (et sécurisé) avec la machine avec laquelle on veut communiquer. De plus pour que la machine puisse communiquer de façon sécurisée avec 100 machines, il faudrait qu'elle possède 100 clés différentes, cela devient vite difficile a gérer.
La méthode du certificat X509 permet de palier au problème du nombre de clés, c'est grâce à la cryptographie asymétrique que l'authentification est faite (cf paragraphe 1.3.2). La contrainte est qu'il faut mettre en place une PKI X509 (Public Key Infrastructure).
La méthode par DNS n'apporte pas en elle-même une amélioration des certificats, mais à terme si les certificats sont stockés dans le DNSsec, l'authentification pourra être plus aisée car une PKI X509 ne sera plus nécessaire (la structure DNS(sec) étant généralisée).

Les algorithmes de chiffrement et de hachage sont choisis par l'administrateur en fonction de la confiance que celui-çi a en eux et en fonction de l'intéropérabilité recherchée.
Il peut être courant qu'un administrateur refuse les chiffrements avec DES car jugés pas assez sûrs, mais qu'un autre administrateur choisisse plutôt l'interoperabilité avec le maximum de personnes.

2.6 Problèmes éventuels

Les firewalls

Les firewalls doivent être configurés pour permettre l'utilisation de certaines fonctionnalités d'IPsec.

Pour utiliser la négociation IKE, il faut permettre l'utilisation du port 500 en UDP. C'est en effet par ce port que passent tous paquets de type ISAKMP.

Dans le cas où le mode Tunnel est choisi, il faut veiller à ce que les protocoles IP 50 (AH) et 51 (ESP) soient autorisés à passer.

Adresses sources multiples

Si une machine possède plusieurs adresses IP, il faut déterminer quelle est l'adresse source choisie par le système pour atteindre une autre machine.
En effet, comme IPsec se base sur les adresses comme identifiant, une mauvaise adresse source peut empêcher la négociation IPsec.

Un simple ping permet en général de vérifier l'adresse source choisie.

IPsec sur un réseau local

Dans le cas où on utilise IPsec sur un réseau local en IPv6 en mode *require*, il peut y avoir un problème. En mode *require* la connexion IPsec est établie avant tout envoi de paquet. Donc les paquets ICMPv6 neighbour solicitation sont bloqués et l'échange IKE ne peut avoir lieu.

Il faut donc autoriser les messages ICMPv6 neighbour solicitation en réseau local en rajoutant une régle "none" à la configuration de la SPD.

3ème Partie - Configuration d'Isakmpd

3.1 Introduction

Isakmpd est un démon fonctionnant sur le même principe que Racoon.
Ce document regroupe un ensemble d'informations permettant la mise en place de Isakmpd.
Les configurations étudiées précédemment pour Racoon sont ici transposées pour Isakmpd.

3.2 Les fichiers de configuration

3.2.1 isakmpd.conf

Le premier fichier est **isakmpd.conf**, qui contient la description des machines (locales ou distantes) ainsi que la manière dont on peut mettre en place IPsec pour communiquer avec ces machines.

La syntaxe de ce fichier est particulière, il est composé de blocs dont voici la description (semi-formelle) :

[**Tag**]
Entrée = **Information**

Tag est une chaîne de caractères choisie par l'utilisateur.
Entrée est un mot clé reservé (sauf cas exceptionnel) qui précède le caractère "="
Information est soit un mot clé réservé, soit une information donnée par l'utilisateur (exemple une adresse IP), soit une suite d'**Informations** (séparées par des ',') qui suit le caractère "=".
Ainsi les données peuvent être groupées dans des blocs qui sont utilisés par la suite (le placement dans le fichier ne compte pas).

Exemple :

```
[Phase 2]
Connections= work-gw-my-gw
[work-gw-my-gw]
Phase= 2
ISAKMP-peer= work-gw
Local-ID= ps2
Remote-ID= xbox
Configuration= Default-quick-mode
[ps2]
ID-type= IPV4_ADDR
Address= 192.108.119.175
```

En **gras** les Tags définis par l'utilisateur, par exemple le tag **ps2** est utilisé dans le bloc **work-gw-my-gw** et définit dans le bloc **ps2**.
Les blocs **work-gw, xbox, Default-quick-mode** ne sont pas définis dans notre extrait, mais dans un fichier complet ils devront être définis.

De nombreuses valeurs sont définies par défaut cependant l'utilisateur peut les surcharger à sa guise. Il faut donc aussi se méfier car leurs valeurs ne sont pas visibles directement par l'utilisateur et peuvent venir perturber la configuration.
Par exemple la méthode d'authentification par défaut est la méthode du secret partagé. Dans la syntaxe, il faut faire attention aux espaces : jamais d'espace après les attributs, sinon ils seront pris en compte!

Un autre problème est que la page de man de isakmpd.conf ne reflète pas l'état actuel du code réel. Beaucoup de fonctionnalités décrites dans la documentation ne sont pas implémentées.

3.2.2 isakmpd.policy

Le deuxième fichier de configuration est **isakmpd.policy**.
Ce fichier permet de mieux contrôler les choix effectués par les démons IKE, en rajoutant des règles que les attributs IPsec doivent respecter.

Le plus simple (mais suffisant) des fichiers de configuration possible pour isakmpd.policy est :

```
-----------------------------------------------------------------
KeyNote-Version: 2
Authoriser: "POLICY"
-----------------------------------------------------------------
```

3.3 Configurations

Lors de l'étude de Racoon, les méthodes d'authentification par secret partagé et par certificats X509 avaient été présentées.
Nous reprenons ici les cas de configurations équivalentes pour Isakmpd :
- Authentification par secret partagé.
- Authentification par certificat X509.
- Mode tunnel et mode transport.

3.3.1 La méthode du secret patagé

Le fichier de configuration correspond ici à celui de la machine xbox.

```
# bloc d'informations sur le comportement général du démon.
[General]
Policy-File=                                    /etc/isakmpd/isakmpd.policy
Listen-on=                                      2001:660:305::1

#on définit une machine avec laquelle on va établir une communication IPsec
[Phase 1]
2001:660:305::2=                                peer-host

# liste de connexions qui doivent être mises en place automatiquement.
[Phase 2]
Connections=                                    peer-host-my-host

# identité de la machine distante
[chronos]
ID-type=                                        IPV6_ADDR
Address=                                        2001:660:305::2

# identité de la machine locale
[xbox]
ID-type=                                        IPV6_ADDR
Address=                                        2001:660:305::1

#Informations pour la connexion  avec la machine peer-host
[peer-host]
Phase=                                          1
Transport=                                      udp
#on peut préciser l'adresse locale à utiliser s'il y a plusieurs adresses locales disponibles
Local-address=                                  2001:660:305::1
Address=                                        2001:660:305::2
# ici on précise la méthode pour effectuer la connexion
```

```
Configuration=                                          Default-main-mode
# permet de définir l'identité à présenter à la machine distante
ID=                                                     xbox
#secret partagé
Authentication=                                         totosecretqwerty

#definition de certains attributs pour le main mode
[Default-main-mode]
DOI=                                                    IPSEC
EXCHANGE_TYPE=                                          ID_PROT
Transforms=                                             3DES-MD5

[3DES-MD5]
ENCRYPTION_ALGORITHM=                                   3DES_CBC
HASH_ALGORITHM=                                         MD5
# C'est ici qu'on indique qu'on veut effectuer de l'authentification grâce à la méthode du secret
partagé
AUTHENTICATION_METHOD=                                  PRE_SHARED
GROUP_DESCRIPTION=                                      MODP_1024

#paramètres pour la deuxième phase
[peer-host-my-host]
Phase=                                                  2
ISAKMP-peer=                                            peer-host
#les identités à utiliser
Local-ID=                                               xbox
Remote-ID=                                              chronos
Configuration=                                          Default-quick-mode


[Default-quick-mode]
DOI=                                                  IPSEC
EXCHANGE_TYPE=                                        QUICK_MODE
Suites=                                               QM-ESP-3DES-MD5-PFS-SUITE

[QM-ESP-3DES-MD5-PFS-SUITE]
Protocols=                                            QM-ESP-3DES-MD5-PFS

#on précise que l'on veut chiffrer les communications
[QM-ESP-3DES-MD5-PFS]
PROTOCOL_ID=                                          IPSEC_ESP
Transforms=                                           QM-ESP-3DES-MD5-PFS-XF

[QM-ESP-3DES-MD5-PFS-XF]
# on utilise 3DES pour chiffrer
TRANSFORM_ID=                                         3DES
#on utilise le mode transport, on peut mettre TUNNEL pour avoir le mode tunnel.
ENCAPSULATION_MODE=                                   TRANSPORT
AUTHENTICATION_ALGORITHM=                             HMAC_MD5
GROUP_DESCRIPTION=                                    MODP_1024
```

3.3.2 La méthode des certificats X509

Pour utiliser les certificats avec Isakmpd, il faut commencer par construire son propre certificat.
On utilisera pour cela la documentation de configuration de Racoon.

Il faut utiliser obligatoirement les Subject Alternatives Names (**SubjAltName**) décrits en Annexe 2.
Comme les certificats X509v3 sous openssl ne supportent que les adresses IPv4 ou les FQDN, on choisira pour IPv6 le FQDN.
Pour la phase deux (Quick Mode) seules les adresses IP sont supportées par Isakmpd, il faut donc utiliser des identifiants différents pour la phase un et la phase deux.

Voici les fichiers de configuration des deux machines (test5 et test6).
La première machine (test5) va être configurée en mode où elle va tenter d'établir une connexion IPsec avec l'autre machine (test6).
La seconde machine (test6) est passive et attend une demande d'échange ISAKMP venant de n'importe qui (le Tag **Default** est utilisé pour définir le comportement par défaut).

Concernant les certificats voici le contenu des répertoires de test5 ou test6 dans ce cas:

```
/etc/isakmpd/ca/ :
                  ca.crt
```

```
                    1980016a.0 -> ca.crt
/etc/isakmpd/certs/ :
                    local.crt
 /etc/isakmpd/private/:
                    local.key
```

Fichier de configuration de test5:

```
[General]
Policy-File=                                    /etc/isakmpd/isakmpd.policy
Retransmits=                                    2
Exchange-max-time=                              10
Listen-on=                                      2001:688:1f98:1d5a:2c0:4fff:fea9:8128

#chemin d'accès aux certificats
[X509-certificates]
CA-directory=                                   /etc/isakmpd/ca/
Cert-directory=
Private-key=                                    /etc/isakmpd/private/local.key

[Phase 1]
2001:688:1f98:1d5a:200:c0ff:fe4a:d8c5=          peer-host

[Phase 2]
Connections=                                    peer-host-my-host

[peer-host]
Phase=                                          1
Transport=                                      udp
Local-address=                                  2001:688:1f98:1d5a:2c0:4fff:fea9:8128
Address=                                        2001:688:1f98:1d5a:200:c0ff:fe4a:d8c5
Configuration=                                  Default-main-mode
#pour la phase 1 on utilise le FQDN comme identifiant
ID=                                             test5-ID

[test5-ID]
ID-type=                                        FQDN
Name=                                           test5.armor.irisa.fr

[test5]
ID-type=                                        IPV6_ADDR
Address=                                        2001:688:1f98:1d5a:2c0:4fff:fea9:8128

[test6]
ID-type=                                        IPV6_ADDR
Address=                                        2001:688:1f98:1d5a:200:c0ff:fe4a:d8c5

[peer-host-my-host]
Phase=                                          2
ISAKMP-peer=                                    peer-host
# pour la phase 2, on utilise les adresses IP comme identifiants.
Local-ID=                                       test5
Remote-ID=                                      test6
Configuration=                                  Default-quick-mode

[Default-main-mode]
DOI=                                            IPSEC
EXCHANGE_TYPE=                                  ID_PROT
Transforms=                                     3DES-MD5


[Default-quick-mode]
DOI=                                            IPSEC
EXCHANGE_TYPE=                                  QUICK_MODE
Suites=                                         QM-ESP-3DES-MD5-PFS-SUITE


[3DES-MD5]
ENCRYPTION_ALGORITHM=                           3DES_CBC
HASH_ALGORITHM=                                 MD5
#on utilise RSA_SIG ou autrement dit les certificats X509
AUTHENTICATION_METHOD=                          RSA_SIG
GROUP_DESCRIPTION=                              MODP_1024

[QM-ESP-3DES-MD5-PFS-SUITE]
Protocols=                                      QM-ESP-3DES-MD5-PFS
```

```
[QM-ESP-3DES-MD5-PFS]
#on chiffre les commnications
PROTOCOL_ID=                                    IPSEC_ESP
Transforms=                                     QM-ESP-3DES-MD5-PFS-XF

[QM-ESP-3DES-MD5-PFS-XF]
#on utilise 3DES pour chiffrer les communications
TRANSFORM_ID=                                   3DES
ENCAPSULATION_MODE=                             TRANSPORT
AUTHENTICATION_ALGORITHM=                       HMAC_MD5
GROUP_DESCRIPTION=                              MODP_1024
```

Fichier de configuration de test6:

```
[General]
Policy-File=                                      /etc/isakmpd/isakmpd.policy
Listen-on=                                        2001:688:1f98:1d5a:200:c0ff:fe4a:d8c5

[X509-certificates]
CA-directory=                                     /etc/isakmpd/ca/
Cert-directory=                                   /etc/isakmpd/certs/
Private-key=                                      /etc/isakmpd/private/local.key

[Phase 1]
# les lignes précédées par ### sont les parties mises en commentaire pour permettre d'effectuer
la communication avec tout le monde.
###2001:688:1f98:1d5a:2c0:4fff:fea9:8128=        peer-host
Default=                                          peer-host

[Phase 2]
Connections=                                      peer-host-my-host

[peer-host]
Phase=                                            1
###Transport=                                     udp
###Address=                                       2001:688:1f98:1d5a:2c0:4fff:fea9:8128
# on met comme Address :: c'est-à-dire l'adresse par défaut pour spécifier qu'il s'agit de
n'importe qui.
Address=                                          ::
Local-address=                                    2001:688:1f98:1d5a:200:c0ff:fe4a:d8c5
Configuration=                                    Default-main-mode
ID=                                               test6-ID

[test6-ID]
ID-type=                                          FQDN
Name=                                             test6.armor.irisa.fr

[test6]
ID-type=                                          IPV6_ADDR
Address=                                          2001:688:1f98:1d5a:200:c0ff:fe4a:d8c5

[peer-host-my-host]
Phase=                                            2
ISAKMP-peer=                                      peer-host
Local-ID=                                         test6
###Remote-ID=                                     test5
Configuration=                                    Default-quick-mode

[Default-main-mode]
DOI=                                              IPSEC
EXCHANGE_TYPE=                                    ID_PROT
Transforms=                                       3DES-MD5

[Default-quick-mode]
DOI=                                              IPSEC
EXCHANGE_TYPE=                                    QUICK_MODE
Suites=                                           QM-ESP-3DES-MD5-PFS-SUITE

[3DES-MD5]
ENCRYPTION_ALGORITHM=                             3DES_CBC
HASH_ALGORITHM=                                   MD5
AUTHENTICATION_METHOD=                            RSA_SIG
GROUP_DESCRIPTION=                                MODP_1024

[QM-ESP-3DES-MD5-PFS-SUITE]
Protocols=                                        QM-ESP-3DES-MD5-PFS
```

```
[QM-ESP-3DES-MD5-PFS]
PROTOCOL_ID=                                    IPSEC_ESP
Transforms=                                     QM-ESP-3DES-MD5-PFS-XF

[QM-ESP-3DES-MD5-PFS-XF]
TRANSFORM_ID=                                   3DES
ENCAPSULATION_MODE=                             TRANSPORT
AUTHENTICATION_ALGORITHM=                       HMAC_MD5
GROUP_DESCRIPTION=                              MODP_1024
```

Remarque :
Ce mode de configuration n'est pas très stable, et fonctionne bizarrement parfois! Il y des phases essais/erreurs de la part d'Isakmpd, mais il finit par établir la communication sécurisée.

3.4 Lancement de Isakmpd

Lors des tests, il est conseillé de lancer isakmpd en mode non démon et en mode debug.
La commande est alors:

```
isakmpd -d -DA=99 -c /etc/isakmpd/isakmpd.conf
```

La sortie donne alors des indications sur les causes d'erreurs. Les erreurs les plus importantes sont rapportées dans la catégorie **Default.**

Si l'on arrête le programme à l'aide d'un Ctrl-C, il faut supprimer "à la main" les éventuelles règles mises en place par le programme.
Pour cela faire un *setkey -FP* (suppression de toutes les règles de la SPD) et un *setkey -F (*suppression de toutes les règles de la SADB).

4 Bibliographie

Ghislaine Labouret, *IPSEC : Présentation technique* . 16 juin 2000. Hervé Schauer Consultants (http://www.hsc.fr/resssources/ipsec).
Joseph Benoit, *Modélisation des performances du protocole IPsec* . 2000. (http://benoit-joseph.mine.nu/tfe/).
Patrick Ethier, ISAKMP and IPsec in the VPN envireonment. 2000. (http://www.secureops.com/vpn/ipsecvpn.html).
Jörgen Granstam, How to use X.509v3 certificates for authentication with isakmp. 2000. (http://hem.passagen.se/hojg/isakmpd/).
Nicolas Bernard, Etablissement de tests permettant de mesurer l'impact sur le performances de la sécurisation d'IPv6 par IPsec (http://www.ens-lyon.fr/~nbernard/ipsec/index.html).

RFC 2401, *Security Architecture for the Internet Protocol*, nov 1998.
RFC 2411, *IP security Document Roadmap*, nov 1998.
RFC 2402, *IP Authentification Header*, nov 1998.
RFC 2406, *IP Encapsulation Header*, nov 1998.
RFC 2408, *Internet Security Association and Key Management Protocol (ISAKMP)*, nov 1998.
RFC 2407, *The Internet IP Security Domain of Interpretation for ISAKMP* , nov 1998.
RFC 2408, *Internet Security Association and Key Management Protocol (ISAKMP)* , nov 1998.
RFC 2409, *The Internet Key Exchange (IKE)* , nov 1998.

- les pages de Man de :
  - *racoon* pour les options de lancement du demon
  - *racoon.conf* pour les options de configuration dans le ficher **racoon.conf**

- *setkey* pour configurer le fichier **ipsec.conf**

- Le README.PKI expliquant les démarches pour créer des certificats et utiliser **certpatch**.
http://www.openbsd.org/cgi-bin/cvsweb/src/sbin/isakmpd/README.PKI?rev=1.7

- Le site *http://tirian.magd.ox.ac.uk/~nick/openssl-certs/others.shtml* comporte de bon conseil sur la manipulation des certificats.

- Référence de **makecertrrs.pl** : *http://research.verisign.com/Projects/cert-dns-smime.htm*

- Ethereal : http://ethereal.com/

- Dsniff : (Dug Song) http://naughty.monkey.org/~dugsong/dsniff/

Annexe 1 : les certificats X509

Les certificats X509 sont couramment utilisés dans le monde des PKI (Public Key Infrastructure).
Ce sont des fichiers qui contiennent une **clé publique** d'une personne, d'une machine ou d'une organisation.

Ces certificats contiennent aussi l'identité de la personne ainsi que celle d'une **Autorité de Certification** et une signature d'eux-même par cette Autorité de Certification.

Cette signature est un hash du certificat signé par la clé privé de l'autorité de certification.
Cela permet de vérifier que la clé publique est bien celle de la personne indiquée dans le certificats.

Il faut évidemment avoir confiance en cette autorité de certification. La clé publique de cette autorité de certification peut elle même être certifiée par des autorités supérieures de certification. Pour les plus hautes autorités de certifications, il faut que tout le monde connaisse leurs clés publiques (ou leurs certificats).

Les Certificats ne sont valides que pendant une période qui est précisée dans les certificats (et protégée par la signature).
De plus, il existe aussi (pour X509v3) une Liste de Révocation de Certificat (CRL), qui permet de dénoncer un certificat avant sa date limite. Toute personne vérifiant un certificat doit consulter l'autorité de certification pour vérifier qu'il n'a pas été revoqué.

Dans X509v3, il est possible de rajouter des extentions qui fournissent des informations supplémentaires.

Exemple de contenu de certificat X509 :

```
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number: 2 (0x2)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=FR, L=Rennes, CN=courtay
        Validity
            Not Before: Jan  9 08:49:00 2003 GMT
            Not After : Jan  9 08:49:00 2004 GMT
        Subject: C=FR, L=Rennes, CN=courtay/Email=olivier.courtay@irisa.fr
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (512 bit)
                Modulus (512 bit):
                    00:bc:62:f6:ec:f4:00:8e:7f:1b:1e:6a:60:e0:91:
                    57:43:41:18:30:b1:ce:fe:af:25:45:4d:77:b5:10:
                    77:9c:71:c7:cf:db:21:c6:9d:15:23:d9:f0:6c:19:
                    a8:92:db:87:13:a0:81:ae:f9:da:b6:11:1d:df:f9:
                    36:01:0c:38:85
                Exponent: 65537 (0x10001)
    Signature Algorithm: md5WithRSAEncryption
        10:81:cd:46:4e:f5:f0:ef:76:4a:66:19:e1:af:51:e8:5c:77:
        3a:41:3c:76:91:9f:cd:0d:6e:ca:5d:d9:78:4a:58:01:36:a8:
        6a:42:46:75:56:82:5d:de:7c:fd:d0:7a:ed:5b:01:53:63:83:
```

```
        d9:ed:d6:98:66:21:61:1c:3d:87
-----BEGIN CERTIFICATE-----
MIIBczCCAR0CAQIwDQYJKoZIhvcNAQEEBQAwMDELMAkGA1UEBhMCRlIxDzANBgNV
BAcTBlJlbm5lczEQMA4GA1UEAxMHY291cnRheTAeFw0wMzAxMDkwODQ5MDBaFw0w
NDAxMDkwODQ5MDBaMFkxCzAJBgNVBAYTAkZSMQ8wDQYDVQQHEwZSZW5uZXMxEDAO
BgNVBAMTB2NvdXJ0YXkxJzAlBgkqhkiG9w0BCQEWGG9saXZpZXIuY291cnRheUBp
cmlzYS5mcjBcMA0GCSqGSIb3DQEBAQUAA0sAMEgCQQC8Yvbs9ACOfxseamDgkVdD
QRgwsc7+ryVFTXe1EHecccfP2yHGnRUj2fBsGaiS24cToIGu+dq2ER3f+TYBDDiF
AgMBAAEwDQYJKoZIhvcNAQEEBQADQQAQgc1GTvXw73ZKZhnhr1HoXHc6QTx2kZ/N
DW7KXdl4SlgBNqhqQkZ1VoJd3nz90HrtWwFTY4PZ7daYZiFhHD2H
-----END CERTIFICATE-----
```

Annexe 2 : ajout d'un Subject Alternative Name

Racoon peut avoir besoin pour utiliser le certificat d'un champ d'information supplémentaire appelé le *"Subject Alternative Name"*.
Il peut correspondre à un FQDN, en général le nom de la machine, mais aussi à une adresse IP.
Cela peut être nécessaire notamment pour l'interopérabilité avec Isakmpd qui n'utilise que ce système.

Pour ajouter ce champ, il faut utiliser sous FreeBSD l'utilitaire certpatch :

#certpatch -t fqdn -i xbox.dnssec.rennes.enst-bretagne.fr -k ca.key local.crt xbox.crt

L'utilitaire certpatch est fourni avec le démon IKE Isakmpd.
Le certificat local.crt se voit ajouter une extension X509v3 SubAltName et xbox.crt correspond à ce nouveau certificat.

Visualisation d'un certificat :

```
xbox# openssl x509 -text < /usr/local/openssl/certs/xbox.crt
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 2 (0x2)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=FR, L=Rennes, CN=courtay
        Validity
            Not Before: Jan  9 08:49:00 2003 GMT
            Not After : Jan  9 08:49:00 2004 GMT
        Subject: C=FR, L=Rennes, CN=courtay/Email=olivier.courtay@irisa.fr
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (512 bit)
                Modulus (512 bit):
                    00:bc:62:f6:ec:f4:00:8e:7f:1b:1e:6a:60:e0:91:
                    57:43:41:18:30:b1:ce:fe:af:25:45:4d:77:b5:10:
                    77:9c:71:c7:cf:db:21:c6:9d:15:23:d9:f0:6c:19:
                    a8:92:db:87:13:a0:81:ae:f9:da:b6:11:1d:df:f9:
                    36:01:0c:38:85
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Alternative Name: critical
                DNS:xbox.dnssec.rennes.enst-bretagne.fr
    Signature Algorithm: md5WithRSAEncryption
        a4:3e:61:68:04:9f:93:77:71:06:96:5a:6b:cc:44:a1:f3:00:
        32:b9:38:21:25:9c:e9:d3:03:35:19:94:1e:4c:8e:05:b9:b0:
        d8:37:a7:51:3a:fe:6c:b3:2c:73:e6:14:8b:f9:91:80:d7:8c:
        ab:6c:17:ff:a7:46:16:e1:71:5e
-----BEGIN CERTIFICATE-----
MIIBrzCCAVmgAwIBAgIBAjANBgkqhkiG9w0BAQQFADAwMQswCQYDVQQGEwJGUjEP
MA0GA1UEBxMGUmVubmVzMRAwDgYDVQQDEwdjb3VydGF5MB4XDTAzMDEwOTA4NDkw
MFoXDTA0MDEwOTA4NDkwMFowWTELMAkGA1UEBhMCRlIxDzANBgNVBAcTBlJlbm5l
czEQMA4GA1UEAxMHY291cnRheTEnMCUGCSqGSIb3DQEJARYYb2xpdmllci5jb3Vy
dGF5QGlyaXNhLmZyMFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBALxi9uz0AI5/Gx5q
YOCRV0NBGDCxzv6vJUVNd7UQd5xxx8/bIcadFSPZ8GwZqJLbhxOgga752rYRHd/5
NgEMOIUCAwEAAaM1MDMwMQYDVR0RAQH/BCcwJYIjeGJveC5kbnNzZWMucmVubmVz
LmVuc3QtYnJldGFnbmUuZnIwDQYJKoZIhvcNAQEEBQADQQCkPmFoBJ+Td3EGllpr
zESh8wAyuTghJZzp0wM1GZQeTI4FubDYN6dROv5ssyxz5hSL+ZGA14yrbBf/p0YW
4XFe
-----END CERTIFICATE-----
```

Il faut alors dans le fichier racoon.conf changer l'entrée my_identifier :

```
my_identifier fqdn "xbox.dnssec.rennes.enst-bretagne.fr";
```

Par exemple pour dans le cas d'un SubAltName de type fqdn et de valeur box.dnssec.rennes.enst-bretagne.fr

Annexe 3 : performances d'IPsec

Une des questions posées est de quantifier la chute de performances due à IPsec par rapport à une liaison classique non sécurisée.

Pour cela, des tests simples ont été realisés. Il s'agit d'un "ping" modifié qui permet l'émission de requêtes ICMP echo en quantité desirée. La variante utilisée est donc un "ping flood", qui permet de saturer la liaison, et donc de pousser les logiciels jusqu'à des limites intéressantes.
Les éléments à prendre en compte sont le temps moyen de transit sur un grand nombre de paquets et le taux de perte. Le premier élément remarqué est le comportement similaire des différentes implémentations d'IPsec, qui sont comparables entre-elles en terme de performances.

Le tableau ci-dessous illustre les résultats chiffrés de quatre tests réalisés entre deux machines et significatifs de la tendance que nous avons observée.

	flooding	nombre de paquets transmis	temps moyen	pertes
liaison en clair	non	2 000	181 ms	0 %
IPsec	non	2 000	426 ms	0 %
liaison en clair	oui	240 000	168 ms	30 %
IPsec	oui	230 000	394 ms	29 %

La différence est extrêmement conséquente entre la liaison en clair et la liaison IPSEC, mais le service que rend IPSEC notamment au niveau de la confidentialité et de l'authentification peut, dans certains cas, rendre tolérable une telle chute des performances.

Annexe 4 : Prise en compte d'IPsec par le noyau

Il est peut-être obligatoire de recompiler le noyau de FreeBSD pour qu'il incorpore les fonctionnalités IPsec.
Pour savoir si le noyau supporte IPsec il suffit de lancer la commande setkey -D, si on obtient la réponse "pfkey_open: Protocol not found" alors c'est qu'IPsec n'est pas supporté.

Pour activer IPsec :
- éditer le fichier /usr/src/sys/i386/conf/GENERIC (adapter suivant votre architecture matérielle).
- rajouter les lignes suivantes :
  options IPSEC
  options IPSEC_ESP
  options IPSEC_DEBUG (optionnel)
- lancer les commandes
  config GENERIC
  cd ../../compile/GENERIC
  make depend
  make
  make install

Annexe 5 : patch

Isakmpd comporte certains problèmes sous IPv6, voici un patch à appliquer avant utilisation.

Pour cela placez-vous dans le repertoire /usr/port/security/isakmpd de votre FreeBSD
et lancez la commande patch -p1 < patch-isakmpd

Le fichier patch-isakmpd :

```
---------------------------------------------------------------------------------------------------------------------------
diff -urNb isakmpd/work/isakmpd/pf_key_v2.c isakmpd-IDsA/work/isakmpd/pf_key_v2.c
--- isakmpd/work/isakmpd/pf_key_v2.c    Wed Apr  3 14:08:15 2002
+++ isakmpd-IDsA/work/isakmpd/pf_key_v2.c       Wed Dec  4 16:52:25 2002
@@ -1519,6 +1519,9 @@
 {
   int n;
   bit_ffc (mask, 128, &n);
+
+  if(n == -1) n=128;
+
   return n;
 }

diff -urNb isakmpd/work/isakmpd/sysdep/freebsd/sysdep.c
isakmpd-IDsA/work/isakmpd/sysdep/freebsd/sysdep.c
--- isakmpd/work/isakmpd/sysdep/freebsd/sysdep.c        Fri Oct 26 14:23:45 2001
+++ isakmpd-IDsA/work/isakmpd/sysdep/freebsd/sysdep.c   Fri Dec  6 16:32:07 2002
@@ -137,6 +137,9 @@
   char *policy[] = { "in bypass", "out bypass", NULL };
   char **p;
   int ipp;
+ int opt;
+  char *msgstr;

   if (app_none)
     return 0;
@@ -145,9 +148,13 @@
     {
     case AF_INET:
       ipp = IPPROTO_IP;
+     opt = IP_IPSEC_POLICY;
+     msgstr = "";
       break;
     case AF_INET6:
       ipp = IPPROTO_IPV6;
+      opt = IPV6_IPSEC_POLICY;
+     msgstr = "V6";
       break;
     default:
       log_print ("sysdep_cleartext: unsupported protocol family %d", af);
@@ -167,20 +174,20 @@
         log_error ("sysdep_cleartext: %s: %s", *p, ipsec_strerror());
         return -1;
       }
-
-     if (setsockopt(fd, ipp, IP_IPSEC_POLICY, buf,
-                    ipsec_get_policylen(buf)) < 0)
+if (setsockopt(fd, ipp, opt, buf, ipsec_get_policylen(buf)) < 0)
       {
         log_error ("sysdep_cleartext: "
```

```
-                 "setsockopt (%d, IPPROTO_IP, IP_IPSEC_POLICY, ...) failed",
-                 fd);
+                   "setsockopt (%d, IPPROTO_IP%s, IP%s_IPSEC_POLICY, ...) "
+                   "failed", fd, msgstr, msgstr);
          return -1;
        }
+
      free(buf);
    }

  return 0;
 }
+

 int
 sysdep_ipsec_delete_spi (struct sa *sa, struct proto *proto, int incoming)
```
