**MASTER EN SCIENCES INFORMATIQUES**

# Étude et mise en œuvre d'une infrastructure de connexion sécurisée par Ipsec

Landrain, Christophe

*Award date:*
2002


**Facultés Universitaires Notre Dame de la Paix, Namur**
**Institut d'Informatique**
**Année Académique 2001-2002**

# Etude et mise en oeuvre d'une infrastructure de connexion sécurisée par Ipsec

**Christophe Landrain**

**Mémoire présenté en vue de l'obtention du grade de Maître en Informatique.**

# Etude et mise en oeuvre d'une infrastructure de connexion sécurisée par IPsec

Christophe Landrain

4 septembre 2002

# Résumé

Ce document décrit la mise en oeuvre d'une infrastructure d'accès sécurisé par IPsec dans le cadre d'une entreprise. Les étapes d'analyse, de mise en oeuvre et de test du projet y sont examinés en détail. Le premier chapitre expose les concepts théoriques nécessaires à la compréhension des technologies utilisées, en commençant par un rappel des protocoles TCP/IP, puis en couvrant l'ensemble des protocoles IPsec après en avoir introduit le fonctionnement général. Le deuxième chapitre décrit les caractéristiques principales de l'implémentation d'IPsec utilisée dans le projet. Enfin, le dernier chapitre détaille le cheminement suivi pendant la réalisation du projet, de l'analyse du problème spécifique de la société 3S à la mise en place d'une solution complète et fonctionnelle, répondant aux impératifs de sécurité. Les tests effectués pour valider cette solution sont également décrits.

L'ensemble se termine par quelques mots sur l'avenir du projet, ainsi des perspectives de développements futurs basés sur le savoir-faire acquis au cours de ce projet.

**Mots clefs :** IPsec, VPN, ISAKMP, IKE, Certificats, Sécurité des réseaux, OpenBSD.

## Abstract

This document is about the setup of an IPsec-based infrastructure for secured remote access to a corporate network. The analysis, implementation and testing steps are thoroughly detailed. The first chapter introduces the theoretical concepts required for the understanding of technologies involved, beginning with a recall of TCP/IP protocols, followed by an overview of IPsec general mecanisms, the rest of the chapter details each protocol of IPsec. The second chapter shows the main characteristics of the OpenBSD IPsec implementation that is used in our project. The last chapter describes in detail the different steps that were followed in the realization of the project, from 3S's specific problem analysis to the final setup of a fully functional solution that complies with security requirements. Tests that were made in order to validate the solution are also described.

The whole ends on a few words about the project's futur, and about possibilities of other developments based on the know-how acquired during its realization.

**Keywords :** IPsec, VPN, ISAKMP, IKE, PKI, Network Security, OpenBSD.

# Avant-propos

La réalisation de ce projet n'aurait pas pu avoir lieu sans l'aide et le soutient dont m'ont gratifié de nombreuses proches, collègues et amis.

Je voudrais tout d'abord remercier la société Streamlined Solutions and Services pour m'avoir donné l'opportunité de réaliser ce projet, au départ une initiative personnelle, dans le cadre de ses infrastructures techniques. Un grand merci en particulier à Frederic Schmitt pour le support technique et logistique apporté tout au long du projet.

Tous mes remerciements également à mon promoteur, le professeur B. Le Charlier, qui m'a guidé par ses précieux conseils et judicieuses remarques lors de la rédactions de ce document.

Merci encore à toute ma famille pour le soutient que tous m'ont offert pendant les moments difficiles, et pour la relecture qu'ils ont faite de ce document. Assurément sans vous rien n'aurait été possible ! Je veux tout particulièrement remercier ma grand-mère qui a passé des heures à traquer les fautes de frappe et d'orthographe. Merci.

Et à toi, Gabriela, qui m'a donné la force de mener ce projet à son terme, je dédie ce travail.

# Table des matières

# Table des figures

# Glossaire

**ADSL** **A**synchronous **D**igital **S**ubscriber **L**ine. Protocole d'accès couramment utilisé pour les connexions individuelles à L'Internet. Comme son nom l'indique, le débit de ce type de connexion n'est pas identique dans les deux sens, l'utilisateur aura une meilleure bande passante pour les données qui viennent vers lui (et donc une plus grande vitesse de téléchargement). Ce type de moyen d'accès est disponible à un prix fixe par mois et pour une quantité de données transférées forfaitaire importante.

**ARP** **A**ddress **R**esolution **P**rotocol. Protocole permettant de connaître l'adresse de niveau physique d'une machine à partir de son adresse IP.

**ATM** **A**synchronous **T**ransfer **M**ode. Technologie de réseau basée sur le transfert de données dans des paquets très petits et de taille fixe, et sur l'établissement de circuits virtuels entre les machines en communication, avec éventuellement une garantie de service. Cette technologie est surtout utilisée pour les réseaux à haut débit, entre les fournisseurs d'accès à l'Internet par exemple.

**BGP** **B**order **G**ateway **P**rotocol. Protocole Internet permettant à un groupe de routeurs de partager les informations de routage de manière à pouvoir établir des chemins de routage efficaces et sans boucle. BGP est souvent utilisé par les routeurs des fournisseurs d'accès à L'Internet. Ce protocole est définit dans le RFC 1771.

**CA (Certification Authority)** Autorité de Certification. Entité dont le rôle est de garantir l'authenticité et la validité des certificats employés au sein d'une communauté d'utilisateurs.

**CAST** Algorithme de chiffrement par bloc, utilisant des blocs de 64 bits et une clef de chiffrement de 64 bits de long. Plus de détails sur cet algorithme sont disponibles dans le chapitre 14 de [1].

**Checksum** Valeur calculée en appliquant une fonction mathématique à un ensemble de données, dont le but est de vérifier si ces données n'ont pas été modifiées par une altération du support qui les contient.

**DES, 3DES** **D**ata **E**ncryption **S**tandard et triple Data Encryption Standard. Algorithme standard de chiffrement de données du gouvernement américain pendant plus de 20 ans. Il s'agit d'un algorithme de chiffrement par bloc de 64 bits, utilisant une clef de 56 bits. Le chapitre 12 de [1] est entièrement consacré à cet algorithme. 3DES est une variante du même algorithme qui consiste à appliquer 3 fois le processus de chiffrement avec des clefs qui sont dérivées au cours des itérations. Vous trouverez également les détails dans la références citée plus haut.

**DMZ** **Dem**ilitarized **Z**one, en français Zone Démilitarisée. Terme employé pour désigner un sous-réseau d'une organisation dont les accès sont contrôlés par un firewall, et auquel seront connectés les serveurs ayant des interactions avec l'Internet. Le firewall contrôle les accès entre la DMZ et l'Internet, et aussi entre la DMZ et le réseau interne de l'organisation. Cette zone joue un rôle similaire à celui de la zone située entre les deux murailles d'une fortification : si l'ennemi parvient à y pénétrer, il n'est pas encore à l'intérieur de la forteresse (c'est-à-dire du réseau interne).

**Encapsulation, dé-encapsulation** Mécanisme utilisé au cours du traitement par les protocoles formant les couches successives de la pile TCP/IP, consistant à envelopper les données reçues de la couche immédiatement supérieure, en plaçant une entête au début

de ce qui constituera l'unité de transfert du protocole qui effectue cette encapsulation. Le processus inverse aura lieu à la réception d'une unité de transfert en provenance de la couche inférieure : les données sont "déballées" (dé-encapsulées) en enlevant l'entête propre au protocole. Le mécanisme d'encapsulation est expliqué dans l'introduction à TCP/IP dans le chapitre 1 de ce document.

**Firewall** Passerelle ayant des capacités de filtrage du trafic qui passe par elle, utilisée pour contrôler le connexions entre deux réseaux. Le filtrage peut se faire sur base des adresses IP et des ports indiqués dans les entêtes de paquets. Dans la plupart des cas les règles spécifient quel trafic est autorisé, tout autre trafic est bloqué. La plupart des firewalls peuvent effectuer des conversions d'adresse, ce qui permet de rendre une machine sur une DMZ (utilisant une adresse privée) accessible de l'extérieur via une adresse IP publique du firewall.

**Hash** D'après la définition donnée dans [1], une fonction de hashage (en anglais Hash Function) est "une fonction, mathématique ou autre, qui prend une chaîne d'entrée de longueur variable et la convertit en une chaîne de longueur fixe (en général plus petite)". La chaîne de résultat est appelée valeur de hash. Les fonction de hashage utilisées en cryptographie ont la propriété qu'il est facile de générer une valeur de hash à partir d'une entrée, mais qu'il est très difficile de récupérer la chaîne d'entrée correspondant à une valeur de hash donnée.

**ICMP** **I**nternet **C**ontrol **M**essage **P**rotocol. Extension du protocole IP définit dans le RFC 792. ICMP permet l'échange de messages informatifs, de contrôle et de diagnostic permettant de gérer les situations exceptionnelles au cours d'une communication basée sur IP. La commande de test de connexion Ping est l'exemple le plus connu d'utilisation d'ICMP.

**IETF** **I**nternet **E**ngineering **T**ask **F**orce. Organisation de standardisation principale de l'Internet. L'IETF est une grande communauté internationale composée d'ingénieurs réseau, d'opérateurs, de vendeurs et de chercheurs soucieux de l'évolution et du bon fonctionnement de l'Internet. L'IETF délivre des standards connus sous le nom de RFC (pour Request For Comments) qui décrivent tous les protocoles et services qui assurent le fonctionnement de l'Internet.

**IGRP** **I**nterior **G**ateway **R**outing **P**rotocol. Protocole de routage développé par Cisco System, et basé sur l'algorithme de routage par vecteur de distance. Une description assez complète de ce protocole est disponible dans [2].

**Implémenter, implémentation** Terme du jargon informatique, dérivé de l'anglais "to implement", qui signifie "mettre en oeuvre", "développer une solution". Une implémentation est une mise en oeuvre d'un concept ou d'une solution.

**IP** **I**nternet **P**rotocol. Protocole de la couche réseau, utilisé sur l'Internet. Un description de ce protocole est disponible dans la première partie du chapitre 1 de cet ouvrage.

**IPsec** **I**nternet **P**rotocol **Sec**urity. Suite de protocoles intégrée à la couche IP, dont le but est d'assurer l'authentification, l'intégrité et éventuellement la confidentialité des données transmises via les protocoles TCP/IP. Ces protocoles seront décris au long de ce document.

**ISDN** **I**ntegrated **S**ervices **D**igital **N**etwork. Un standard de communication international pour la transmission de son, d'image et de données sur une ligne téléphonique numérique utilisant les câbles standards. ISDN supporte les transferts de donnée à un débit de 64

Kbps (64000 bits par secondes). La plupart des lignes ISDN offertes par les compagnies téléphoniques incluent deux lignes en une, les canaux B. Il est possible d'utiliser les deux canaux indépendamment, ou de les grouper pour obtenir un débit de donnée de 128 Kbps, ce qui donne une connexion plus rapide qu'avec un modem sur une ligne téléphonique classique.

**LAN** **L**ocal **A**rea **N**etwork. Reseau local. On parle en général d'un réseau local pour désigner un réseau dont l'étendue est limitée à un batiment.

**LDAP** **L**ightweight **D**irectory **A**ccess **P**rotocol. Ce protocole est une version allégée du protocole décrit dans la norme X.500 pour l'accès aux répertoires d'information. LDAP est un protocole qui supporte TCP/IP. Parfois aussi appelé X.500-lite. Les répertoires accessibles par LDAP peuvent contenir différents types d'information, l'exemple le plus courant étant les adresses email et les certificats.

**MTU** **M**aximum **T**ransmission **U**nit. La plus grande taille d'un paquet physique, mesurée en bytes (octets) qu'un réseau peut transmettre. Tout message dont la taille excède le MTU est divisé en paquets plus petits avant d'être envoyé. Chaque réseau a un MTU différent qui peut être configuré par l'administrateur, chaque type de réseau est caractérisé par un MTU typique.

**OSPF** **O**pen **S**hortest **P**ath **F**irst. Protocole de routage développé pour les réseaux IP, basé sur l'algorithme d'état de lien (link-state algorithm). Il a pour caractéristique de faire converger le routage du réseau vers un état stable rapidement en cas d'évènements comme une congestion d'une portion du réseau. Il permet ainsi de résoudre rapidement les problèmes de bouclage. Ce protocole succède à RIP comme le protocole de routage de l'Internet. Il est décrit dans le document RFC 1583.

**Passerelle** Noeud d'un réseau qui sert de point d'entrée vers un autre réseau. Sur un réseau d'entreprise la passerelle reliant le réseau local interne à l'Internet agira également comme un firewall et filtrera le trafic entre l'intérieur et l'Internet. Une passerelle pourra aussi jouer le rôle de proxy, en adressant les requêtes pour un service donné vers l'extérieur pour le compte des machines sur le réseau interne, afin de protéger ces dernières des attaques extérieures et des contenus potentiellement dangereux (virus).

**Perfect Forward Secrecy** Propriété des négociations de clefs successives au cours de la vie d'une connection de type IPsec, selon laquelle les clefs successivement échangées sont indépendantes l'une de l'autre. De cette façon si une clef est "cassée" par un attaquant, les clefs suivantes ne sont pas compromises, c'est-à-dire que l'attaquant ne pourra pas déduire les clefs suivantes de la clefs qu'il a cassé. Cette propriété fait partie des paramètres de configuration du protocole d'échange de clef d'IPsec.

**PKI** **P**ublic **K**ey **I**nfrastructure. Infrastructure dont le but est de permettre l'utilisation de certificats numériques pour authentifier et protéger des échanges de données. Afin de garantir la validité et l'authenticité des certificats auprès de tous les participants d'un tel système, une autorité centrale servira de référence et sera chargée d'émettre les certificats pour tous. le système de clef publique permettra à chaque participant de vérifier qu'un certificat a bien été émis par l'autorité centrale, qu'il pourra alors utiliser en toute confiance.

**PMTU** **P**ath **M**aximum **T**ransmission **U**nit. Concept identique à celui de MTU, mais appliqué au chemin IP complet reliant deux hôtes. Le PMTU est la plus grande taille d'un

paquet IP pouvant être transmis vers un hôte sans qu'il y ait besoin de fragmenter ce paquet.

**RIP** **R**outing **I**nf**o**rmation **P**rotocol. Protocole de routage décrit dans le RFC 1058, qui spécifie comment les routeurs échangent leurs informations de routage. Avec RIP, les routeurs échangent périodiquement l'entièreté de leur table de routage. Cet algorithme manque d'efficacité et est rapidement remplacé par le protocole OSPF (voir définition plus haut).

**Root Certificate** Certificat de l'autorité de certification dans une infrastructure de clef publique (PKI). Un root certificate permet de vérifier qu'un certificat a bien été émis par l'autorité de certification et qu'il est donc digne de confiance.

**RSA** Nommé d'après les initials de ses auteurs Ron Rivest, Adi Shamir et Leonard Adelman, sans aucun doute le plus célèbre algorithme de chiffrement à clef publique. Il repose sur l'utilisation de l'arithmétique modulaire sur des nombres premiers de très grande taille. RSA peut être utilisé pour le chiffrement et la signature numérique. Le lecteur intéressé trouvera une description complète de cet algorithme au chapitre 19 de [1].

**SHA** **S**ecure **H**ash **A**lgorithm. Algorithme de hash conçue par l'agence nationale de sécurité américaine (NSA), et produit des valeurs de hash de 160 bits de long. Une description de cet algorithme est disponible au chapitre 18 de [1].

**SSH** **S**ecure **SH**ell. Application de connexion sécurisée à distance, reposant sur l'utilisation de SSL. Cette application permet l'administration d'une machine à distance et le transfert de fichiers de manière sécurisée à travers un réseau public.

**SSL** **S**ecure **S**ocket **L**ayer. Protocole se positionnant au-dessus de TCP/IP, permettant de sécuriser des connexions à travers l'Internet. Ce protocole est très utilisé dans le domaîne du commerce électronique sur l'Internet. L'inconvénient de ce protocole est que les applications doivent être conçues ou modifiées spécifiquement pour utiliser SSL.

**TCP** **T**ransmission **C**ontrol **P**rotocol. Protocole de la suite TCP/IP assurant un service de transmission des données orienté connexion, avec contrôle de flux et récupération des erreurs par retransmission. TCP assure un service dit fiable. Le lecteur trouvera plus de détails sur TCP dans le chapitre 1 de cet ouvrage (page 6).

**tcpdump** Utilitaire réseau permettant de capturer et d'observer le trafic TCP/IP sur une interface réseau. Cet utilitaire est un outil de diagnostic et d'apprentissage précieux. Il fait partie d'une famille de programmes réseaux appelés "sniffers". D'autres logiciels de ce type existent pour différents systèmes, tcpdump est le plus connu.

**UDP** User Datagram Protocol. Protocole de la suite TCP/IP assurant un service minimal de transport des données. UDP assure un service dit non-fiable. Le lecteur trouvera plus de détails sur UDP dans le chapitre 1 de cet ouvrage (page 1.1.1).

**VPN** Virtual Private Network. Terme désignant les techniques de connexion permettant de relier plusieurs hôtes ou réseaux locaux de manière sécurisée à travers des liaisons publiques comme l'Internet. Ces VPNs reposent sur l'utilisation d'algorithmes cryptographiques telles qu'IPsec pour protéger les échanges de données et n'autoriser l'accès que par les utilisateurs authentifiés.

**X.500** Norme définissant une structure hiérarchique pour les répertoires d'inf**o**rmation globaux. Cette structure est découpée par niveau, suivant différents types d'information comme le pays, l'état, la ville,... Il est ainsi possible d'accéder à l'inf**o**rmation en donnant sont chemin dans la hiérarchie.

**X.509** Standard définissant un format pour les certificats numériques. Cette définition a toujours le titre de recommandation par l'ITU (International Communication Union), et n'a pas le statut officiel de standard. La conséquence est qu'il existe différentes variantes de mise en oeuvre de cette recommandation, sous forme d'extensions qui ne sont pas toutes supportées par toutes les implémentations existantes. X.509 est malgré cela la norme la plus utilisée pour les certificats numériques.

# Introduction

L'Internet est de nos jours un des moyens de communication les plus importants et est omniprésent. Il permet à tout un chacun d'accéder à une masse considérable d'informations et d'échanger avec le monde entier. Il est aussi beaucoup utilisé dans le monde des affaires, pour faire du commerce, de la publicité, pour échanger des informations critiques, pour efffectuer des transactions financières. De plus en plus d'échanges de données sur l'Internet sont sensibles, et doivent être protégés des utilisateurs indiscrets ou mal-honnêtes.

Les protocoles TCP/IP, qui furent mis au point il y a maintenant plus de 20 ans à la naissance de ce qui allait devenir l'Internet moderne, n'ont pas été conçus avec la sécurité pour objectif, car au début l'Internet était un réseau militaire fermé. Hors les choses ont changé, mais ces protocoles du début sont toujours là, ce sont eux qui gèrent tout le trafic du réseau des réseaux. Ces protocoles ne répondent pas aux besoins actuels en matière de sécurité, à savoir principalement le besoin de confidentialité et d'intégrité des données échangées.

Plusieurs solutions ont vu le jour pour tenter de remédier à cela. Le problème de la plupart d'entre elles est leur manque d'intégration réelle aux protocoles TCP/IP. Cela a pour conséquence un manque de transparence à l'utilisation, et surtout la nécessité de modifier les applications existantes pour bénéficier des protections offertes par ces solutions.

IPsec est l'aboutissement de travaux des chercheurs de l'Internet Engineering Task Force, dont le but est de fournir une solution adaptée et universelle aux besoins de sécurité modernes. IPsec est une solution entièrement intégrée au protocole IP, il fait d'ailleurs partie intégrante de la prochaine version du protocole IP, connue sous le nom d'IPv6. IPsec est cependant utilisable avec la version actuelle du protocole IP. Grâce à cette intégration IPsec permet à toute application de l'Internet de bénéficier du niveau de sécurité requis par les standards industriels actuels.

Ce document décrit la mise en oeuvre d'une infrastructure d'accès à distance sécurisé au réseau interne de la société de services informatiques Streamlined Solutions & Services (3S), basée à Luxembourg. Ayant pour origine une initiative personnelle, ce projet a pour but de réaliser une solution pour permettre l'accès par les employés et l'administrateur au réseau local à partir de leur machine personnelle connectée à l'Internet, tout en garantissant un niveau de sécurité le plus élevé possible. La solution qui sera décrite repose sur l'utilisation d'IPsec et d'un système d'authentification des utilisateurs grâce à des certificats. Nous verrons ensemble la problématique liée à la mise en place d'une telle solution dans un réseau d'entreprise, les choix adoptés pour l'implantation de la passerelle d'accès, ainsi que la configuration et la mise en place des différents éléments entrant en jeu. Nous verrons également les différents tests effectués pour valider la solution choisie, les problèmes rencontrés pendant ces tests et les solutions adoptées.

Dans un premier chapitre nous examinerons les concepts théoriques nécessaires à la compréhension du fonctionnement d'IPsec, ainsi que les spécifications des protocoles utilisés. Ce chapitre commencera par un rappel des protocoles TCP/IP et de leurs problèmes de sécurité. Ensuite nous introduirons IPsec en décrivant son fonctionnement général, et en soulignant quels sont les problèmes auxquels IPsec apporte une solution. Le reste du premier chapitre sera consacré à l'examen plus approfondi des différents protocoles d'IPsec, tels qu'ils sont décrits dans les documents de spécification (les RFCs).

Le chapitre 2 sera consacré à l'implémentation d'IPsec sur le système Unix OpenBSD. Nous verrons comment les différents concepts et protocoles d'IPsec sont mis en oeuvre sur la

plateforme utilisée dans notre projet.

C'est finalement dans le chapitre 3 que nous verrons les différentes étapes d'analyse du problème des connexions à distance au réseau de la compagnie 3S, puis de déploiement et de test de l'infrastructure IPsec.

En fonction de ses connaissances en matière de réseau et de protocoles de sécurité, le lecteur pourra commencer par la partie théorique sur TCP/IP ainsi que l'introduction à IPsec, qui lui expliquera les concepts nécessaires, ou bien passer directement au chapitre 3 et aborder l'étude de cas. Les descriptions détaillées des protocoles IPsec ainsi que le chapitre 2 pourront être consultés pour une connaissance plus approfondie ou à titre de référence.

# Chapitre 1

# Partie Théorique

Ce chapitre a pour but de donner au lecteur une base de connaissances sur les concepts et les mécanismes évoqués dans l'étude de cas. En fonction de son niveau le lecteur pourra commencer par l'introduction à la suite TCP/IP, ou passer directement à l'introduction sur IPsec à la page 16. La partie plus détaillée où sont décrits en détail les protocoles de la suite IPsec commence à la page 24. Le lecteur pourra la lire entièrement, ou s'y référer pendant la lecture du chapitre 3.

## 1.1 TCP/IP

IPsec est une suite de protocoles destinée à sécuriser les connexions utilisant les protocoles de la suite TCP/IP. Il semble donc opportun de commencer par un rappel des protocoles constituant TCP/IP.

### Historique

La suite de protocoles TCP/IP a vu le jour au début des années 1970, dans le cadre du projet ARPANET du Département de la Défense américaine (DOD). Un des objectifs de cette suite de protocoles était de pouvoir faire communiquer des systèmes hétérogènes, à une époque où chaque vendeur de matériel informatique fournissait aussi ses propres protocoles de communication. En permettant à ces équipements différents de communiquer entre eux par un "langage" commun, il sera possible de distribuer les différentes ressources informatiques qui jusque-là étaient souvent centralisées sur de gros systèmes. Un autre objectif était aussi de développer un réseau où même si certains noeuds de communications venaient à être coupés, il soit toujours possible de communiquer entre les systèmes toujours en fonction. Afin d'arriver à cet objectif il faut qu'il y ait toujours plusieurs chemins pour arriver d'un point à l'autre, ainsi qu'un mécanisme permettant de réagir aux changements de conditions sur le réseau pour acheminer les données par le chemin optimal. En cas de disparition d'un noeud du réseau, le trafic qui transitait par ce noeud devra être dirigé sur un chemin alternatif, de manière à ne pas interrompre son acheminement. De cette manière on obtient un réseau hétérogène et tolérant à la panne ou à la disparition d'une partie des noeuds de communication (en cas de conflit armé par exemple).

Nous allons maintenant voir comment les fonctionnalités pour remplir ces objectifs sont réparties entre les différents éléments de la suite TCP/IP.

### Architecture, rôle des différents protocoles

La suite de protocoles TCP/IP est découpée en couches, chacune des couches ayant un rôle précis et offrant des services à la couche qui lui est supérieure. Nous allons décrire ces couches en partant des couches supérieures qui sont plus proches des applications utilisant le réseau pour communiquer, et en descendant progressivement, en découvrant les aspects qui jusque-là étaient masqués car entièrement pris en charge par les couches inférieures.

Ces couches sont, en partant de la couche la plus haute :

**Application** Cette couche est composée par les applications qui font usage des protocoles TCP/IP pour communiquer avec d'autres applications. On y trouve les applications classiques, comme ftp, telnet,... Ces applications définissent chacune un protocole qui leur est propre pour communiquer entre client et serveur, c'est-à-dire les données à transmettre et le format dans lequel elles sont organisées. Cette organisation est propre à chaque application et n'influence pas le reste du traitement, les protocoles des couches inférieures les recevront comme une chaîne binaire à transmettre.

**Transport** Cette couche offre le service de transport des données de la couche application d'un hôte à l'autre. Afin de distinguer les différentes applications utilisatrices de la couche transport, chacune d'elle se voit attribuer un numéro encodé sur 16 bits. Ce numéro s'appelle un *port*. Les 1024 premiers numéros sont réservés aux applications standards du système, et sont appelés en anglais "well known ports" ; les numéros au-delà de 1023 sont utilisés par les autres applications. Quand une application est distribuée c'est une bonne idée de lui faire attribuer un numéro de port par l'IANA[1], de manière à éviter que deux applications différentes n'utilisent le même numéro par accident. Tous les numéros de port attribués sont disponibles dans [3], et se situent entre 1024 et 49151. Les numéros au delà sont utilisés par les applications utilisant un numéro dynamique, comme c'est le cas de la partie client qui utilise un port aléatoire pour ouvrir une connexion vers un numéro de port attribué au serveur.

Donc, la couche transport aura besoin pour remplir son rôle de recevoir en paramètres le numéro de port de source et de destination, les adresses de source et de destination (nous parlerons des adresses plus loin), et les données à transmettre.

La couche 'Transport est composée de deux protocoles, qui assurent un niveau de service différent : TCP (Transmission Control Protocol) et UDP (User Datagram Protocol). Ces deux protocoles seront expliqués un peu plus en détail plus loin, disons seulement maintenant que TCP assure un service de transport orienté connexion, avec garantie de transmission sans erreur ; UDP assure un service dit "best effort", sans connexion ni aucune garantie.

**Réseau** La couche réseau prend en charge plusieurs aspects. C'est elle qui prend en charge l'adressage, et le routage des informations jusqu'à leur destination. Elle joue un rôle d'abstraction des différents types de réseau qui peuvent être traversés, grâce au fait que chaque machine sur l'Internet doit avoir (au moins théoriquement) une adresse unique. Le protocole qui assure les services de la couche réseau est IP (Internet Protocol). C'est un protocole de commutation de paquets : les données sont découpées en paquets qui seront traités individuellement. Il reçoit en paramètres les adresses de source et de destination et les données à transmettre déjà traitées par la couche de transport.

[1]Internet Authority for Number Attribution

IP gère aussi la fragmentation et le ré-assemblage des paquets. En résumé, la fragmentation arrive lorsque l'un des réseaux traversés supporte une taille de bloc de donnée maximum inférieure à la taille du paquet IP. Dans ce cas, la couche IP de la machine connectée à ce réseau fragmente le paquet en plusieurs morceaux assez petits pour passer sur ce réseau. A la fin les morceaux seront ré-assemblés, et les données reçues par la couche transport de destination seront identiques (sauf erreur de transmission) à celles qui furent confiées à IP par la machine expéditrice.

**couche physique** Cette couche gère l'accès au réseau physique auquel la machine est reliée. Par "réseau physique", on entend vraiment le réseau matériel, les câbles et les différents équipements qui le compose, ainsi que le protocole qui régit les signaux électriques ou lumineux qui servent à la transmission d'informations binaires. Il existe beaucoup de types de réseaux physiques différents, et c'est l'interconnexion de ces réseaux physiques qui forme un grand réseau que l'on pourrait qualifié de "logique", et qui est lui géré par la couche Réseau de l'architecture TCP/IP.

Le cas de réseau physique le plus connu est Ethernet pour les réseaux locaux, mais on trouve aussi Token-Ring (de moins en moins), FDDI (réseau en fibre optique), X.25, ATM, .... On assume que ces protocoles sont non-fiables, c'est-à-dire qu'on ne compte pas sur eux pour garantir une transmission de données sans erreur (car tous ne le font pas). Ce sera donc aux couches supérieures d'assurer la fiabilité des transmissions si nécessaire.

Nous n'en dirons pas plus, contentons-nous de savoir que ces réseaux assurent la transmission des données sur un lien local entre deux noeuds du réseau, et que cette couche est complètement abstraite par la couche réseau IP.

Ces quatre couches sont illustrées par la figure 1.1.

### Encapsulation

Le traitement des données par les couches successives de la pile TCP/IP se fait par *encapsulation* : le protocole de chaque couche ajoute une entête avec les informations nécessaires à la transmission et au traitement par la machine réceptrice devant les informations reçues de la couche supérieure dans le cas de données sortantes. Les données venant de l'application sont envoyées à TCP (par exemple) qui ajoute une entête contenant les paramètres qui lui sont propres (les numéros de port entre autres) à ces données. TCP passe ensuite le tout ainsi que les paramètres nécessaires à IP, qui à son tour ajoute une entête contenant les adresses de source et de destination.

A la réception, c'est le processus inverse qui a lieu : chaque couche reçoit un ensemble de données constitué de son entête et des données encapsulées par son homologue sur la machine émettrice. L'entête est alors traitée et enlevée, et ce sont les données une fois "déballées" qui sont passées à la couche supérieure. La figure 1.2 permet de mieux comprendre le concept.

Voyons maintenant un peu plus en détail les principaux protocoles de la suite, à savoir UDP, TCP et IP.

## 1.1.1 UDP

UDP est le protocole de la couche transport le plus simple. Il n'assure pas que la transmission se passe sans erreur, ni aucune retransmission automatique. On pourrait dire de manière

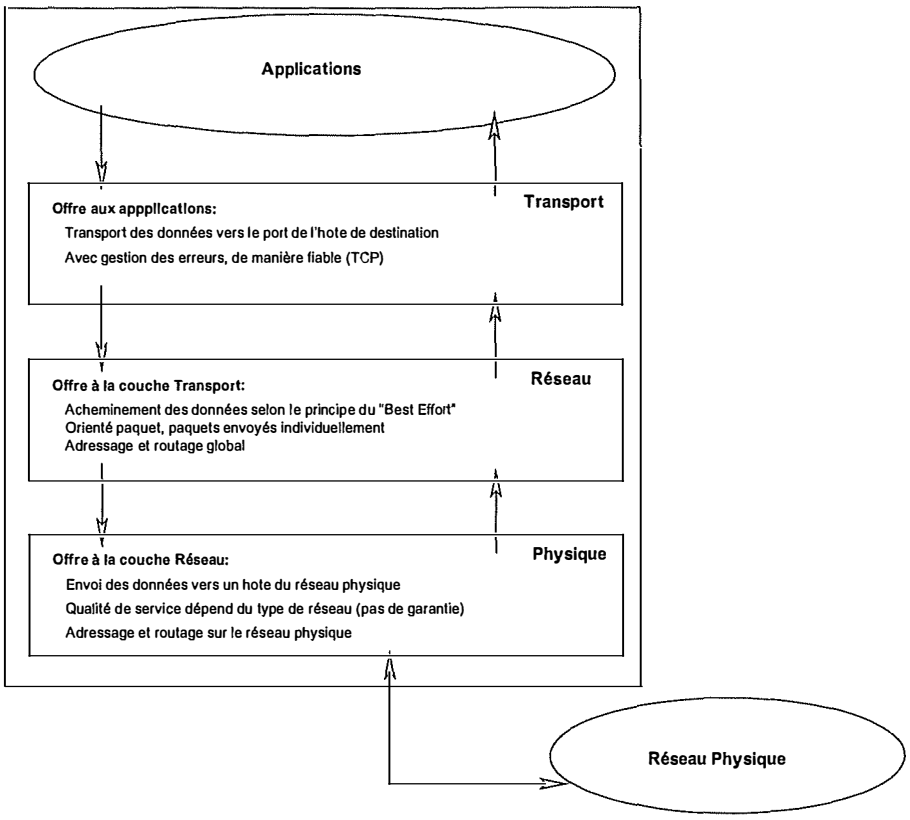



FIG. 1.1 – Couches de l'architecture TCP/IP, rôle et interactions

simplifiée que UDP ne fait qu'ajouter la notion de port à la couche réseau. Les applications utilisant ce protocole doivent donc prendre en charge elles-même la gestion des erreurs et la retransmission si nécessaire. Une caractéristique importante d'UDP est qu'il n'est pas orienté connexion. Chaque datagramme (c'est le nom de l'unité de transmission UDP) est envoyé individuellement, sans qu'un contexte soit maintenu, il n' y a pas de négociation d'ouverture de la communication. On dit d'UDP qu'il offre un service de transport non-fiable. L'avantage d'UDP est sa légèreté et donc sa rapidité.

Parmi les applications utilisant UDP on trouve les serveurs DNS (Domain Name Service), le service de configuration dynamique DHCP,... Ainsi que le serveur IKE (Internet Key Exchange) utilisé par IPsec et dont nous parlerons dans ce document.

Le protocole UDP est décrit dans [4].

### 1.1.2 TCP

TCP offre un service de transport beaucoup plus élaboré, orienté connexion, avec contrôle et récupération d'erreur, acquittement[2] des données reçues, et gestion du flux des données. Ce protocole est décrit dans [5].

[2]Le mot utilisé en anglais est *Acknowledgement*, ce qui se traduirait plus exactement en français par "Accusé de réception". Cependant le terme *acquittement* est souvent utilisé dans la littérature technique francophone, nous utiliserons donc ce terme tout au long de ce document.

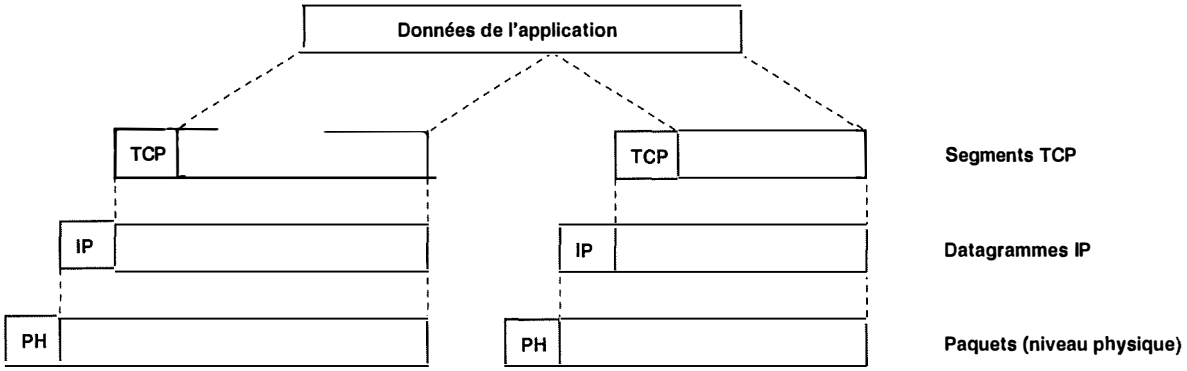



FIG. 1.2 – Mécanisme d'encapsulation

**Orienté connexion :** quand deux hôtes entrent en communication grâce à TCP il y a négociation et ouverture d'une connexion. Cette négociation se fait en trois étapes :

1. l'hôte A envoie une demande de connexion à l'hôte B. Une demande de connexion se caractérise par le drapeau SYN de l'entête TCP à 1.
2. l'hôte B répond par un segment acquittant la demande de A (drapeau ACK à 1) et demandant lui aussi l'ouverture d'une connexion (drapeau SYN à 1). Si B refuse la connexion, le drapeau SYN est à 0 et le drapeau RST (reset) est à 1, ce qui termine la tentative de connexion.
3. Si l'hôte A reçoit l'acquittement et la demande de B, il acquitte lui aussi cette demande en envoyant un segment portant le drapeau ACK à 1. La connexion est alors ouverte et la transmission des données peut commencer.

L'ouverture de connexion permet de synchroniser les numéros de séquences qui seront utilisés pour contrôler le flux pendant les transmissions. En effet chaque segment porte un numéro calculé à partir du numéro de séquence du paquet précédent et de la taille des données envoyées. Les numéros de séquence initials sont échangés lors de l'ouverture de la connexion.

La fermeture de la connexion se fera lorsque les deux parties auront signifié qu'elles n'ont plus de données à transmettre, par un segment ayant le drapeau FIN à 1. La fermeture se fait donc en deux étapes.

TCP devra donc maintenir l'état de chaque connexion, et gérer toutes les transitions d'état au cours de la vie de ces connexions, de leur ouverture jusqu'à leur fermeture.

**Contrôle et récupération d'erreur :** l'entête TCP contient une somme de contrôle (checksum) calculée sur les données et l'entête. Si à la réception du segment la somme reçue ne correspond plus à celle calculée à partir des données et de l'entête reçues, alors il y a eu une erreur durant la transmission. Le mécanisme d'acquittement permet de gérer la retransmission des paquets mal-transmis ou non-reçus. En effet chaque segment correctement reçu est acquitté par l'envoi d'un segment ayant le drapeau ACK à 1 et portant le numéro de séquence du dernier segment bien reçu. Si aucun acquittement n'est reçu pour un numéro de séquence donné (ou un numéro supérieur) au bout d'un certain temps, l'expéditeur retransmet le paquet (avec le même numéro de séquence). Le

délai d'attente avant retransmission devra être fixé de manière à ne pas avoir d'impact négatif sur les performances, et pourra être calculé sur base du temps aller-retour entre l'hôte et son interlocuteur.

**Gestion du flux :** Elle repose sur l'utilisation des numéros de séquence et d'une fenêtre de transmission coulissante.

Prenons l'exemple d'une connexion entre A et B. Chaque hôte maintient une fenêtre coulissante pour le trafic qu'il envoie, prenons l'exemple de celle de l'hôte A. La fenêtre coulissante est un tableau maintenu par A dont la taille est égale au nombre de segment que B accepte de recevoir d'un coup, et dont la borne inférieure est égale au numéro de séquence du dernier segment acquitté par B. il est à noter qu'un segment n'est acquitté par B que si les segments qui le précèdent ont été bien reçu ; donc tous les segments dont le numéro est avant le début de la fenêtre ont déjà été acquittés. la fenêtre est décalée ainsi au fur et à mesure des acquittements par B. A transmet des segments jusqu'à ce que la borne supérieure de la fenêtre soit atteinte, et arrête de transmettre si aucun acquittement ne provoque le décalage de la fenêtre.

De cette façon l'hôte récepteur indique à quel rythme il traite les paquets à la machine émettrice, et celle-ci adapte la vitesse d'expédition des segments en conséquence. La taille de la fenêtre est ajustée dynamiquement, chaque segment envoyé contenant le plus grand numéro de séquence que l'émetteur du segment est prêt à recevoir, ce qui permet de calculer le nombre d'octets maximum à envoyer.

### Format de l'entête TCP

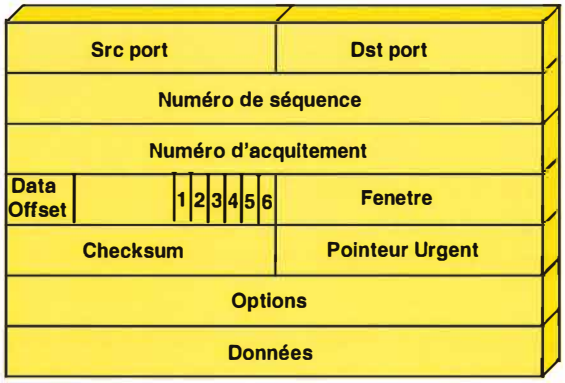



FIG. 1.3 – Entête TCP

Voyons les informations présentes dans une entête TCP, illustrée par la figure 1.3 :

**Source port :** indique le port de l'application expéditrice du paquet. Ce port sera utilisé comme port de destination par l'interlocuteur de la connexion.

**Destination_port :** indique le port de l'application sur la machine de destination.

**Numéro de séquence :** indique le numéro du premier octet de donnée du segment. A l'ouverture de la connexion un numéro de séquence initial est choisi, et est ensuite mis à jour en l'incrémentant du nombre d'octets de données envoyés dans chaque segment.

**Numéro d'acquittement :** utilisé dans le mécanisme d'acquittement des segments correctement reçus, ce champ indique le prochain numéro de séquence attendu, et signale à l'autre hôte que les segments dont le numéro précède celui-ci ont été bien reçus.

**Data Offset :** indique l'emplacement du début des données par rapport au début du segment (entête comprise).

**1-drapeau URG :** bit indiquant si le champ Pointeur Urgent (voir plus loin) est utilisé ou non.

**2-drapeau SYN :** bit de synchronisation, indiquant une demande d'ouverture de connexion.

**3-drapeau ACK :** bit indiquant que le champ d'acquittement est utilisé.

**4-drapeau RST :** bit indiquant à l'autre partie que la connexion doit être redémarrée.

**5-drapeau PSH :** bit indiquant que l'émetteur souhaite que les données de ce segment soient délivrées le plus tôt possible au destinataire.

**6-drapeau FIN :** bit indiquant que l'émetteur n'a plus de données à transmettre, et demande donc la fermeture de la connexion.

**Fenêtre :** indique, sur 16 bits, le nombre d'octets que la machine émettrice du segment accepte de recevoir. Ce champ est utilisé par le mécanisme de fenêtre coulissante décrit plus haut.

**Checksum :** Somme de contrôle calculée sur le contenu de l'entête et des données, permettant de détecter une éventuelle erreur de transmission.

**Pointeur Urgent :** indique l'emplacement de la fin du message urgent dans les données du segment.

**Options :** contient des options du protocole TCP, principalement des options de routage.

**Données :** Les données transportées par le segment.

### 1.1.3 IP

Le protocole IP a pour rôle d'acheminer chaque *paquet* vers sa destination dans l'esprit du "best effort", c'est-à-dire qu'il fera de son mieux pour y arriver mais si ça ne marche pas tant pis, il n'y a pas de garantie de résultat. IP est un protocole de commutation de paquets, les données reçues des couches supérieures sont encapsulées dans des paquets IP et chaque paquet est envoyé d'un noeud[3] à un autre, jusqu'à atteindre l'hôte de destination. C'est ce qu'on appelle le *routage*. Ce routage repose sur le fait que chaque noeud sait vers quel noeud suivant envoyer un paquet en fonction de sa destination, de telle manière que ce paquet arrive à bon port par le meilleur chemin possible.

La version actuellement utilisée est IPv4. C'est de celle-là que nous parlerons, mais nous mentionnerons les principaux changements entre IPv4 et le protocole IPv6 qui le remplacera dans les années à venir.

#### Adressage

Le système d'adressage IP prévoit que chaque machine, ou plus précisément chaque interface connectée au réseau doit avoir une adresse distincte et unique. Une adresse IP est une

[3]les noeuds du réseau sont les routeurs qui sont chargés de faire suivre les paquets vers leur destination.

chaîne de 32 bits. cette chaîne est habituellement représentée selon la notation décimale pointée, c'est-à-dire une suite de 4 nombres compris entre 0 et 255, correspondant aux 4 octets de l'adresse, chaque nombre étant séparé par un point.

Le réseau Internet est subdivisé en unités plus petites, appelées sous-réseaux. Un sous-réseau IP est formé d'un ensemble d'adresses partageant un préfixe commun. Une adresse IP peut donc être découpée en deux parties : la partie commune à toutes les adresses du sous-réseau (appelée "adresse de sous-réseau"), et la partie propre à chaque hôte qui identifie cet hôte sur le sous-réseau. Par exemple le bloc allant de 193.1.1.0 à 193.1.1.255 forme un sous-réseau de 256 adresses. La partie commune est composée des 24 premiers bits (les trois premiers nombre en notation décimale pointée), notés "193.1.1". Le dernier nombre des adresses identifie chaque machine de ce sous-réseau. Pour indiquer la zone commune des adresses d'un sous-réseau on utilise un masque binaire de la même taille qu'une adresse IP (32 bits). La zone de bits contigus la plus à gauche (bits de poids fort) dont la valeur est 1 indique la partie d'adresse du sous-réseau, la zone correspondant aux bits à 0 pourra varier afin de distinguer les machines sur le sous-réseau. La figure 1.4 permet de mieux visualiser le masque binaire. Il existe deux notations possibles du masque de sous-réseau :

- la notation décimale pointée. Dans l'exemple donné plus haut le masque du sous-réseau sera noté 255.255.255.0
- une notation dont le principe est d'indiqué la taille du masque en nombre de bits. Dans ce cas l'adresse du sous-réseau de notre exemple sera notée 193.1.1.0/24.

```
Adresse décimale:    193    .   1    .    1    .    1
Adresse binaire:   11000001 00000001 00000001 00000000
Masque binaire:    11111111 11111111 11111111 00000000
Masque décimal:      255    . 255    .   255    .    0
```

FIG. 1.4 – Adresse IP - masque et sous-réseau

En général, on regroupe sur un même sous-réseau un ensemble de machines connectées sur un ou plusieurs réseaux physiques, au sein duquel les hôtes communiquent directement sans qu'il y ait besoin de recourir aux services d'une machine intermédiaire pour la transmission des paquets.

Pour déterminer si une adresse fait partie d'un sous-réseau donné, on applique le masque de sous-réseau à l'adresse du sous-réseau et à l'adresse à vérifier, et on compare les deux résultats obtenus. Si ils sont identiques alors l'adresse fait bien partie du sous-réseau. L'application du masque se fait en appliquant et ET logique bit à bit entre une adresse et le masque, ce qui a pour résultat de préserver la valeur de la zone masquée de l'adresse (celle qui correspond aux bits à 1) et de mettre à zéro la partie non-masquée. Ainsi en comparant deux adresses auxquelles le même masque a été appliqué on peut voir si elles appartiennent au même sous-réseau.

A l'origine, il avait été décidé de définir des classes de sous-réseau, en fonction de la taille de la partie fixe et donc du nombre de machines que ce sous-réseau pourrait adresser. Voici les 3 classes principales :

- les sous-réseaux de classe A dont le masque a une taille de 8 bits, et qui peuvent donc adresser plus de 16 millions de machines ($2^{24}$). Le premier bit de ces adresses doit être à 0. Il y a donc 128 sous-réseaux de classe A possibles, dont 2 sont réservés.
- les sous-réseaux de classe B dont le masque a une taille de 16 bits, et qui peuvent adresser 65536 machines ($2^{16}$). Les deux premiers bits de ces adresses doivent avoir 10 pour valeur. Il y a donc 16384 sous-réseau de classe B possibles, dont 2 sont réservés.
- les sous-réseaux de classe C dont le masque a une taille de 24 bits et qui peuvent adresser 256 machines ($2^8$). Les trois premiers bits de ces adresses doivent avoir 110 pour valeur. Il y a donc 2'097'151 sous-réseaux de classe C possibles, dont 2 sont réservés.

Au départ l'attribution des adresses pour un sous-réseau se faisait en attribuant un bloc de la classe appropriée à la taille du sous-réseau à adresser. Cependant, vu l'explosion du nombre de machines connectées on s'est vite rendu compte que cette granularité n'était pas adaptée. De nos jours l'attribution d'un sous-réseau n'est pas limité à ces trois classes, et on rencontre donc des sous-réseaux avec des masques de 28 bits par exemple, qui peuvent adresser 16 machines.

L'explosion du nombre de machines connectées a également provoqué une pénurie d'adresses, les 32 bits ne suffisent plus pour fournir une adresse unique à toutes les interfaces connectées. Pour palier à cela on a recours à l'utilisation d'un artifice technique appelé *translation d'adresse (NAT,* pour *Network Adress Translation)* : le routeur qui connecte un sous-réseau privé à l'Internet se voit attribué une adresse unique, les machines qui se trouvent *derrière* ce routeur, sur le réseau privé, utilisent des adresses privées. toutes les communication entre les machines du réseau et l'Internet passent par le routeur, qui transforme l'adresse privée de chaque machine interne et la remplace par la sienne. De même, à la réception du flux venant de l'Internet pour une connexion avec une machine interne, le routeur remplacera l'adresse source (qui a pour valeur l'adresse du routeur) par celle de la machine interne. Le routeur effectue cette conversion grâce au fait qu'il mémorise les connexions ouvertes et à qui elles appartiennent. Ce mécanisme permet de n'allouer qu'une seule adresse pour tout un sous-réseau privé, économisant ainsi les adresses publiques devenues rares.

Mais ce n'est qu'une solution temporaire, qui ne fait que retarder l'échéance d'une pénurie totale. Pour remédier à cela le prochain standard du protocole IP, IPv6, utilisera des adresses d'une taille de 128 bits, ce qui à été estimé à environ 32 adresses par pouce-carré de surface terrestre sèche ! Cela devrait régler le problème de disponibilité des adresses pour longtemps. Le lecteur désireux d'en savoir plus sur IPv6 pourra consulter [6, 7].

### Routage

Comme nous l'avons vu plus haut, l'Internet est composé de réseaux physiques, qui forment soit seuls, soit en groupe des sous-réseaux dont les machines sont adressées dans un même bloc d'adresses contiguës. Deux machines voulant communiquer au sein d'un même sous-réseau le feront grâce à un mécanisme faisant correspondre les adresses du réseau physique et les adresses IP, le paquet sera ensuite transmit directement vers la machine ayant l'adresse physique correspondante, cette transmission est entièrement prise en charge par le protocole de la couche physique.

Une machine peut faire partie d'un ou plusieurs réseaux physiques, et donc d'un ou plusieurs sous-réseaux IP. Ce sera par exemple le cas d'une machine équipée de plusieurs interfaces réseau, chacune connectée à un réseau physique différent.

Prenons un exemple : la couche IP de la machine A doit envoyer un paquet vers une adresse IP x. Elle doit alors décider vers quelle interface physique envoyer le paquet, et vers quelle

machine. Plusieurs cas peuvent se présenter :

- L'adresse x fait partie d'un sous-réseau IP auquel A appartient aussi. Dans ce cas le paquet sera envoyé via l'interface physique correspondant à ce sous-réseau, vers la machine dont l'adresse physique correspond à l'adresse IP x[4].
- L'adresse x ne fait pas partie d'un sous-réseau IP auquel A appartient. Dans ce cas il faudra recourir à un intermédiaire pour transmettre le paquet vers le bon sous-réseau, et donc vers la machine. Afin de pouvoir déterminer quelle est cette machine intermédiaire, A possède une table, qui lui dit pour chaque sous-réseau avec lequel elle peut communiquer quelle est la machine vers laquelle envoyer les paquets destinés à ce sous-réseau. Cette machine doit bien entendu faire partie d'un sous-réseau auquel A appartient. La table contenant ces informations s'appelle la *table de routage*, et les entrées qu'elle contient sont *les règles de routage*, qui déterminent vers quel hôte envoyer le trafic pour les destinations possibles. La table de routage reprend en plus des règles spéciales qui indiquent quelle interface physique correspond à quelle sous-réseau IP.

Donc dans le cas où le sous-réseau correspondant à l'adresse de destination x figure dans cette table, A sait vers quelle machine envoyer les paquets. Mais dans le cas où le sous-réseau correspondant à l'adresse x ne se trouve pas dans la table, que faire ? Pour résoudre ce problème, il existe une entrée spéciale dans la table, qui indique la machine vers laquelle envoyer le trafic destiné au sous-réseau 0.0.0.0/0. Cette entrée est donc celle qui sera utilisée si aucune autre ne correspond, et s'appelle la *route par défaut.*

Ces règles de routage seront utilisées pour les paquets IP que les applications de A envoient.

Comme nous l'avons dit plus haut, une machine peut faire partie de plusieurs sous-réseaux IP, en particulier si elle comporte plusieurs interfaces physiques connectées à différents réseaux physiques. Si cette machine est configurée pour pouvoir transmettre des paquets qui ne lui sont pas destinés sur une interface et à les transmettre sur une autre interface vers sa destination, on dit que cette machine est une *passerelle.* Une passerelle dédiée est aussi appelée un *routeur.* Les passerelles jouent un rôle très important sur l'Internet, car elles relient les différents sous-réseaux IP pour constituer l'Internet.

Reprenons l'exemple du début, et supposons que A est en fait une passerelle reliant plusieurs sous-réseaux IP. Que se passe-t-il dans le cas où A reçoit un paquet venant du sous-réseau 1 et qui ne lui est pas directement destiné ? A va alors appliquer exactement le même comportement que si ce paquet venait d'une de ses applications, et consulter sa table de routage pour déterminer vers quel machine et via quelle interface envoyer ce paquet. Dans le cas où il n'existe pas de règle spécifique, le paquet sera envoyé vers la *passerelle par défaut*, indiquée par la route par défaut. L'action de faire suivre un paquet d'un sous-réseau à l'autre est appelée "forwarding" en anglais.

Donc, pour reprendre les choses d'un niveau plus global, un paquet dont la destination ne se trouve pas sur le même sous-réseau IP va être transmit à la passerelle adéquate grâce aux règles de routage de l'expéditeur, la dite passerelle fera à son tour suivre le paquet vers la passerelle suivante, et ainsi de suite, jusqu'à sa destination finale.

Bien sûr ce n'est pas si simple : il peut exister plusieurs chemins (routes) valables pour une même destination. Pour déterminer laquelle utiliser on donne un poids à chaque règle, qui représente le coût de son utilisation calculé en fonctions de différents critères (qualité du réseau physique correspondant par exemple). La passerelle choisira toujours la règle la moins

[4]la conversion entre adresse IP et adresse physique est faite grâce à un protocole spécialisé appelé ARP, pour Address Resolution Protocol

chère.

Un problème important qui se pose sur un réseau aussi vaste que l'Internet est la tenue à jour des tables de routage sur les différentes passerelles. En effet, si les règles sont mal établies on peut se retrouver dans des situations où les paquets tournent en boucle dans le réseau et n'atteignent jamais leur destination, parce qu'une ou plusieurs règles de routage sur une ou plusieurs passerelles ne sont pas correctes. Différents algorithmes associés à des protocoles spécialisés permettent de régler ce type de problème, en permettant aux routeurs de communiquer les changements de condition sur réseau, par exemple si un routeur ne répond plus, la passerelle qui s'en rend compte peut prévenir les routeurs adjacents qui vont augmenter le poids de la route correspondante et par la même occasion favoriser l'utilisation d'un chemin alternatif.

Pour le problème des boucles, plusieurs choses permettent d'éviter qu'un paquet ne tourne éternellement dans le réseau. D'une part le champ TTL (Time To Live) du paquet IP qui est décrémenté de 1 à chaque fois qu'il traverse un routeur, et qui provoque le rejet du paquet une fois arrivé à zéro. D'autre part là aussi des algorithmes permettent aux routeurs de détecter le problème et d'ajuster les poids des règles de routage en conséquence pour éviter les règles fautives.

Cette section ne donne qu'un bref aperçu du problème du routage, et du type de solution qui y est apporté. Une bonne introduction est disponible au chapitre 6 de [2].

### Fragmentation

Le réseau Internet est composé d'innombrables réseaux plus petits et de types différents. Il se peut que la taille maximum de l'unité de données sur un des réseaux empruntés par un paquet soit inférieure à la taille du paquet. Dans ce cas la couche IP du routeur à la frontière de ce réseau pourra éclater le paquet en plusieurs paquets plus petits qui pourront passer sur ce réseau. Chaque paquet contiendra alors les informations nécessaires pour pouvoir être ré-assemblé par la machine de destination finale. Ces informations sont contenue dans un drapeau spécifiant que le paquet original a été fragmenté (More Fragment flag), et l'emplacement du fragment dans le paquet original.

Il se peut que la machine de destination ne désire pas que le paquet soit fragmenté, par exemple pour des raisons d'efficacité car la fragmentation et le ré-assemblage prennent du temps. Elle positionnera alors le drapeau *Don't Fragment* (DF) à 1. Si un tel paquet se trouve dans la situation décrite plus haut, le routeur frontière le rejettera et enverra une notification à la machine source en précisant la taille maximum des paquets que le réseau peut supporter. La machine source pourra alors ajuster la taille de ses paquets, et ainsi éviter la fragmentation. La taille maximum des paquets supporté par l'ensemble des réseaux entre deux noeuds s'appelle *PMTU* (pour Path Maximum Transfer Unit). Une technique couramment utilisée consiste à découvrir la PMTU au moment de l'établissement de la connexion, en envoyant un paquet avec le drapeau DF à 1, et en réessayant si le paquet n'arrive pas jusqu'au bout en réduisant progressivement sa taille. Cette technique est appelée en anglais *PMTU discovery.*

La figure 1.5 illustre le fonctionnement de l'ensemble. Ce schéma représente deux hôtes connectés à travers une passerelle grâce à TCP/IP. Le trait bleu en pointillé représente le parcours des données à travers la pile réseau de A, en partant du port d'une application qui fait appel à TCP pour le transport de ses données. TCP appelle ensuite la couche IP pour l'acheminement des données à travers l'Internet. La couche IP va à son tour faire appel

aux services de la couche physique juste en-dessous dans la pile. La couche physique gère le transfert sur le réseau local jusqu'à la passerelle X. La couche IP de celle-ci reçoit les paquets IP, et à l'issue de l'opération de routage fait suivre le paquet sur le réseau physique 2 utilisant le protocole 2. L'hôte de destination B reçoit les paquets transmis par le réseau physique, et la couche physique transmet les données qui étaient contenues dans le paquet du protocole 2 à la couche IP de B. Celle-ci vérifie s'il n'y a pas eu d'erreur, dé-encapsule le segment TCP et le transmet à la couche de transport. Le segment est alors dé-encapsulé pour récupérer les données de l'application, qui sont transmises à l'application dont le port correspond à celui de destination du segment.

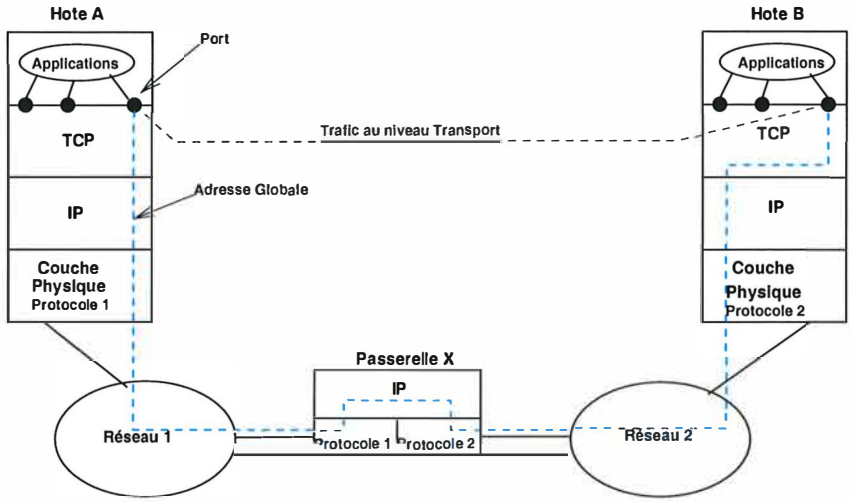



FIG. 1.5 – Communication grâce à TCP/IP

### 1.1.4 Protocoles Annexes

Il existe une série de protocoles attachés à la suite TCP/IP dont nous ne parlerons pas ici, mais qui jouent un rôle essentiel dans le fonctionnement de l'Internet. Citons les principaux, avec une courte description de leur rôle :

**ARP** Address Resolution Protocol, permet de faire le lien entre les adresses IP et les adresses physiques sur un réseau local Ethernet. Le protocole ARP est décrit dans [8]. Il existe des protocoles assurant les mêmes fonctionnalités que ARP pour d'autres types de réseau physiques qu'Ethernet.

**ICMP** Internet Control Message Protocol, définit un ensemble de messages utilisés pour gérer les situations spéciales sur le réseau, comme par exemple un hôte ou un sous-réseau inaccessible ou une congestion, un problème lié à la fragmentation,... Le protocole ICMP est également utilisé par l'utilitaire de diagnostic *ping*. Le protocole ICMP est décrit dans [9].

**RIP, IGRP, OSPF, BGP** sont différents protocoles de routage ayant différentes caractéristiques et basés sur différents algorithmes pour maintenir les tables de routage et éviter les boucles. Une description individuelle de chacun de ces acronymes figure dans le glossaire au début de cet ouvrage.

**DNS** Domain Name System, une infrastructure de l'Internet qui traduit les noms de domain en adresse IP, rendant l'utilisation de l'Internet plus facile. Un grand nombre d'informations sur le sujet peut être trouvé à partir du site http ://www.ihac.org/dns-refs/.

### 1.1.5 Problèmes de sécurité de TCP/IP

Comme nous l'avons vu au début, la sécurité ne faisait pas partie du cahier des charges lors de la conception initiale de TCP/IP, tout simplement parce que ce protocole était destiné à être utilisé sur un réseau militaire fermé, et pas sur un réseau mondial accessible par tout le monde. La sécurité résidait alors dans l'inaccessibilité au réseau, le protocole ne devait en principe rien avoir à faire à ce niveau-là. Mais TCP/IP qui fut conçu au départ comme une solution transitoire entre les protocoles propriétaires de l'époque et d'autres protocoles internationaux (les protocoles OSI) est devenu le standard international, depuis plus de 25 ans.

Les principaux problèmes de sécurité que l'on peut rencontrer sur l'Internet sont les suivants :

1. Attaques par déni de service (Denial Of Service, DOS) : consistent à occuper toutes les ressources d'un service sur le réseau, par un nombre important de fausses demandes de connexion par exemple. La conséquence est que les utilisateurs légitimes ne peuvent plus accéder au service attaqué. Ces attaques sont faciles, il est simple de fabriquer un grand nombre de paquets avec une adresse source IP aléatoire, le routage ne vérifie pas si un paquet vient effectivement de la machine ayant l'adresse figurant dans le paquet. De nos jours on a vu apparaître des attaques en déni de service distribuées, où l'attaquant commande un grand nombre de clients installés sur des machines partout sur le réseau (souvent à l'insu de leur propriétaire), déclenchant ainsi un véritable raz-de-marée de requêtes de connexion venant de toutes les directions en même temps, et qu'il est très difficile de stopper.
2. Espionnage sur le réseau (en anglais Eavesdropping) : consiste à examiner les données transmises sur le réseau à partir d'un des noeuds de routage, ou en détournant le routage vers une autre machine. Comme les données sont transmises sans traitement particulier destiné à rendre leur contenu secret (sauf si cela est fait au niveau applicatif), il suffit d'un programme comme tcpdump pour récupérer les paquets échangés et ensuite en extraire le contenu.
3. Attaque par modification des données : de la même façon que l'espionnage, il est possible de ne pas se contenter d'espionner, mais aussi de modifier le contenu des données des paquets. La somme de contrôle est facilement recalculée, et donc le destinataire n'y verra que du feu. Une attaque basée sur ce principe est l'attaque dite "man-in-the-middle" : l'attaquant se place entre le client et le serveur, et singe le comportement du serveur pour le client, et du client pour le serveur, de manière que ni l'un ni l'autre ne s'en rende compte.
4. Attaque de niveau applicatif : il est possible d'exploiter des failles dans les applications utilisant TCP/IP pour obtenir l'accès non-autorisé aux ressources du système sur lequel ces applications tournent. L'attaque de ce type le plus connu s'appelle en anglais "buffer-overrun" ou "stack smashing" et consiste à exploiter un défaut de vérification des paramètres d'entrée d'une application pour faire exécuter des instructions arbitraires sur le système, comme par exemple obtenir un shell avec les droits de l'utilisateur sous lequel

l'application tourne. Ce type d'attaque est possible car TCP/IP n'assure aucune authentification, chaque application doit s'en occuper elle-même, et il y a donc multiplication des mécanismes et donc des risques de bug.

5. Attaques basées sur l'utilisation du routage par la source : Ce type d'attaque consiste à exploiter une possibilité offerte par IP de fournir avec chaque paquet la liste des noeuds par lesquels les paquets contenant le trafic de réponse devra passer. Cela peut être utilisé par un attaquant pour se faire passer pour une machine connue de la cible, en utilisant l'adresse IP de cette machine connue et en interceptant les réponses dont l'attaquant a forcé la route pour qu'elles passent par lui. L'attaquant aura éventuellement assuré le silence de la machine pour laquelle il se fait passé grâce à une attaque de type déni de service décrite plus haut.

Ces attaques ainsi que quelques autres sont bien expliquées dans [10].

Nous verrons plus loin comment IPsec peut résoudre une partie de ces problèmes.

## 1.2 IPsec : Introduction

La suite de protocoles TCP/IP que nous avons décrite dans la partie précédente est utilisée depuis plus de 20 ans maintenant, et est à la base du fonctionnement de l'Internet. Du réseau militaire et de recherche qu'il était à ses débuts, l'Internet a connu une explosion en terme du nombre de machines connectées dans les années 90, et compte maintenant plusieurs dizaines de millions d'hôtes. Le public ayant accès au réseau s'étend maintenant à monsieur tout-le-monde, et les applications utilisant le réseau sont très variées, du simple échange de fichier et de courrier éléctronique, à la possibilité d'effectuer des transactions financières ou des achats avec une carte de crédit. De nombreuses compagnies utilisent l'Internet pour communiquer entre leurs différents sièges.

Tous ces changements ont bien vite fait apparaître des besoins critiques en matière de protection des données circulant sur l'Internet. Comme nous l'avons vu précédemment la sécurité n'a pas fait partie des critères de conception des protocoles de la suite TCP/IP. Un texte envoyé sans précaution particulière sur l'Internet pourrait être lu par quiconque étant en mesure d'intercepter le flux de paquets qui véhiculent ce message, voir même d'en altérer le contenu sans que le destinataire soit en mesure de détecter la fraude. Et même plus : le destinataire n'a pas de moyen fiable de s'assurer que l'information qu'il reçoit provient bien de la source indiquée. L'envoi d'un courrier éléctronique est équivalent en terme de confidentialité à l'envoi d'un message sur carte postale, dans un monde où n'importe qui pourrait entrer plus ou moins librement dans les centres de tri postaux.

De nombreuses solutions techniques ont vu le jour pour répondre à ces problèmes. Le protocole SSL[5] est l'une des plus répandues, cette couche qui se situe en amont de la pile TCP/IP permet d'assurer la confidentialité des données d'une application, tout en offrant un service d'authentification et de vérification de l'intégrité des données. Les applications utilisant SSL sont nombreuses : connexions à des sites Internet sécurisés par le protocole https, SSH[6], et même les connexions à certaines bases de données.

Les solutions de ce type, bien que fort efficaces, ont toutefois un inconvénient majeur : elles ne sont pas transparentes pour les applications qui les utilisent. Les applications doivent être conçues ou modifiées pour utiliser des protocoles comme SSL.

[5] Secure Socket Layer

[6] Secure SHell

Des recherches ont donc été faites sous la coordination de l'IETF[7] pour répondre à ces problèmes, et la suite de protocoles formant IPsec[8] est le fruit de ces travaux. IPsec offre les services de confidentialité, d'authentification et d'intégrité des données d'une connexion utilisant le protocole IP. Ces services sont assurés au niveau de la couche réseau (IP), et donc de manière transparente pour toutes les applications reposant sur IP. Les différents services peuvent être combinés selon les besoins, il est par exemple possible d'assurer l'intégrité des données d'une connexion sans offrir de confidentialité.

#### 1.2.0.1 Quelques définitions utiles

Avant de poursuivre nous allons donner quelques définitions de certains termes mentionnés dans le paragraphe précédent.

**Intégrité**

Le problème de l'intégrité des données numériques se pose dans divers situations. Si nous prenons le cas d'un programme informatique, le changement d'un seul bit des informations constitutant son code peut le rendre inutilisable. Ce genre de chose peut arrivé suite à une dégradation du support sur lequel le programme est stocké, la surface magnétique d'une disquette perdant ses propriétés avec le temps ou suite à l'exposition à un champ magnétique, par exemple. Dans le cas de transmission de donnée sur un réseau le problème peut avoir des origines diverses : corruption du signal due à des interférences éléctriques, ou altération accidentelle ou volontaire des données par un des noeuds du réseau. Ces modifications peuvent affecter aussi bien les données proprement dites que les informations propres aux protocoles de transmission. Pour détecter les cas d'erreurs accidentelles, des sommes de contrôle (checksum) sont calculées sur base de l'ensemble des données à vérifier. Les fonctions de calcul de ces checksums sont choisies de telle sorte que la modification d'un ou plusieurs bits dans la séquence des données modifie la somme. Ainsi la machine qui envoie les données y joindra le checksum qu'elle aura calculé, et le destinataire vérifiera que le calcul de cette somme sur les données reçue correspond bien au checksum reçu. Cette technique est utilisée par de nombreux système de compression et d'archivage de données, ainsi que par les protocoles TCP/IP pour détecter les erreurs de transmission dans les paquets.

Cette technique est suffisante pour détecter des erreurs de transmission accidentelles, mais qu'en est-il des modifications intentionnelles, par un tiers malveillant ayant pris le contrôle d'un noeud du réseau, pour modifier les données relatives à des transactions bancaires pendant leur transit du client vers sa banque par exemple ? Ce dernier pourrait très bien modifier les données ET la somme de contrôle. Dans le cas de l'utilisation de checksum ordinaire (telle que celle utilisée par TCP/IP), il est impossible de détecter la fraude.

L'utilisation d'une fonction de hashage avec clef permet de solutionner ce problème. L'idée générale est la même que pour le checksum : la fonction de hashage est utilisée pour calculer une valeur à partir des données dont on veut contrôler l'intégrité, avec comme paramètre supplémentaire une valeur secrète, la clef. Cette clef, connue seulement par l'expéditeur et le destinataire des données, est indispensable pour calculer la valeur de contrôle. Si un tiers malveillant tente de modifier les données, il ne pourra pas calculer une valeur de contrôle valide car il ne possède pas la clef. De même, il lui sera impossible de modifier les données de

[7] Internet Engineering Task Force
[8] IPsec est une contraction de IP Security.

telle sorte que la valeur de contrôle reste valide, car les fonctions de hash sont choisie pour leur propriété de dispersion des résultats dès la moindre altération des données d'entrée, et une très faible probabilité que deux chaînes d'entrées différentes donnent le même résultat (phénomène qu'on appelle "collision"). Le lecteur désireux d'en apprendre d'avantage au sujet des fonction de hashage pourra se référer à [1, 11].

La clef doit bien sûr être connue des deux parties et d'elles seules. Il existe plusieurs moyens de partager une clef sans en révéler la valeur au reste du monde. Soit par un simple échange de la main à la main, sur un support quelconque. Soit à travers le réseau grâce à l'utilisation de techniques cryptographiques comme l'algorithme d'échange de clef Diffie-Hellman.

Dans la définition de l'authentification, nous verrons aussi une autre technique de protection de l'intégrité basée sur l'utilisation de chiffrement à clef publique.

### Algorithmes de chiffrement symétriques vs. asymétriques

Un algorithme de chiffrement est dit symétrique si la clef qui sert au chiffrement et celle qui est utilisée pour le déchiffrement sont identiques. Il existe un grand nombres d'algorithmes symétriques, utilisant des longueurs de clef fixes ou variables, et fonctionnant selon différents modes, soit en traitant les données par blocs de taille fixe, soit en traitant le flux de donnée directement.

Un algorithme de chiffrement est dit asymétrique si la clef de chiffrement et celle de déchiffrement sont différentes. Les deux clefs sont générées ensemble, ce qui est chiffré avec l'une n'est déchiffrable qu'avec l'autre. L'une de ces clefs devient la clef privée, que son propriétaire gardera en sécurité, et l'autre devient la clef publique qui sera distribuée à tous ceux qui veulent communiquer avec le propriétaire de la clef privée. La clef publique servira à chiffrer les données que seul le propriétaire de la clef privée pourra déchiffrer. La clef privée pourra également être utilisée pour chiffrer le hash d'un ensemble de données que l'on veut authentifier ; la clef publique pourra alors être utilisée pour déchiffrer cette signature et ainsi établir que l'auteur est bien celui en possession de la clef privée. L'avantage de ce système est que la clef publique peut être transmise sans aucune précaution particulière, alors que la transmission de la clef pour un algorithme symétrique est critique.

Un fait très important à retenir est que les opérations de chiffrement et déchiffrement des algorithmes asymétriques sont beaucoup plus coûteuses en terme de puissance de calcul que celles des algorithmes symétriques. Il faut tenir compte de cela lors du choix d'une méthode de chiffrement.

### Authentification

Le problème de l'authentification des données consiste à s'assurer que l'auteur (ou l'expéditeur dans un contexte réseau) des données est bien celui que l'on suppose. Il peut s'agir d'un message reçu par courrier électronique ou, dans le cas qui nous préoccupe, des données contenues dans la séquence de paquets transmise dans le cadre d'une connexion réseau. Sur le réseau TCP/IP l'adresse source n'est pas une preuve d'origine fiable, car comme nous l'avons vu dans la section précédente il est possible de modifier l'adresse source en cours de transit, ou tout simplement d'envoyer des paquets avec une fausse adresse source.

La technique décrite dans la section sur l'intégrité peut servir à l'authentification, dans le cas ou deux parties sont les seules en possession d'une clef qu'elles utiliseront comme paramètre d'une fonction de hashage avec clef. Mais le partage de clef symétrique de manière sûre n'est

pas toujours possible. Un algorithme asymétrique peut alors être utilisé : l'expéditeur chiffre un hash simple des données avec sa clef privée, ce hash chiffré constituera la signature des données. Le destinataire, en possession de la clef publique de son correspondant, pourra alors comparer le hash déchiffré à l'aide de cette clef publique au hash qu'il aura calculé (en utilisant la même fonction). L'identité de ces deux hash prouve :

1. que l'auteur est bien celui qu'on croit (authenticité)
2. que les données n'ont pas été modifiées (intégrité).

Un concept annexe à l'authentification est celui de *non-répudiation*. On parle de non-répudiation quand, par exemple dans le cas d'un échange de donnée, on est en mesure de prouver que seul l'expéditeur a pu nous les envoyer. Dans le cas de l'utilisation de signatures numériques, seul le possesseur de la clef privée peut générer une signature vérifiable à l'aide de sa clef publique. Il ne peut donc pas nier qu'il est l'auteur des données, car personne d'autre que lui n'aurait pû le faire. On dit alors que l'échange est non-répudiable.

### Confidentialité

Le problème de confidentialité est très simple : des données numériques, généralement encodées dans un format standard, sont lisibles par quiconque y a accès. Sur le résau TCP/IP cela signifie potentiellement beaucoup de monde. Outre la (mauvaise) possibilité d'utiliser un encodage non-standard secret[9], l'utilisation d'un algorithme de chiffrement permet de protéger les données des regards indiscrets, en les rendant incompréhensibles et indéchiffrable pour tous sauf pour le détenteur de la clef appropriée. Dans le cas de l'utilisation d'un algorithme symétrique la clef aura dû être partagée au préalable ; dans le cas d'un algorithme asymétrique la clef publique du destinataire sera utilisée pour le chiffrement, ce dernier utilisera alors sa clef privée pour déchiffrer et lire les données.

Une technique alternative consiste à combiner l'utilisation d'un algorithme symétrique et d'un algorithme asymétrique : les données sont chiffrées avec l'algorithme symétrique en utilisant une clef très longue, et cette clef est alors chiffrée par l'algorithme asymétrique. De cette façon on réduit le coût de l'opération de chiffrement tout en conservant l'avantage du système à clef publique (facilité d'échange de clef).

### Confidentialité + Authentification et intégrité

Il est possible de combiner ces trois services de la façon suivante, tout d'abord du point de vue de l'expéditeur :

1. Chiffrer les données selon la technique convenue entre les parties en communication.
2. l'authentification et l'intégrité sont alors garanties en utilisant une fonction de hashage sur les données chiffrées, comme décrit plus haut.

En procédant dans cet ordre les données seront mieux protégées des attaques par analyse cryptographique, le hash d'authentification ne donnant pas d'indication sur le contenu des données.

[9]Cette démarche, appelée sécurité par l'obscurité, est considérée comme une mauvaise approche par l'ensemble des professionnels de la sécurité. Elle consiste en effet à protéger des données en gardant un encodage, ou tout autre mécanisme technique, secret. Si ce mécanisme est percé à jour, il faut tout refaire. L'idée derrière les algorithmes cryptographique sest que le secret n'est pas dans le mécanisme (qui est en général publié), mais dans la clef de chiffrement. En cas de problème il sera bien plus facile de renouveler les clefs que de changer de standard d'encodage par exemple.

Le récepteur appliquera les opérations de vérification en premier, puis, si la vérification est positive, il procédera au déchiffrement des données.

### 1.2.1 Vue générale du fonctionnement d'IPsec

Dans cette section nous allons faire un survol rapide d'IPsec, et découvrir son fonctionnement sans entrer dans les détails. Les lecteurs désireux d'en savoir d'avantage pourront ensuite continuer la lecture des sections suivantes où sont décrit chacun des protocoles séparément.

Nous pouvons séparer les protocoles constituant IPsec en deux catégories :

- Les protocoles d'échange de données. Ce sont eux qui seront utilisés dans le cadre d'une collection sécurisée pour garantir les services d'IPsec, à savoir la confidentialité et l'intégrité des données et l'authentification de leur source. Ces sservices reposnet sur l'utilisation d'algorithmes cryptographiques, et ces algorithmes nécessitent des clefs.
- Les protocoles d'échange de clefs. Ce sont ces derniers qui fourniront les clefs nécessaires au protocole d'échange de données choisit. Ces clefs seront échangées au tout début de la connexion.

En fait il y a bien plus que les clefs qui sont négociées à l'ouverture d'une connexion IPsec entre deux hôtes : tous les paramètres relatifs à la connexion, le choix du protocole d'échange, les services à assurer et les algorithmes à utiliser font aussi partie de la négociation. Tous ensemble, ils forment une sorte de contrat entre les deux hôtes en communication. Un tel contrat liant un hôte à un autre, et spécifiant les services à assurer et tous les paramètres nécessaires, s'appelle une Association de Sécurité[10]. Une Association de Sécurité IPsec est unidirectionnelle, et ne fournit les paramètres que pour transmettre d'une hôte à l'autre en sens unique. Il faut donc 2 SA pour chaque connexion, chaque hôte ayant, en plus de la SA qui lui permet de transmettre à son interlocuteur, une copie de celle de son correspondant afin de pouvoir traiter les données qu'il reçoit de ce dernier.

Penchons-nous un instant sur les protocoles d'échange de données. Ensuite nous ferons un tour rapide des protocoles d'échange de clef.

#### Protocoles d'échange de données.

Il existe deux protocoles d'échange de données dans IPsec : le protocole ESP, pour *Encapsulated Security Payload*, et le protocole AH, pour *Authentication Header*.

**ESP** offre les services de confidentialité, de protection de l'intégrité des données et de l'authentification de la source. Ces différents services peuvent être combinés entre eux selon la politique de sécurité choisie.

**AH** offre le service d'authentification et d'intégrité des données uniquement, mais avec protection et authentification de l'adresse IP source et d'une partie des champs de l'entête IP.

Ces deux protocoles peuvent être utilisés dans deux modes différent : le mode Transport ou le mode Tunnel. Le choix du mode affecte l'étendue des données traitées par le protocole. Gardons à l'esprit que les protocoles d'échange de données reçoivent en entrée un paquet IP, et qu'ils produirons un paquet IP contenant les données protégées.

**Transport** Ce mode portera sur le contenu encapsulé dans le paquet IP reçu, à savoir l'entête et les données du protocole de transport (le plus souvent TCP ou UDP). L'entête du

[10] en anglais Security Association, SA en abrégé.

paquet IP en entrée est conservé, en modifiant toutefois les champs nécessaires (next header par exemple). Les données du protocole transport traitées sont précédées d'une entête correspondant au protocole IPsec utilisé (ESP ou AH). Dans le cas de AH, la protection couvrira aussi les champs non-mutables (les champs dont la valeur ne varie pas au cours de la transmission) de l'entête IP.

**Tunnel** est utilisé, comme son nom l'indique, pour faire du *tunnelling*. Ce procédé consiste à encapsuler le paquet IP entier reçu en entrée dans un nouveau paquet IP dont l'entête est construite en fonction des informations contenues dans la SA. Ainsi dans le cas d'un tunnel établit entre deux passerelles, l'adresse IP de destination du nouveau paquet sera celle spécifiée dans la SA, qui n'est pas nécessairement celle qui était présente dans l'entête du paquet IP reçu. Il est important ici de remarquer que dans le cas de l'utilisation du service de confidentialité des données (ESP) en mode Tunnel, les adresses de source et de destination du paquet IP reçu en entrée de traitement sont protégés de la vue par des tiers.

Avec ces deux modes à notre disposition, il est possible d'établir des connexions IPsec de différent types :

**hôte vers hôte** où les extrémités de la connexion IPsec sont aussi les extrémité de la connexion IP protégée par IPsec. Dans ce cas le mode transport ou le mode Tunnel peuvent être utilisés, en remarquant toutefois que dans ce dernier cas il n'y a pas de réelle protection des adresses de source et destination qui sont identiques dans l'entête IP fabriquée lors de l'encapsulation.

**passerelle à passerelle** où les extrémités de la connexion IPsec reçoivent le trafic venant d'autres machines, et l'acheminent via une connexion sécurisée par IPsec vers l'autre extrémité du tunnel, ou les données protégées sont alors dé-encapsulées et transmisent à la machine de destination finale. Dans ce cas là le mode le plus intéressant est le mode Tunnel, qui assurera une protection des adresses des connexions protégées (dans le cas d'utilisation de ESP avec confidentialité). C'est d'ailleurs le mode obligatoire pour une passerelle.

**hôte à passerelle** où un hôte communique avec des machines situées derrière une passerelle, en utilisation IPsec pour protéger les données jusque celle-ci. Dans ce cas l'utilisation du mode Tunnel est également obligatoire, et permettra de protéger l'adresse des machines situées derrière la passerelle (sur un réseau local par exemple).

Pour résumer, les données échangées grâce à une connexion IPsec seront protégées par le protocole ESP ou AH, dont le choix est négocié en même temps que la SA. AH n'assure pas la confidentialité, mais couvre l'entête IP externe du paquet, alors qu'ESP ne couvre que les données de la couche de transport. Le protocole choisit fonctionnera selon le mode spécifié dans la SA, suivant qu'un des hôtes soit une passerelle ou non. Dans le cas de la présence d'une passerelle come extrémité de la connexion IPsec le mode Tunnel est obligatoire, sinon les deux sont possibles. Le mode Tunnel permet la protection d'une seule ou des deux adresses de destination finale.

### Protocoles d'échange de clef.

Il y a plusieurs méthodes possibles pour établir les paramètres d'une connexion IPsec. L'une est de créer manuellement les associations de sécurité, en configurant les paramètres et

les clefs sur chacune des machines qui devra communiquer via IPsec. La méthode manuelle comporte plusieurs défauts, dont la lourdeur d'administration n'est pas le moindre, surtout si le nombre de machines est grand.

L'autre possibilité est de confier cette tâche à un programme utilisant un protocole d'échange de clef. Ce programme sera configuré avec les paramètres généraux conformes à une politique de sécurité, comme par exemple le type d'algorithme à utiliser, la méthode d'authentification choisie, le mode d'encapsulation...

Le protocole d'échange de clef standard utilisé pour négocier les SA pour IPsec est IKE, pour Internet Key Exchange. Ce protocole est lui-même basé sur un framework nommé ISAKMP[11], qui décrit un ensemble d'échanges composés d'une série de messages, qui permettent de procéder à des échanges de clef, sans toutefois préciser les détails de mise en oeuvre. Par exemple ISAKMP décrit la structure des messages permettant aux parties en négociation de s'authentifier, mais ne précise pas le mécanisme d'authentification. Le protocole IKE reprend donc les messages et échanges décrits par ISAKMP, en ajoute quelques uns qui lui sont propres, et précise les éléments laissés libres par ISAKMP, notament les différentes techniques d'authentification.

IKE reprend aussi les deux phases de négociation décrites par ISAKMP : durant une première phase, les deux parties négocient une SA de haut niveau, souvent appelée SA ISAKMP. C'est pendant cette phase 1 que l'authentification a lieu, ainsi qu'un premier échange de clef[12]. La SA ISAKMP permettra ensuite de protéger les échanges des phases 2, pendant lesquelles les SA IPsec seront négociées. Chaque phase 2 constituera la négociation d'une ou plusieurs SA IPsec, donc les clefs seront soit dérivées de celles de la SA ISAKMP, soit générées grâce à un nouvel échange de clef. Tous les échanges des phases 2 sont protégés par les algorithmes et les clefs négociés pendant la phase 1.

L'avantage de cette négociation en deux phases est que la phase 1 n'a lieu qu'une fois pour plusieurs phases 2, et cette phase 1 inclus l'authentification qui est une des opérations cryptographiques les plus coûteuses.

Le protocole IKE sera donc utilisé pour négocier les SA au début d'une connexion IPsec, et également pour le renouvellement de ces SA à leur expiration.

Il existe un autre protocole d'échange de clef nommé Photuris. Ce protocole peut lui aussi être utilisé pour la gestion des SA IPsec. Nous n'en parlerons pas plus dans ce document, le lecteur intéressé pourra se référer à [12] pour plus d'informations.

### 1.2.2 IPsec : un exemple

Afin d'avoir une idée générale du fonctionnement d'IPsec, je vous propose d'examiner ensemble les étapes d'une connexion IPsec entre un hôte A et un hôte B.

Avant de pouvoir entrer en communication, A et B doivent avoir convenu d'un moyen de s'authentifier mutuellement. Il peut s'agir d'un mot de passe ou d'une clef partagée à l'avance, ou de certificats.

Dans l'exemple nous parlerons d'un concept important dans le monde IPsec : l'Association de Sécurité, SA en abrégé. Ce concept sera définit en détail plus loin, contentons-nous pour le moment de savoir qu'une SA est l'ensemble des informations permettant d'identifier une connexion IPsec et de traiter les trafic protégé par celle-ci.

Voyons les différentes étapes de l'établissement de la connexion IPsec entre A et B :

[11] Internet Security Association and Key Management Protocol

[12] les échanges de clefs se font grâce à l'algorithme Diffie-Hellman, dont une description se trouve en annexe.

1. A, qui est l'initiateur de la connexion, envoie à B une demande d'établissement d'une SA de haut niveau. Cette demande contient une serie de propositions pour les algorithmes et le mode d'authentification à utiliser. La SA de haut niveau servira à négocier les SA utilisées pour IPsec dans une deuxième phase de négociation.
2. B répond en choisissant la proposition qui lui convient parmi les propositions reçues de A. Si aucune proposition ne convenait la tentative de connexion s'arrèterait sur l'envoi d'une notification vers A.
3. Au cours de la négociation de cette première SA, un échange de clef et une authentification réciproque ont lieu. A l'issue de cette première phase une SA commune est établie, les deux hôte se sont authentifiés mutuellement et possèdent tout deux le matériel cryptographique nécessaire à la négociation des SA IPsec. Les échanges qui suivront seront protégés grâce aux algorithmes et aux clefs négociés pour la SA de haut niveau.
4. A commence la négociation d'une SA IPsec de la même manière qu'au point 1, en envoyant à B une demande contenant une série de propositions relatives au protocole à utiliser pour IPsec, au type de service désiré, aux algorithmes d'authentification et/ou de chiffrement à utiliser.
5. B répondra de la même façon, en mentionnant dans sa réponse la proposition qui lui convient.

   Si les hôtes l'exigent, un nouvel échange de clef aura également lieu pendant cette négociation, afin de garantir que les clefs négociées au cours de cet échange sont indépendantes des clefs de la SA de haut niveau (qui sont utilisées dans la dérivation des nouvelles clefs).
6. A l'issue de cette seconde négociation chaque hôte dispose d'une SA lui permettant de transmettre des données sécurisées vers son interlocuteur, ainsi que des informations nécessaires pour traiter le trafic reçu. En fonction des services négociés, ils utiliseront soit le protocole AH, ou le protocole ESP, avec les paramètres contenus dans les SA IPsec. Il existe deux modes d'utilisation de ces deux protocoles, c'est le mode qui détermine la façon dont les données sont encapsulées par IPsec.
7. Au cours de la vie de la connexion les SA IPsec, pour lesquelles une durée de vie maximum a été négociée, peuvent expirer. Dans ce cas une nouvelle négociation doit être démarée, comme expliquée dans les points 4 et 5. La communication continuera en utilisant les nouvelles SA, les anciennes seront détruite à l'expiration de leur limite "rigide", limite au delà de laquelle plus aucun trafic ne peut être traité par une SA.
8. A la fin de la connexion, les SA qui y sont relatives sont détruites. L'hôte qui coupe la connexion envoie un message de notification pour signifier la fin de la connexion à son interlocuteur, de sorte que celui-ci peut lui aussi détruire la ou les SA concernées.

### 1.2.3 Problèmes de sécurité résolus par IPsec

Reprenons les problèmes énumérés dans la section de rappel sur TCP/IP, et voyons lesquels IPsec peut résoudre et comment.

1. Attaques par déni de service. Dans la mesure où une application réseau n'est accessible que par une connexion IPsec établie, on peut dire qu'IPsec isole cette application de tous les attaquants potentiels qui ne peuvent pas ouvrir une connexion IPsec. Mais cela n'empêche pas les attaques de ce type, surtout que IPsec lui-même peut être sujet au

déni de service comme cela est expliqué dans [13]. Nous avons testé une des vulnérabilités décrites, le compte-rendu se trouve dansle dernier chapitre traitant l'étude de cas.

2. Espionnage sur le réseau. Ce type de problème est résolu par l'utilisation du protocole ESP d'IPsec qui assure la confidentialité des données transmises, et même des adresses de source et/ou de destination si il s'agit d'un tunnel et qu'au moins l'une des deux machines est une passerelle IPsec. En effet un tiers interceptant le trafic protégé par ESP sera incapable d'en déchiffrer le contenu, et même si il y parvenait pour l'une des clefs, il devrait recommencer tout le travail quand la SA sera renouvelée et une nouvelle clef générée.
3. Attaques par modification des données. Ce type d'attaque est rendu impossible par l'utilisation du protocole AH ou ESP qui fournissent une méthode de contrôle de l'intégrité des données basée sur l'utilisation d'algorithmes cryptographiques. Si un attaquant essaye de modifier les données véhiculée par une connexion, il lui sera impossible de reconstruire une valeur de contrôle d'intégrité valide, et la fraude sera détectée par les parties en communication.
4. Attaques de niveau applicatif. Ce type d'attaque n'est pas vraiment résolu par IPsec. Une application comportant une faille comme un *buffer overflow* aura toujours cette faille même si le protocole utilisé est IPsec. Toutefois, dans la mesure où l'accès à une application sensible à ce type d'attaque n'est possible que via IPsec, et donc par des utilisateurs s'étant correctement authentifiés, IPsec constitue une barrière de protection en empêchant l'accès par la plupart des gens mal-intentionnés sur le réseau. On ne peut donc pas dire qu'IPsec solutionne ce problème, mais il apporte une protection supplémentaire vis-à-vis des utilisateurs non-autorisés parmi lesquels se trouvent la plupart des attaquants potentiels.
5. Attaques basées sur l'utilisation du routage par la source. Ce type d'attaque est rendu impossible par le fait que la source de chaque paquet est authentifiée individuellement, donc un attaquant utilisant le routage par la source sera détecté car les paquets qu'il enverra ne passeront pas le contrôle d'authenticité.

Comme nous pouvons le voir, IPsec ne résoud pas tous les problèmes, mais apporte une solution à la plupart d'entre eux.

## 1.3 IPSec - Spécification des protocoles

Nous allons maintenant examiner plus en détail les différents protocoles qui, ensembles, forment IPsec. Nous commencerons par examiner ce que sont les Associations de Sécurité, quel rôle elles jouent, les informations qui les composent, et leur cycle de vie.

Ensuite nous examinerons les protocoles d'échange de données ESP et AH. Nous verrons les services qu'ils assurent, quelles structures de données ils utilisent, et quels traitements ils appliquent aux données pour assurer leur rôle.

Nous terminerons par le framework ISAKMP qui définit les messages et les échanges nécessaires à la négociation des SA, et par le protocole d'échange de clef IKE qui est basé sur le framework ISAKMP et qui est utilisé par IPsec pour l'échange des clefs et la négociation des SA.

L'architecture générale d'IPsec est décrite dans [14].

### 1.3.1 Un concept clef : l'Association de Sécurité (SA)

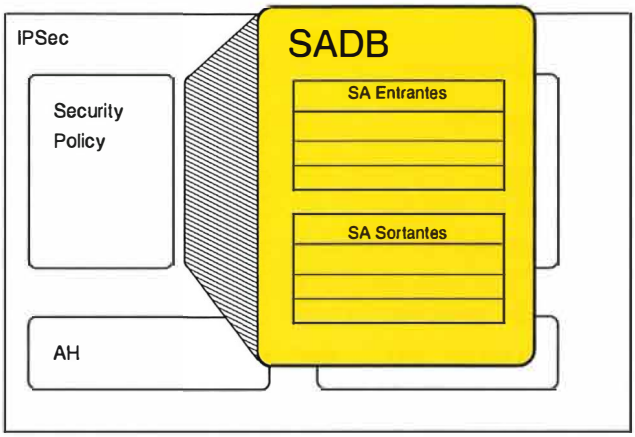



FIG. 1.6 – Architecture IPSec - SADB et SAs

Pour comprendre ce qu'est une Association de Sécurité (Nous utiliserons l'acronyme SA, de l'anglais *Security Association*, dans le reste du texte), imaginons un hôte A voulant établir une connexion sécurisée avec un hôte B. Lors de la négociation de cette connexion les deux hôtes, après s'être assuré de l'identité de leur interlocuteur (authentification), vont se mettre d'accord sur le type de service à assurer, ainsi que les modalités de ces services, c'est-à-dire les algorithmes utilisés, les clefs à utiliser, ainsi que d'autres paramètres que nous examinerons plus loin. Si les deux machines tombent d'accord, les échanges sécurisés peuvent alors avoir lieu. On reconnait ici l'analogie avec la négociation d'un contrat de service entre deux parties, car il s'agit exactement de cela. Ainsi, à l'issue de la négociation A et B seront chacun liés par un contrat reprenant les services à assurer et les modalités de ce service. C'est ce contrat qu'on appelle l'*Association de Sécurité* (SA). La SA de A et celle de B vont ensemble, mais elles sont bien distinctes et unidirectionnelles : la SA de A permettra à A de transmettre vers B, et celle de B de transmettre vers A. Et, pour poursuivre l'analogie avec un contrat juridique, chaque partie aura une copie du contrat liant l'autre partie, afin de pouvoir vérifier si le service presté est conforme aux cahier des charges. Dans le cas de notre connexion, B conserve une copie de la SA de A afin de vérifier que les données transmises correspondent aux services convenus, et également afin de pouvoir les déchiffrer et/ou vérifier leur authenticité grâce aux paramètres enregistrés dans la SA (Clefs, algorithmes à utiliser).

Abandonnons là notre analogie et examinons les SA de manière plus formelle. On pourrait définir la SA comme *l'ensemble des données échangées lors de la (re)-négociation d'une connexion IPSec, pour un protocole donné, et qui permettront l'envoi des données par l'initiateur et la réception par le récepteur avec le niveau de service convenu par les deux hôtes.*

Reprenons notre exemple de connexion entre A et B, et concentrons nous sur le rôle de A : C'est A qui désire ouvrir la connexion, il est donc l'*initiateur*, et B le *récepteur*. A va donc négocier avec B les algorithmes et les clefs nécessaires à l'envoi de données vers B, en fonction de ce que A peut offrir et ce que B supporte. D'autres données sont également nécessaires à l'échange, par exemple la périodicité de renouvellement des clefs.

La négociation des SA est un mécanisme complexe, c'est en fait la partie la plus complexe d'IPSec, qui fait l'objet d'un protocole spécifique appelé IKE (Internet Key Exchange). Nous

examinerons ce protocole en détail plus loin. Pour le moment voyons ce que contient une SA, quelles sont les données qui l'identifient, et comment elles sont gérées pour l'envoi et la réception de données.

#### 1.3.1.1 Informations contenues dans une SA

##### 1.3.1.1.1 Une SA est identifiée par un triplet composé de :

1. l'adresse IP de destination. Attention, il s'agit ici de la destination finale des données ; si l'hôte B avec qui a lieu la négociation est l'hôte destinataire, alors il s'agit de l'adresse de B. Mais si B est une passerelle utilisée pour envoyer des données à un hôte C, alors c'est l'adresse IP de C qui sera utilisée ici. Plus de détails dans la section sur le mode de transmission plus loin dans ce chapître.
2. le protocole utilisé : ISAKMP, AH ou ESP. Le premier sera utilisé dans le cas de la phase 1 de négociation du protocole IKE, les deux autres protocoles sont ceux vraiment utilisés pour l'échange de donnée et seront sélectionnés dans la phase 2 d'une négociation par IKE. Nous verrons cela en détail dans la section consacrée à IKE.
3. le *Security Parameter Index* (SPI), une chaîne de 32 bits qui sera ajoutée par le récepteur (B) lors de la négociation. Le SPI sera utilisé par B pour identifier la SA correspondante pour chaque paquet reçu dans le cadre de la connexion avec A. B doit donc s'assurer de l'unicité du SPI qui constitue une partie de sa clef primaire dans la base de donnée des SAs entrantes (celles qui permettent de traiter le trafic entrant, c'est-à-dire venant de l'extérieur). L'initiateur joindra toujours le SPI de la SA utilisée à chaque paquet envoyé.

L'initiateur A pourra identifier la SA à utiliser pour une connexion grâce à l'IP de destination et le protocole utilisé. Par contre B n'aura pas toujours accès à l'adresse IP de destination finale des paquets reçus (car elle pourrait être chiffrée si on utilise ESP). En effet il faut d'abord identifier la SA correspondante pour pouvoir déchiffrer le contenu du paquet, et l'adresse de destination est contenue dans ce paquet, c'est donc le problème de la poule et de l'oeuf. L'utilisation du SPI permet de résoudre ce problème, le SPI sera toujours transmis en clair.

Il pourra aussi arriver que le triplet < dst, spi, proto > ne suffise pas à identifier la bonne SA sans ambiguïté dans le cas d'une machine comportant plusieurs interfaces réseau. Pour résoudre ce problème il faut soit ajouter l'adresse source comme élément supplémentaire pour l'identification, soit assurer l'unicité globale du spi (pour toutes les SA servant à la réception sur le système concerné).

##### 1.3.1.1.2 Autres informations contenue dans une SA :

1. **Numéro de séquence :** un compteur de 32 bits dont la valeur sera augmentée de 1 pour chaque paquet sécurisé par la SA. Utilisé en conjonction avec la fenêtre anti-rejeu (*anti-replay window*), il permet de détecter les attaques par rejeu de paquet. Le risque de dépassement du compteur est très faible car en général la durée de vie d'une SA ne permet pas de transmettre $2^{32}$ paquets.
2. **Fenêtre anti-rejeu :** on peut la décrire comme un tableau dont la borne inférieure est égale au numéro de séquence du prochain paquet attendu pour la SA, et dont la taille est fixée sur un nombre de paquets à recevoir (en général 32 ou 64). Le tableau est décalé au fur et à mesure de la réception, de telle sorte qu'un paquet dont le numéro de séquence

ne se trouvant pas dans la fenêtre (comme ce sera le cas pour un paquet rejoué par un attaquant) sera rejeté. *NB :* un paquet retransmit par l'autre participant à la connexion aura un nouveau numéro de séquence, la retransmission étant assurée par la couche de transport qui se situe au dessus d'IPSec dans le modèle en couche des protocoles réseau.

3. **Durée de vie :** définit, en terme de durée ou de trafic, la longévité maximum de la SA. On trouve deux types de limite :
   - la limite "soft" (souple) qui peut être dépassée et indique qu'il est temps de renouveler la SA.
   - limite "hard" (rigide) qui interdit toute utilisation ultérieure de la SA une fois atteinte.

   Une nouvelle SA doit donc être créée entre ces deux limites. On peut définir pour une même SA des limites temporelles et/ou des limites de trafic.
4. **Mode :** Nous verrons plus loin qu'il existe deux modes d'encapsulation différents pour les connexions IPSec - transport ou tunnel. Toute SA entre dans une de ces deux catégories.
5. Paramètres propres au protocole utilisé pour la SA (AH ou ESP). Ces paramètres seront détaillés dans les sections dédiées à AH et ESP.

#### 1.3.1.2 Cycle de vie d'une Association de Sécurité

1. Création

   La création d'une nouvelle SA se produit dans deux circonstances possibles :
   - lors de l'établissement d'une connexion IPSec,quand deux hôtes veulent communiquer entre eux directement ou en traversant une passerelle sécurisée.
   - lorsque la limite d'expiration d'une SA existante est atteinte. Une nouvelle SA est alors créée pour la remplacer. C'est le système de gestion automatique des SA qui devra prendre ce renouvellement périodique en charge. En cas d'utilisation manuelle des SA il n'est pas possible de les renouveler périodiquement ; on perd donc des possibilités comme la protection anti-rejeu et la propriété de Perfect Forward Secrecy.
2. Destruction

   Une SA sera détruite :
   - soit sur requète d'un des deux hôtes lors de la fin de l'échange sécurisé
   - soit au terme de sa durée de vie. Lorsque la limite souple est atteinte, la SA peut encore être utilisée mais doit être renouvelée dés que possible. Une fois la limite rigide ("hard") atteinte la SA ne peut plus être utilisée. Il se peut donc que, entre ces deux limites, la SA en fin de vie et une nouvelle SA coexistent, sans constituer de problème.

### 1.3.2 Encapsulated Security Payload

Le protocole ESP assure les services de confidentialité des données, avec en option la protection de l'intégrité des données et l'authentification de la source. ESP fournit aussi une protection optionnelle contre les attaques par rejeu de paquet. Ces différents services peuvent se combiner selon les besoins, il est possible d'utiliser ESP avec confidentialité mais sans authentification et intégrité, et vice-versa. Il est toutefois déconseillé d'utiliser l'option de confidentialité seule, car dans ce cas rien ne permettra de détecter une attaque active qui modifierait les données chiffrées, et la protection anti-rejeu est également impossible. Le protocole ESP est décrit dans [15].

Nous allons examiner, dans l'ordre :

- La structure des paquets ESP, en particulier vis-à-vis de la portée des traitements sur les différents champs, en fonction du type d'encapsulation (tout cela pour IPv4)
- le traitement d'un paquet sortant (encapsulation)
- le traitement d'un paquet entrant (dé-encapsulation).

#### 1.3.2.1 Structure des paquets ESP

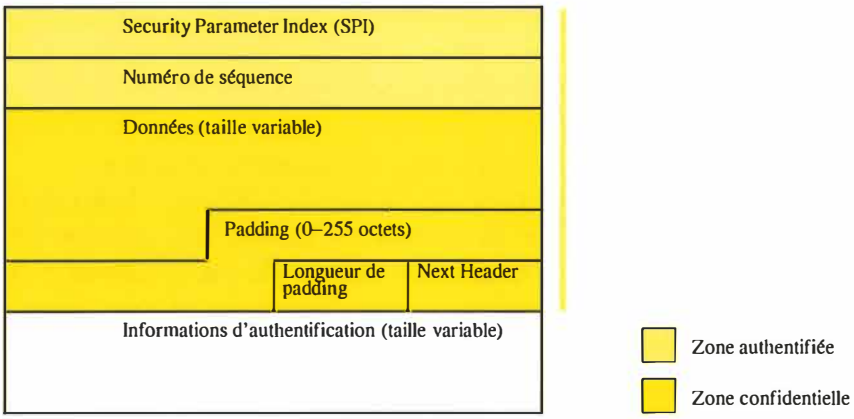



FIG. 1.7 – Format d'un paquet ESP

Examinons les champs composant le format des données traitées par ESP :

**Security Parameter Index** champs de 32 bits généré lors de l'établissement de la SA à laquelle est attaché ce paquet. Le SPI permettra à la machine réceptrice d'identifier la SA correspondante pour le traitement du paquet.

**Numéro de Séquence** compteur numérotant chaque paquet traité par une SA. Ce compteur est incrémenté de 1 pour chaque nouveau paquet envoyé, et peut être utilisé par le récepteur pour la protection anti-rejeu. L'émetteur *doit* obligatoirement maintenir ce compteur, quel que soit le traitement qu'en fera le récepteur.

**Données** contient les données protégées par ESP. Il s'agira soit d'un segment TCP ou d'un datagramme UDP issu de la couche de transport dans le cas du mode d'encapsulation *transport*, soit d'un paquet IP complet dans le cas du mode *tunnel*. L'encapsulation est détaillée plus loin.

Ce champ contiendra éventuellement un vecteur d'initialisation qui ne sera pas chiffré. Ce vecteur d'initialisation servira à synchroniser l'algorithme au moment du déchiffrement.

**Padding** permet de compléter la taille des données à chiffrer dans le cas où l'algorithme de chiffrement nécessite une taille de bloc multiple d'un nombre d'octets donné. En dehors de ce cas, il doit aussi être utilisé pour assurer l'alignement du champ de donnée comme multiple de 4 octets, de telle sorte que le champ de longueur de padding commence au deuxième octet du mot de 32 bits dont il fait partie. Ce padding a également pour but possible de cacher la taille réelle des données chiffrées.

La taille de ce champ sera comprise entre 0 et 255 bits, et devra[13] contenir une série d'octets représentant des valeurs entières non-signées et monotoniquement croissantes de 1. Le premier octet vaudra 1, le second 2, etc.

**Longueur de padding** champ de 8 bits indiquant la taille du champ de padding décrit ci-avant.

**Next Header** contient le numéro de l'entête suivante, tel que défini dans la liste des numéros de protocoles gérée par l'IANA[14].

**Informations d'authentification** champ optionnel qui ne sera présent que si le service d'authentification est utilisé. Ce champ est de longueur variable qui dépendra de l'algorithme utilisé.

#### 1.3.2.2 Traitement ESP et mode d'encapsulation

Comme l'indique la figure 1.7, le chiffrement porte uniquement sur les champs de donnée, de longueur de padding et le "next header". L'authentification porte sur tout le paquet ESP, incluant donc en plus de la zone couverte par le chiffrement le SPI et le numéro de séquence, les protégeant ainsi de toute modification au cours du transit. Cela est indispensable pour la protection anti-rejeu, sinon le numéro de séquence ne serait pas fiable.

Il existe deux modes d'encapsulation différents.

1. Le mode *Transport* protège les données de la couche de transport, sans modifier l'entête IP. Dans le cas de données transmises via TCP, ESP devra donc traiter une structure composée d'une entête TCP et des données en provenance de l'application, l'entête ESP sera insérée après l'entête IP. Dans le cas d'un paquet IPv6, l'entête ESP sera insérée après certaines entêtes d'extension spécifiquement requises pour l'acheminement du paquet, à savoir les extensions "hop-by-hop", de routage, de destination et de fragmentation. Le lecteur désireux d'en savoir plus sur IPv6 pourra consulter [6, 7].
2. Le mode *Tunnel* protège l'ensemble du paquet IP et l'encapsule entièrement dans le champ de donnée ESP. Une nouvelle entête IP sera donc créée, avec pour adresse de destination celle spécifiée dans la SA. De par le fait que l'entête IP d'origine fait partie des données chiffrées, on a donc une protection de la source et de la destination du trafic, cela tout au moins dans le cas où ce trafic est transmis via 2 passerelles utilisant le service de confidentialité d'ESP. Dans le cas où au moins l'un des deux hôtes est une extrémité du tunnel, son identité n'est pas protégée puisque c'est son adresse qui figurera comme source IP dans l'entête externe des paquets qu'il envoie, et comme destination des

[13] sauf spécification contraire par l'algorithme de chiffrement utilisé. Cette spécification devra être intégrée à un RFC définissant l'utilisation de l'algorithme.

[14] Internet Assigned Number Authority

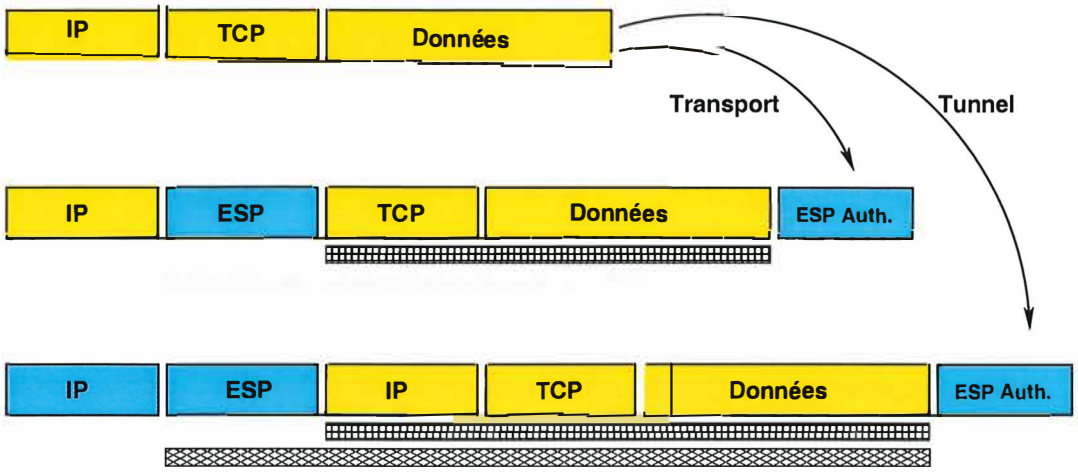



FIG. 1.8 – Modes Transport et mode Tunnel

paquets qui lui sont envoyés. Il n'y a donc réelle protection des sources et destinations finales du trafic que dans le cas où un tunnel sert à relier deux réseaux locaux via deux passerelles utilisant IPsec entre elles. Des observateurs extérieurs ne verront alors qu'un échange entre deux machines, sans voir le contenu des connexions véhiculées par le tunnel.

#### 1.3.2.3 Traitement d'un paquet sortant

Examinons la séquence d'action correspondant au traitement d'un paquet IP devant être traité par ESP. On suppose donc que la politique de sécurité a déjà été vérifiée (par une recherche dans la base de donnée regroupant les règles de politique, la SPD) et que ce paquet requiert l'utilisation d'ESP.

1. Recherche de la SA correspondante dans la SADB. C'est cette SA qui contient les informations sur les algorithmes à utiliser et les clefs correspondantes. Si aucune SA n'est trouvée le paquet doit être rejeté. Ce rejet provoquera également l'appel au service de gestion automatique des SA implémentant le protocole IKE, afin qu'une nouvelle paire de SA soit créée entre les deux hôtes.
2. Chiffrement du paquet. Cette opération a une portée différente suivant le mode d'encapsulation utilisé :
   - en mode transport, la partie du paquet provenant de la couche transport est chiffrée. Dans le cas de TCP cette partie est composée de l'entête TCP et des données. L'entête IP est reportée au début du paquet, la valeur du champs "next header" est mise à 50, qui est le numéro du protocole ESP attribué par l'IANA.
   - en mode tunnel, le paquet tout entier est chiffré, et une nouvelle entête IP est construite. L'entête intérieure aura pour seule modification le décrément du champ TTL et la mise à jour du *checksum* en conséquence. L'entête extérieure sera construite comme suit :
     - Les adresses de source et de destination seront celles spécifiées par la SA. En général l'adresse source sera une adresse de l'interface par laquelle le paquet sera envoyé, mais ce n'est pas obligatoire. L'adresse de destination sera également différente de

la destination finale du paquet encapsulé, en particulier dans le cas d'un tunnel vers une passerelle.
- Le drapeau MF (More Fragment, indiquant que le paquet est fragmenté et que d'autres morceaux suivent) sera remis à zéro car l'encapsulation se fait toujours sur un paquet entier, après éventuelle défragmentation. Le drapeau DF sera soit copié de l'entête intérieure, soit dépendra de la configuration. Dans le cas de détection du PMTU au cours de la négociation la fragmentation sera en principe évitée au cours de la transmission, le drapeau DF peut donc être mis à 1.
- Le champ TOS sera copié de l'entête intérieure (cas d'IPv4).

Une fois les données à chiffrer sélectionnées, un padding sera éventuellement ajouté. Le tout est ensuite chiffré par l'algorithme spécifié, suivant le mode spécifié (par ex. CBC). Ensuite les données de synchronisation seront ajoutées au texte chiffré, selon les spécifications de l'algorithme.

3. Génération du numéro de séquence. Lors de l'établissement de la SA, ce compteur est initialisé à zéro. Pour chaque paquet traité par cette SA le compteur est d'abord incrémenté de un et la valeur obtenue est insérée dans le paquet. Le premier paquet traité par une SA portera donc le numéro 1.
4. Calcul du champ de vérification d'intégrité. Ce champ sera calculé sur l'ensemble du paquet ESP, donc sans l'entête IP extérieure. Le SPI, le numéro de séquence, les données, le champ de taille et le next header sont couverts par la vérification d'intégrité.

   L'authentification et la vérification d'intégrité ne font en fait qu'un : le calcul du champ de vérification se fera en général par l'utilisation d'un algorithme de hash avec clef (Keyed hash function), sur base d'une clef échangée lors de l'établissement de la SA. Comme mentionné précédemment, la couverture du numéro de séquence est indispensable à l'utilisation du mécanisme de protection anti-rejeu.
5. Fragmentation. Il se peut que la taille du paquet, après traitement par ESP, excède la taille maximum que le chemin vers l'hôte de destination est capable de gérer (PMTU). Dans ce cas la couche IP fragmentera le paquet avant sa transmission.

A la fin de ces étapes, le paquet IP résultant peut être transmis vers l'hôte dont l'adresse est spécifiée dans l'entête IP extérieure.

#### 1.3.2.4 Traitement d'un paquet entrant

1. Défragmentation. Dans le cas où le paquet contenant les données protégéss par ESP a été fragmenté durant sa transmission, la couche IP ré-assemblera le paquet avant de le transmettre à ESP, et ré-initialisera le champ MF et le champ offset à zéro. Si ESP reçoit un paquet fragmenté il doit le rejeter et l'évènement doit être noté dans le journal du système.
2. Recherche de la SA correspondante dans la SADB. Comme nous l'avons vu pour les paquets sortant, la SA contient les paramètres relatifs au traitement à appliquer au paquet : vérification ou non du numéro de séquence, algorithme à utiliser pour le déchiffrement et l'authentification, clefs. Si aucune SA valide n'est trouvée le paquet est rejeté et l'évènement est noté dans le journal du système.
3. Vérification du numéro de séquence. Cette vérification n'est pas obligatoire, le récepteur a le choix de le faire ou pas. Cette vérification est faite grâce à un mécanisme de fenêtre

coulissante similaire à celui employé pour le protocole TCP. Une fenêtre d'au moins 32 numéros est définie, le côté "gauche" correspond au plus petit numéro de séquence non encore reçu. Si un paquet tombe à gauche de la fenêtre, c'est-à-dire si son numéro de séquence est inférieur au numéro le plus à gauche, c'est un doublon et il doit être rejeté. Si le numéro de paquet est dans la fenêtre, il faut vérifier qu'aucun paquet n'a déjà été reçu portant le même numéro, dans lequel cas le nouveau paquet doit être rejeté. Si le numéro du paquet est dans la fenêtre et n'a pas encore été reçu, alors on peut procéder à la vérification de l'authenticité. Ce n'est qu'après cette vérification que la fenêtre sera mise à jour. La vérification a lieu au début du traitement afin de détecter le plus tôt possible les duplicats, dûs ou non à une attaque par rejeu, et de rejeter les paquets fautifs avant d'entamer les opérations coûteuses en terme de calcul.

Il est à noter qu'en cas de perte d'un paquet causé par un problème de réseau (routage,...), le protocole de transport tentera une retransmission, et donc le paquet retransmis sera un paquet différent du point de vue d'ESP, il portera donc un nouveau numéro de séquence et ne posera pas de problème de duplicat.

4. Vérification de l'authenticité et de l'intégrité. Le récepteur calcule le champ de vérification d'intégrité en utilisant l'algorithme et la clef spécifiés dans la SA, selon les mêmes modalités que celles utilisées par l'expéditeur, puis le compare avec celui reçu dans le paquet ESP. L'identité des deux champs constitue une preuve de l'intégrité du message car toute modification de celui-ci entraînerait une différence dans le résultat du calcul. Cela prouve aussi l'identité de l'expéditeur puisque celui-ci est le seul à connaître la clef utilisée pour le calcul. Si les deux champs ne sont pas identiques le paquet doit être rejeté et l'évènement noté dans le journal du système.
5. Déchiffrement. Là encore l'algorithme et la clef à utiliser font partie des informations contenues dans la SA.

### 1.3.3 Authentication Header

Le protocole AH assure les services d'authentification de la source et d'intégrité des données. Ce service diffère de celui offert par ESP, non seulement il n'y a pas de confidentialité des données, mais surtout l'étendue des données dont l'intégrité est assurée est différente. En effet, à la différence d'ESP, la protection par AH couvre l'ensemble du paquet IP, avec toutefois quelques nuances comme nous le verrons plus loin.

Le protocole AH est décrit dans [16].

#### 1.3.3.1 Structure des paquets traités par AH

Le traitement des paquets par AH n'est pas une encapsulation comme c'est le cas pour ESP, mais consiste en l'ajout d'une entête qui se placera juste après l'entête IP. Voyons le contenu de cette entête représentée par la figure 1.9 :

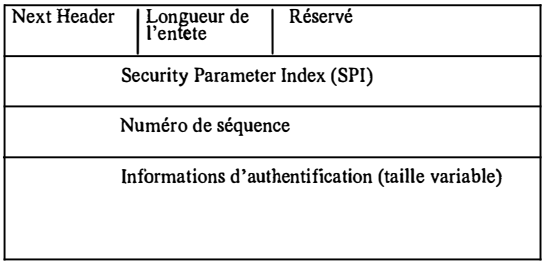



FIG. 1.9 – Format de l'entête AH

**Next Header** contient sur 8 bits le numéro de protocole de l'entête suivante dans le paquet. Dans le cas d'IPv4 il s'agira du numéro du protocole de la couche de transport, le plus souvent TCP ou UDP ; dans le cas d'IPv6 il pourra aussi s'agir du numéro de l'entête d'extension suivante dans la chaîne d'extension.

**longueur d'entête** contient la longueur totale codée sur 8 bits de l'entête AH.

**Réservé** est une zone de 16 bits réservée pour un usage futur. Ce champ doit être rempli de zéros par l'expéditeur et sera ignoré par le récepteur.

**Security Parameter Index** champ de 32 bits généré lors de l'établissement de la SA à laquelle est attaché ce paquet. Le SPI permettra à la machine réceptrice d'identifier la SA correspondante pour le traitement du paquet.

**Numéro de séquence** compteur numérotant chaque paquet traité par une SA. Ce compteur est incrémenté de 1 pour chaque nouveau paquet envoyé, et peut être utilisé par le récepteur pour la protection anti-rejeu. L'émetteur *doit* obligatoirement maintenir ce compteur, quel que soit le traitement qu'en fera le récepteur.

**Information d'authentification** contient la valeur de vérification d'intégrité et d'authentification, calculée comme nous allons le voir dans la suite de cette section. La taille de ce champ doit être un multiple de 32 bits[15].

[15]La taille sera un multiple de 64 bits sous IPv6.

#### 1.3.3.2 Traitement des paquets IP : Champs mutables et non-mutables

A la différence du protocole ESP, la protection d'intégrité offerte par AH s'étend aussi à l'entête IP extérieure, celle qui sera utilisée lors de la transmission. Or certains champs seront changés au cours de la transmission par les routeurs qui font suivre les paquets. Cela pose un problème, puisque dans ce cas la vérification de l'intégrité échouera. Afin d'éviter cela les champs de l'entête IP ont été classés en 3 catégories :

**les champs non-mutables** sont ceux qui conservent leur valeur tout au long de la vie du paquet. Ils peuvent donc être inclus tels quels dans le calcul du champ d'authentification. Les champs non-mutables sont :

- Version
- Longueur de l'entête IP
- Longueur totale
- Identification
- Protocole (aura la valeur 51 pour AH)
- Adresse source
- Adresse de destination, dans le cas où il n'y a pas de routage par la source (source routing)

**les champs mutables** sont ceux dont la valeur sera modifiée et dont on ne peut pas prévoir la valeur finale. Ces champs seront exclus dans le calcul du champ d'authentification, leur contenu sera remplacé par une suite de zéros pour l'opération. Les champs mutables sont :

- *Type de service (TOS)*, pourra être modifié par certains routeurs.
- *les drapeaux (Flags)*, le drapeau DF (Don't Fragment) pourra être mis à 1 par un routeur intermédiaire.
- *l'offset de fragment (Fragment Offset).* AH s'applique toujours à des paquets non-fragmentés (après ré-assemblage éventuel), donc ce champ doit normalement toujours avoir une valeur composée de zéros. Il reçoit donc le même traitement que les paquets mutables (mais dans ce cas il ne sera pas nécessaire de modifier la valeur).
- *Time to Live (TTL)*, sera décrémenté par chaque routeur.
- le Checksum de l'entête, sera recalculé à chaque changement d'un des autres champs.

**les champs mutables dont la valeur est prévisible**

- Adresse de destination, dans le cas de routage par la source. Dans ce cas l'adresse source sera changée pour suivre le chemin indiqué par la source, mais on connait l'adresse de destination finale. C'est cette valeur qui servira pour le calcul du champ d'authentification.

#### 1.3.3.3 Traitement d'un paquet sortant

Examinons la séquence d'actions correspondant au traitement d'un paquet provenant de la couche IP devant être traité par AH. On suppose donc que la SPD a déjà été vérifiée et que ce paquet requiert l'utilisation du protocole AH.

1. Recherche de la SA correspondante dans la SADB. C'est cette SA qui contient les informations sur les algorithmes à utiliser et les clefs correspondantes. Si aucune SA n'est trouvée le paquet doit être rejeté. Ce rejet provoquera également l'appel au service de

gestion automatique des SA implémentant le protocole IKE, afin de créer une paire de SA pour prendre la connexion en charge.

2. Génération du numéro de séquence. Lors de l'établissement de la SA, ce compteur est initialisé à zéro. Pour chaque paquet traité par cette SA le compteur est d'abord incrémenté de 1 et la valeur obtenue est insérée dans le paquet. Le premier paquet traité par une SA portera donc le numéro 1.
3. Calcul du champ d'intégrité et d'authentification. Comme nous l'avons vu plus haut cette valeur sera calculée sur les données du paquet et sur les champs non-mutables de l'entête IP et les valeurs finales de ceux dont la valeur est prévisible. L'algorithme utilisé pour ce calcul ainsi que la clef sont précisés dans la SA.
4. Fragmentation. Il se peut que la taille du paquet, après traitement par ESP, excède la taille maximum que le chemin vers l'hôte de destination est capable de gérer (PMTU). Dans ce cas la couche IP fragmentera le paquet avant sa transmission.

A la fin de ces étapes, le paquet IP résultant peut être transmit vers l'hôte dont l'adresse est spécifiée dans l'entête IP extérieure.

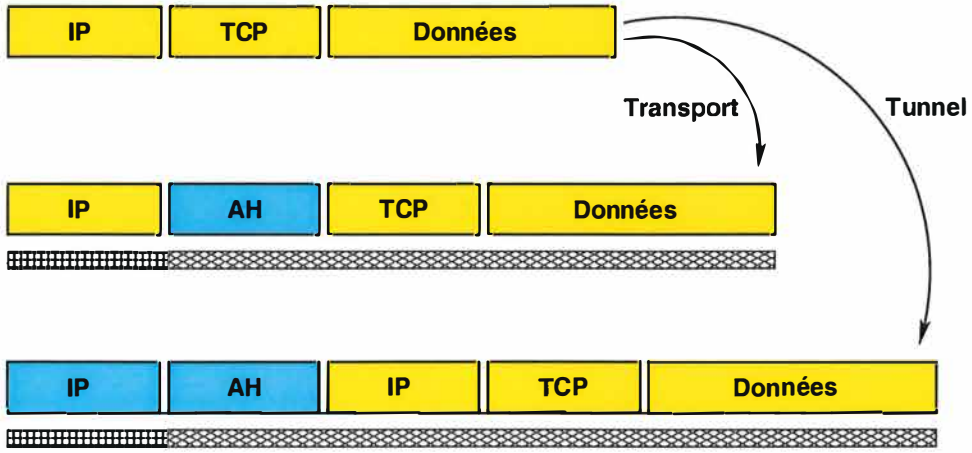



FIG. 1.10 – Modes transport et mode Tunnel

#### 1.3.3.4 Traitement d'un paquet entrant

1. Défragmentation. Dans le cas où le paquet contenant les données protégées par AH a été fragmenté durant sa transmission, la couche IP ré-asemblera le paquet avant de le passer à AH, et ré-initialisera le champ MF et le champ offset à zéro. Si AH reçoit un paquet fragmenté il doit le rejeter et l'évènement doit être noté dans le journal du système.
2. Recherche de la SA correspondante dans la SADB. Comme pour les paquets sortants, la SA contient les paramètres relatifs au traitement à appliquer au paquet : vérification ou non du numéro de séquence, algorithme à utiliser pour le calcul du champ d'intégrité, clefs. Si aucune SA valide n'est trouvée le paquet est rejeté et l'évènement est noté dans le journal du système.
3. Vérification du numéro de séquence. Cette vérification n'est pas obligatoire, le récepteur a le choix de la faire ou pas. Cette vérification est faite grâce à un mécanisme de fenêtre

coulissante similaire à celui employé pour le protocole TCP. Une fenêtre d'au moins 32 numéros est définie, le côté "gauche" correspond au plus petit numéro de séquence non encore reçu. Si un paquet tombe à gauche de la fenêtre, c'est-à-dire si son numéro de séquence est inférieur au numéro le plus à gauche, c'est un doublon et il doit être rejeté. Si le numéro de paquet est dans la fenêtre, il faut vérifier qu'aucun paquet n'a déjà été reçu portant le même numéro, dans lequel cas le nouveau paquet doit être rejeté. Si le numéro du paquet est dans la fenêtre et n'a pas encore été reçu, alors on peut procéder à la vérification de l'authenticité. Ce n'est qu'après cette vérification que la fenêtre sera mise à jour. La vérification a lieu au début du traitement afin de détecter le plus tôt possible les duplicats, dûs ou non à une attaque par rejeu, et de rejeter les paquets fautifs avant d'entamer les opérations coûteuses en terme de calcul.

Il est à noter qu'en cas de perte d'un paquet causée par un problème de réseau (routage,...), le protocole de transport tentera une retransmission, et donc le paquet retransmis sera un paquet différent du point de vue d'AH, il portera donc un nouveau numéro de séquence et ne posera pas de problème de duplicat.

4. Vérification de l'authenticité et de l'intégrité. Le récepteur calcule le champ de vérification d'intégrité en utilisant l'algorithme et la clef spécifiés dans la SA, selon les mêmes modalités que celles utilisées par l'expéditeur, puis le compare avec celui du paquet à valider. L'identité des deux champs constitue une preuve de l'intégrité du message car toute modification de celui-ci entraînerait une différence dans le résultat du calcul. Cela prouve aussi l'identité de l'expéditeur puisque celui-ci est le seul à connaître la clef utilisée pour le calcul. Si les deux champs ne sont pas identiques le paquet doit être rejeté et l'évènement noté dans le journal du système.

### 1.3.4 Internet Security Association and Key Management Protocol

ISAKMP définit le cadre dans lequel la négociation des SAs et l'échange des clefs pourra se faire. Ce cadre comprend les différents messages nécessaires, leur format, le format des attributs qui les composent, et la séquence dans laquelle ils doivent être échangés. ISAKMP ne définit pas comment l'échange de clef doit se faire, cela sera précisé par l'application ou le protocole qui se basera sur ISAKMP, comme nous le verrons pour IKE. ISAKMP reste en dehors de tout contexte particulier, et se contente de décrire des mécanismes et des messages généraux. Les éléments propres à un contexte d'utilisation donné sont spécifiés dans un document appelé Domaine d'Interprétation (DOI). Dans le cas d'IPsec le DOI est spécifié dans [17]. Le protocole ISAKMP est décrit dans [18].

#### 1.3.4.1 Messages ISAKMP

Un message ISAKMP aura toujours la même structure, et pourra contenir un ou plusieurs *payloads*[16]. Tout message commencera par une entête standard illustrée par la figure 1.11. Cette entête contient les champs suivants :

**Cookies** Il y a deux champs de ce type, un pour l'initiateur de l'échange et un pour le récepteur. Ces deux champs sont utilisés en conjonction avec l'identifiant du message en cours d'échange pour gérer l'état de l'échange en cours. Plus de détails seront donnés sur le sujet plus loin dans cette section.

**Prochain payload** indique le type de payload qui suivra cette entête.

**Numéro de version majeur et mineur** indique la version du protocole ISAKMP utilisé pour cet échange.

**Type d'échange** indique de quel type d'échange ISAKMP il s'agit.

**Drapeaux** est un masque binaire dont seulement 3 bits sont utilisés actuellement : un drapeau d'encryption indiquant que les payloads suivant sont chiffrés, un drapeau commit indiquant qu'un des hôtes veut recevoir une notification à la bonne fin de l'échange, et un drapeau utilisé pour indiqué l'utilisation de la récupération de clef dans ISAKMP.

**Identifiant du message** identifiera le message de manière univoque.

**Taille du message** indique la taille totale de tout le message, entête comprise.

ISAKMP définit 13 types de *payload*. Chaque *payload* aura la même structure, il commencera par une entête de *payload* indiquant sa taille, ainsi que le type du *payload* qui suivra. Les informations de ce *payload* seront composées de un ou plusieurs attributs, de taille fixe ou variable. On distingue donc deux types d'attributs : les attributs basiques dont la taille totale ne dépasse pas 32 bits, et les attributs de taille variable pour ceux dont la taille totale excède 32 bits. Le type de l'attribut sera indiqué par son premier bit. Le deuxième champ indique le type de l'attribut en fonction de son contenu. Le troisième champ contiendra la valeur proprement dite pour les attributs basiques (16 bits, avec padding éventuel), et la longueur de la valeur pour les attributs à taille variable. Dans ce dernier cas la valeur viendra derrière, comme on peut le voir sur la figure 1.12.

Un message est donc une chaîne commençant par l'entête de message ISAKMP, suivi des *payloads* successifs. Ces *payloads* sont eux-mêmes composés d'un ou plusieurs attributs, de taille fixe ou variable.

[16] le terme anglais utilisé dans la documentation est "payload", ce qui peut se traduire par "contenu". Comme cela pourrait conduire à des ambigüités nous utiliserons le terme anglais dans tout ce document.

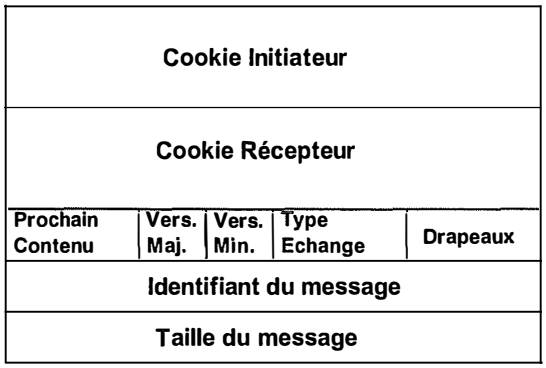



FIG. 1.11 – Entête de message ISAKMP

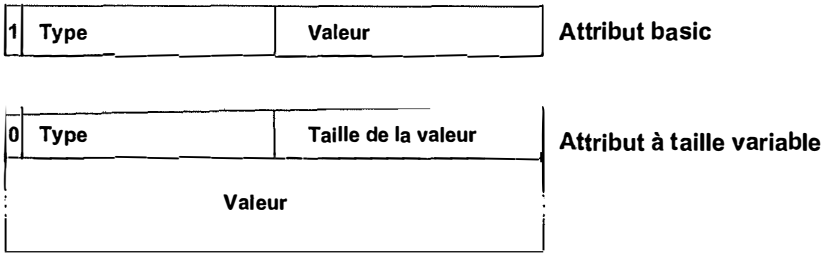



FIG. 1.12 – Structure des attributs

Passons en revue les différents types de *payload* utilisés par ISAKMP :
- **Hash** contient le résultat d'une fonction de hachage.
- **Signature** contient une signature numérique
- **"Nonce"** contient des informations pseudo-aléatoires nécessaires pour l'échange (génération de clef et authentification)
- **Vendeur ID** contient des données propres à un vendeur lui permettant de reconnaître ses implémentations
- **Echange de clef** contient par exemple une valeur publique d'un échange Diffie-Hellman[17] qui permettra l'établissement d'une clef commune.
- **SA** définit une SA, soit pour ISAKMP, soit pour un autre protocole comme IPsec. Ce payload sera toujours accompagné d'une ou plusieurs propositions, elles-mêmes contenant une ou plusieurs transformations chacune. La structure du message contenant la proposition d'une SA ISAKMP doit respecter les conditions suivantes :
  - un *payload* SA ne peut inclure qu'un seul *payload* Proposition (car on ne négocie qu'une seule SA).
  - Le *payload* Proposition peut inclure un ou plusieurs *payloads* Transformation.

  Ces conditions sont valables pour les SA ISAKMP, mais pas pour les SA négociées pour d'autres applications (comme IPsec).
  Le champ "next header" du payload de type SA ne pointera pas sur les *payloads* de proposition ou de transformation inclus, car ceux-ci font partie du *payload* SA.
- **Certificat** contient un certificat qui sera utilisé pour l'authentification (ISAKMP ne

[17] La description de l'algorithme d'échange Diffie-Hellman se trouve en annexe.

définit pas de méthode d'authentification)
- **Requête de certificat** permet de demander à l'autre partie qu'elle envoie son certificat
- **Identité** contient les informations indentifiant un hôte
- **Proposition** sera toujours envoyé lié à un *payload* SA, contient une proposition des paramètres de la future SA, comme le mode d'authentification, et un ou plusieurs payloads de type transformation.
- **Transformation** sera lié à une proposition. Il contient les paramètres relatifs à un algorithme de chiffrement ou d'authentification.
- **Notification** permet de communiquer à l'autre partie des informations en cas d'erreur par exemple.
- **Effacement** permet de demander l'effacement d'une SA à la fin de sa vie.

#### 1.3.4.2 Echanges ISAKMP

ISAKMP définit 5 échanges permettant de négocier une SA. Chaque échange est une séquence de messages envoyés entre l'initiateur et le récepteur, et répond à des exigeances différentes. L'échange de base permet la négociation d'une SA avec authentification et échange de clef, cela en 4 messages. L'échange avec protection d'identité offre en plus la confidentialité de l'identité des parties négociantes, et consiste en 6 messages. L'échange avec authentification uniquement permet la négociation d'une SA sans échange de clef, avec authentification ; cet échange pourra être utilisé sous la protection d'une connexion sécurisée préalable. L'échange agressif a pour objectif de réduire au maximum le nombre de messages échangés en combinant le plus d'informations possible dans chaque message. Il ne comporte que 3 messages, et les identités ne sont pas protégées. Enfin, l'échange informatif comporte un message unique dont le but est de notifier une erreur ou une demande d'effacement d'une SA à l'autre partie.

### Caractéristiques communes

Certaines étapes sont communes à tous les échanges, en particulier pour les deux premiers messages. C'est en effet à ce moment-là que sont échangés les Cookies de l'entête dont nous avons parlé plus haut. Que sont les cookies ? A quoi servent-ils ? Comment sont-ils construits ?

Le terme cookie fait référence aux gateaux américains bien connus, dont chaque cuisinière a sa recette bien à elle, ce qui fait qu'il n'y a pour ainsi dire pas deux cookies vraiment identiques. Dans notre cas, le cookie est une chaîne de 64 bits généré par chacune des parties à partir de certaines informations connues d'elle seule, chacun ayant donc sa "recette". Ainsi il sera théoriquement impossible que deux hôtes puissent générer le même cookie. ISAKMP ne spécifie pas de technique de création, toutefois la technique décrite dans [12] est recommandée : le cookie est obtenu par l'application d'une fonction de hashage rapide sur un ensemble de données composé de l'adresse source et de destination, des ports UDP de source et de destination, d'une valeur aléatoire secrète générée par l'hôte, et de la date et l'heure de génération. De cette façon nous obtenons un cookie unique pour chaque SA, lié à l'adresse IP de l'hôte générateur, et que seul celui-ci pourra regénérer pour vérification. L'utilisation d'une fonction de hashage rapide assurera une consommation minimale du processeur.

1. L'initiateur crée le message initial, avec le cookie qu'il crée pour cette nouvelle négociation de SA, et initialise le champ du cookie du récepteur à zéro. Le cookie généré ainsi que l'adresse du récepteur pourront être stockés pour la vérification ultérieure.

2. Le récepteur crée lui aussi un cookie, génère une "proto-SA" contenant les deux cookies comme identification, et envoie sont message avec le cookie nouvellement généré ainsi qu'une copie du cookie de l'initiateur.
3. L'initiateur, à la réception de la réponse, vérifie que le cookie présent dans le message reçu est identique à celui qu'il avait généré. Si c'est le cas la SA est crée avec les deux cookies pour identifiant. Les messages suivant de l'échange contiendront aussi les deux cookies.
4. Le récepteur, recevant le troisième message de l'échange, pourra à son tour vérifier que le cookie dans le champ du récepteur est bien le même que celui qu'il avait créé, et que donc la demande était authentique. A ce stade aucune opération coûteuse en calcul, comme le calcul de valeur Diffie-Hellman (voir en annexe) par exemple, n'a encore eu lieu. Donc dans le cas d'une attaque par envoi de paquets avec de fausses adresses ces premiers paquets qui resteront sans suite ne mobiliseront pas (ou très peu) de ressources. Ce n'est qu'après réception de ce troisième message que la validité de l'adresse IP de l'initiateur est prouvée, et que les opérations plus coûteuses peuvent avoir lieu. Le récepteur peut alors activer la "proto-SA" en SA et poursuivre l'échange.

### Echange de base

L'échange de base est conçu pour transmettre les informations d'échange de clef et d'authentification en même temps, permettant ainsi de réduire le nombre de messages aller-retour transmis. De par ce fait même les identités ne peuvent pas être protégées puisqu'au moment de leur transmission il n'y a pas encore eu d'échange de matériel cryptographique nécessaire à l'établissement d'une clef commune.

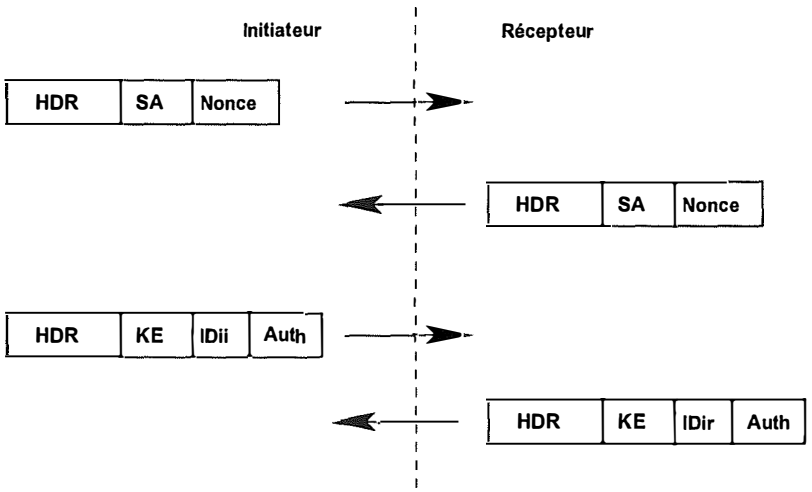



FIG. 1.13 – Echange de base ISAKMP

La figure 1.13 schématise l'échange. Lors des deux premiers messages, les deux parties échangent, en plus de leur cookies, les propositions pour la future SA, ainsi qu'une valeur produite par une fonction pseudo-aléatoire (Nonce). La réponse du récepteur contiendra la proposition choisie par le récepteur en fonction de la politique de sécurité de celui-ci. Le troisième message contient les informations nécessaires à l'établissement d'une clef de session,

les informations d'indentité et celles nécessaires à l'authentification. Du fait que l'identité est transmise en même temps que les informations d'échange de clef il est impossible de la protéger du regard des tiers. Ce n'est qu'à l'issue du quatrième message que la clef de session est établie des deux côtés et que la SA est ainsi pleinement établie et peut être utilisée pour protéger les échanges futurs.

### Echange avec protection d'identité

La protection de l'identité au sein de cet échange se fait au prix de deux messages supplémentaires, soit un total de 6 messages. Le matériel d'échange de clef est échangé avant les données d'identification et d'authentification, de cette façon elles peuvent être protégées grâce à la clef fraichement négociée.

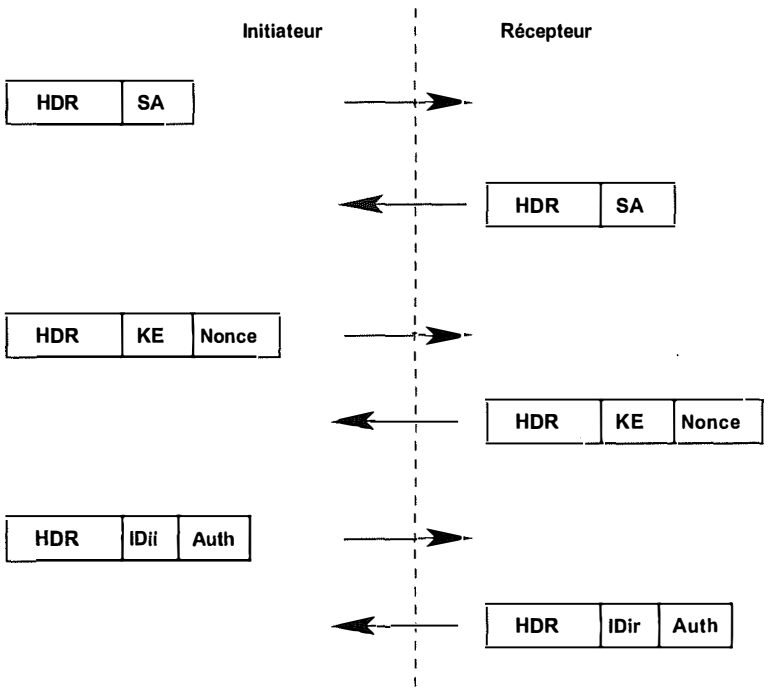



FIG. 1.14 – Echange ISAKMP avec protection d'identité

Les deux premiers messages ne contiennent que la proposition de SA et le choix par le récepteur, comme l'illustre la figure 1.14. Les deux messages suivants contiennent les données relatives à l'échange de clef. A partir de cette étape une clef commune à été négociée et le reste de l'échange peut être protégé. Les deux derniers messages contiennent les données d'identité et d'authentification.

### Echange avec authentification uniquement

Cet échange pourra être utilisé pour négocier une SA dans le cas d'une SA ISAKMP déjà existante. L'échange poura donc être protégé par cette SA. Il n'y a donc pas d'échange de matériel pour l'établissement d'une clef, mais uniquement la proposition de SA (et la réponse

dans l'autre sens) ainsi que les informations d'authentification. Cet échange se fait en trois messages, voir figure 1.15.

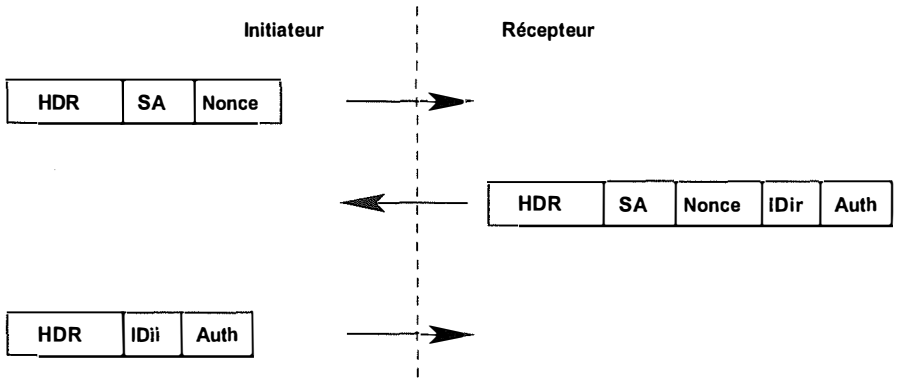



FIG. 1.15 – Echange ISAKMP avec authentification seulement

Le premier message contient la proposition de SA par l'initiateur. Le récepteur répondra en joignant à la proposition de SA choisie ses données d'identité et d'authentification. Le troisième message contient l'identité et l'authentification de l'initiateur.

### Echange agressif

Le but de cet échange est de réduire le nombre de messages au maximum tout en effectuant une négociation de SA complète, avec échange de clef. La protection de l'identité n'est pas assurée. Cet échange se compose de trois messages seulement. Examinons la figure 1.16.

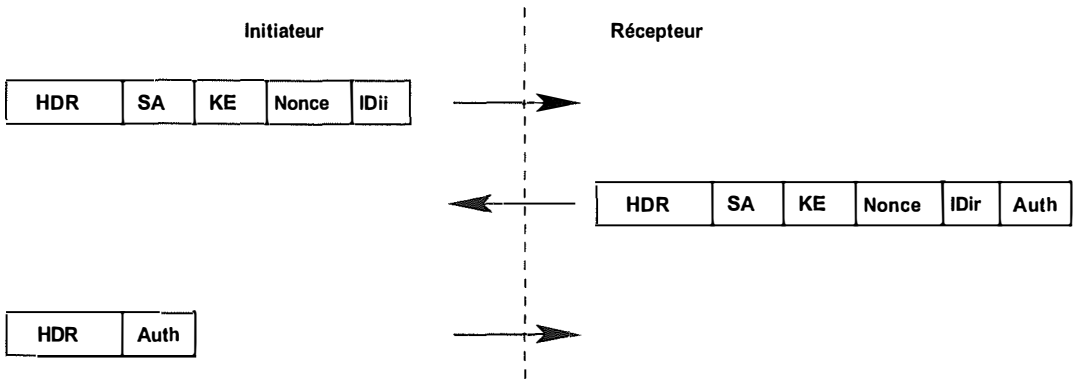



FIG. 1.16 – Echange ISAKMP Agressif

Le premier message contient, en plus de la proposition de SA, le matériel d'échange de clef et l'identité de l'initiateur. La proposition de SA ne peux contenir qu'une seule proposition et une seule transformation.

Le deuxième message contiendra les mêmes informations de la part du récepteur, avec en plus les informations d'authentification. Ces informations seront déjà protégées puisqu'à ce stade le récepteur dispose déja des informations pour calculer la clef de session commune.

Le dernier message permet à l'initiateur d'authentifier l'échange auprès du récepteur, grâce à la méthode d'authentification convenue lors de l'échange qui se termine.

Il est à noter que dans ce cas les opérations coûteuses en calcul par les deux parties n'attendent pas le troisième message comme c'est le cas pour les autres échanges. L'échange agressif présente donc un danger vis-à-vis des attaques en déni de service décrites plus haut dans cette section.

#### Echange informatif

L'échange informatif consiste en un seul message permettant par exemple de notifier une erreur survenue pour un message donné. Il pourra aussi s'agir d'un message d'effacement pour dire à l'autre partie d'effacer une SA de sa base de donnée . Ce type de message devra être transmit sous la protection d'une SA ISAKMP si elle existe.

### 1.3.4.3 Phases ISAKMP

ISAKMP définit 2 phases. La phase 1 consiste en l'établissement d'une SA ISAKMP, en utilisant l'échange de base, avec protection d'identité, ou l'échange agressif. Une fois cette première phase accomplie la Phase 2 consistera en l'établissement d'une SA pour l'application "cliente" d'ISAKMP, l'échange étant protégé pour la SA de la Phase 1. Pour la Phase 2 l'échange avec authentification seulement pourra être utilisé.

### 1.3.5 Internet Key Exchange

Le protocole de négociation de SA et d'échange de clef pour IPsec s'appelle IKE. IKE est un protocole hybride, construit sur ISAKMP et empruntant certaines techniques à Oakley (échange de clef) et à SKEME (authentification par chiffrement à clef publique). Nous allons maintenant voir comment ces différents éléments sont intégrés, et comment le protocole IKE fonctionne.

#### 1.3.5.1 Domaine d'Interprétation

Comme nous l'avons vu dans la section consacrée à ISAKMP, certains détails sont laissés indéfinis, dans un but de souplesse par rapport aux utilisations d'ISAKMP. Ces "zones d'ombres" doivent être définies pour chaque utilisation particulière d'ISAKMP, dans un document appelé Domaine d'Interprétation[18]. Ce document défini les conventions de nomage des identifiants au sein du DOI, l'interprétation du champ de situation du *payload* SA, l'ensemble des politiques de sécurité applicables, la syntaxe des attributs et des payloads propres au DOI, des types d'échange supplémentaires si nécessaire, ainsi que des types de message de notification supplémentaires si nécessaire. Dans le cas d'IPsec il n'y a pas d'échange supplémentaire défini dans le DOI.

Le DOI pour IPsec est défini dans [17].

#### 1.3.5.2 Echanges et Phases IKE

IKE reprend les 2 phases définies par ISAKMP. La Phase 1 est celle pendant laquelle la SA IKE est négociée. Cette négociation utilise les mécanismes généraux définis par ISAKMP. C'est pendant la Phase 2 que la SA IPsec sera négociée, sous la protection de la SA IKE établie en Phase 1. Une fois la Phase 1 terminée et la SA IKE établie, elle pourra être utilisée pour établir autant de SA IPsec que l'on veut.

IKE défini plusieur méthodes d'authentification qui peuvent être utilisées au cours de la Phase 1. Le contenu des échanges reste le même quel que soit le mode choisi, seules les données du payload d'authentification en dépendront.

##### Phase 1 et modes d'authentification

Les deux échanges IKE de la phase 1 sont des instances de l'échange avec protection d'identité et de l'échange agressif d'ISAKMP. Le nombre de messages échangés doit absolument être respecté, si certaines informations supplémentaires doivent être transmises (par exemple un certificat) elles doivent être transmises sans message supplémentaire, avec un des messages existant.

Selon le mode d'authentification convenu la clef maîtresse (à partir de laquelle les autre cles sont dérivées) sera calculée d'une manière différente, mais la dérivation des clefs spécifiques se fera toujours de la même façon. Le principe général consiste à calculer une première valeur à partir d'une série de données qui est différente suivant le mode d'authentification. Ensuite trois clefs sont dérivées de cette valeur : une clef pour l'authentification, une clef pour le chiffrement et une clef qui servira dans l'établissement des SA IPsec.

[18] en anglais Domain Of Interpretation, DOI en abrégé.

Voici les formules de calcul telles qu'on les trouve dans [19], avec un rappel de la notation utilisée :

$SKEYID$ Chaîne calculée à partir de données secrètes par les parties en négociation. Sera utilisée pour dériver les autres clefs.

$SKEYID_d$ Chaîne qui sera utilisée pour dériver les clefs pour la négociation des SA non-IKE (IPsec dans notre cas).

$SKEYID_a$ Chaîne qui sera utilisée pour l'authentification de l'échange en cours.

$SKEYID_e$ Chaîne qui sera utilisée pour assurer la confidentialité de l'échange en cours.

$Ni_b, Nr_b$ Chaîne pseudo-aléatoire générée par l'initiateur et le récepteur, respectivement.

$g^{xi}, g^{xr}$ valeur publique Diffie-Hellman de l'initiateur et du récepteur, respectivement.

$g^{xy}$ valeur résultant de l'échange Diffie-Hellman[19].

$prf(clef, msg)$ Fonction pseudo-aléatoire avec clef, le plus souvent une fonction de hashage avec clef. Le résultat d'une telle fonction ressemble à des données pseudo-aléatoires, et il est techniquement impossible de calculer la valeur de départ à partir du résultat. La fonction à utiliser est celle négociée dans les deux premiers messages d'un échange, et mentionnée dans la proposition acceptée par le récepteur.

$CKY_I, CKY_R$ Cookie de l'initiateur et du récepteur.

$HASH_I, HASH_R$ Résultats de fonction de hashage calculés par l'initiateur et le récepteur.

Formules de calcul :

| | |
|---|---|
| Authentification par signature : | $SKEYID = prf(Ni_b \| Nr_b, g^{xy})$ |
| Authentification par chiffrement à clef public : | $SKEYID = prf(hash(Ni_b \| Nr_b), CKY_I \| CKY_R$ |
| Authentification par clef partagée : | $SKEYID = prf(ClefPartage, Ni_b \| Nr_b)$ |

De la valeur SKEYID, on dérive 3 clefs :

$SKEYID_d = prf(SKEYID, g^{xy} \| CKY_I \| CKY_R \| 0)$
$SKEYID_a = prf(SKEYID, SKEYID_d \| g^{xy} \| CKY_I \| CKY_R \| 1)$
$SKEYID_e = prf(SKEYID, SKEYID_a \| g^{xy} \| CKY_I \| CKY_R \| 2)$

Les valeurs utilisées pour l'authentification (Hash) sont calculées comme suit :

$HASH_I = prf(SKEYID, g^{xi} \| g^{xr} \| CKY_I \| CKY_R \| SAi_b \| IDii_b)$
$HASH_R = prf(SKEYID, g^{xr} \| g^{xi} \| CKY_R \| CKY_I \| SAi_b \| IDir_b)$

Nous verrons dans la suite comment les différentes méthodes d'authentification utilisent ces valeurs.

Examinons les quatres modes d'authentification, et leur impact sur les deux types d'échange de la phase 1 :

1. **Authentification par signature.**

   L'authentification de l'échange se fera par signature de $HASH_I$ par l'initiateur, et de $HASH_R$ par le récepteur. Chaque partie signera avec sa clef privée, et l'autre partie vérifiera grâce au certificat qui aura éventuellement été transmis dans un *payload* de certificat joint au message contenant la signature. Le certificat doit bien sûr être issu d'une autorité de certification reconnue par les deux parties.

   Le payload d'authentification est donc une signature, et le *payload* de type certificat sera éventuellement ajouté avant celui d'authentification.

[19] cfr. Explication du principe du partage de clef Diffie-Hellman en annexe.

2. **Authentification par chiffrement à clef publique.**
   L'authentification se fera en chiffrant les données pseudo-aléatoire ($Ni_b$ pour l'initiateur, $Nr_b$ pour le récepteur) avec la clef publique de l'autre partie. Comme ces données sont utilisées pour la construction de SKEYID, et que cette SKEYID fait elle-même partie des paramètres de calcul des HASH, l'échange des HASH correct suffit à l'authentification. Pour résoudre l'ambiguïté dans le cas où l'une des parties possède plusieurs jeux de clefs-certificats, l'autre partie joindra un hash du certificat utilisé pour le chiffrement. Le payload d'identification sera également chiffré, permettant ainsi une protection d'identitée même dans le cas de l'utilisation du mode agressif. Il est à noté que le payload d'identification est transmis dans le troisième et quatrième message dans le cas du mode principal, au lieu du cinquième et sixième message comme c'est normalement le cas. Il est à noter aussi que ce type d'authentification nécessite que les deux parties possèdent au préalable la clef publique de l'autre partie.
3. **Authentification par chiffrement à clef publique - méthode révisée.**
   L'inconvénient de la méthode précédente est qu'elle nécessite 4 opérations de chiffrement-déchiffrement à clef publique, or ce type de calcul est très coûteux. La méthode révisée réduit ce nombre à deux, tout en gardant l'avantage de la protection d'identité dans les deux modes. Pour cela la valeur $Ni_b$ (dans le cas de l'initiateur) sera toujours chiffrée avec la clef publique du récepteur, mais les *payloads* d'identification, d'échange de clef et (éventuellement) de certificat transmis seront chiffrés en utilisant l'algorithme symétrique négocié dans la SA et une clef dérivée du $Ni_b$. Le chiffrement symétrique est beaucoup moins coûteux en calcul que le chiffrement asymétrique.

   Dans le cas de cette méthode, seul l'initiateur devra connaître au préalable la clef publique du récepteur, puisqu'il peut communiquer son certificat à ce dernier. De même que dans la méthode précédente, un hash du certificat peut être transmis, ce *payload* devant obligatoirement être le premier après l'entête ISAKMP.

   Comme pour la méthode précédente, la capacité d'une partie à reconstituer un hash basé en partie sur des données chiffrées avec clef publique prouve que cette partie est bien en possession de la clef privée correspondante.
1. **Authentification par clef commune.**
   C'est le cas le plus simple, il suppose qu'une clef à été préalablement partagée par un mécanisme externe au protocole. Les échanges, avec protection d'identité ou agressif, ne comportent rien de particulier par rapport à la description donnée par ISAKMP. Une restriction existe dans le cas du mode principal (protection d'identité) : l'identification doit se faire sur l'adresse IP uniquement[20]. En effet les données d'identité sont nécessaires pour le calcul du Hash d'authentification, qui doit être calculé avant réception des données d'identitifcation du récepteur par l'initiateur.

### Phase 2

Le Mode Rapide (Quick Mode) de la phase 2 est un nouvel échange, et n'est pas décrit par ISAKMP. Cet échange est toujours lié à un échange de Phase 1 préalable, duquel il dérivera les clefs utilisées pour la négociation de la SA non-IKE. Dans la suite nous supposerons qu'il s'agit toujours d'une SA IPsec.

[20] Cela est vrai au moins pour le récepteur, ce dernier ayant reçu l'adresse IP de l'initiateur au cours de l'échange doit pouvoir calculer $HASH_R$ sans avoir besoin de connaître l'IP de l'initiateur au préalable.

Avant de poursuivre, définissons un nouveau terme :

**Perfect Forward Secrecy** Cette propriété concerne la façon dont sont dérivées les clefs pour une nouvelle SA non-IKE. Deux approches sont en effet possibles :

- Les clefs de la nouvelle SA peuvent être dérivées des clefs déjà négociées au cours de la phase 1 pour la SA ISAKMP.
- Un nouvel échange de clef (le *payload* KE dans les schémas d'échange) peut être effectué au cours de la phase 2, rendant les clefs générées indépendantes de celles de la phase 1.

La deuxième technique assurera que les clefs de la phase 2 sont indépendantes non seulement de celles de la phase 1, mais également que les clefs successives négociées pour une nouvelle instance de SA entre les deux même hôtes (à l'expiration d'une SA) sont indépendantes. De cette façon, si la clef d'une SA non-IKE est compromise, la clef suivante ne le sera pas car un nouvelle echange de clef aura lieu pour sa génération, elle ne dérive donc pas des mêmes informations secrètes que la clef compromise.

C'est cette propriété d'indépendance des clefs successives grâce à un nouvel échange de clef qu'on appelle Perfect Forward Secrecy (abrégé par *pfs*).

Une des différences entre la SA IKE négociée entre deux hôtes et les SA non-IKE qui seront négociées grâce à celles-ci, c'est que ces dernières peuvent être négociées pour le compte d'autre hôtes. C'est ce qui se passe dans le cas d'un tunnel entre deux passerelles : les deux passerelles IPsec vont d'abord négocier une SA IKE, et ensuite négocier la ou les SA IPsec qui seront utilisées pour protéger les connexions entres les hôtes des réseaux locaux communicant à travers le tunnel. En fournissant les identités des hôtes qui utiliseront la SA, le récepteur sera en mesure de vérifier si ces connexions sont authorisées par sa politique de sécurité et si la ou les SA peuvent être créées ou non. Si aucune information d'identité n'est fournie, la négociation se fera implicitement pour compte des deux hôtes ayant établi la SA IKE en phase 1.

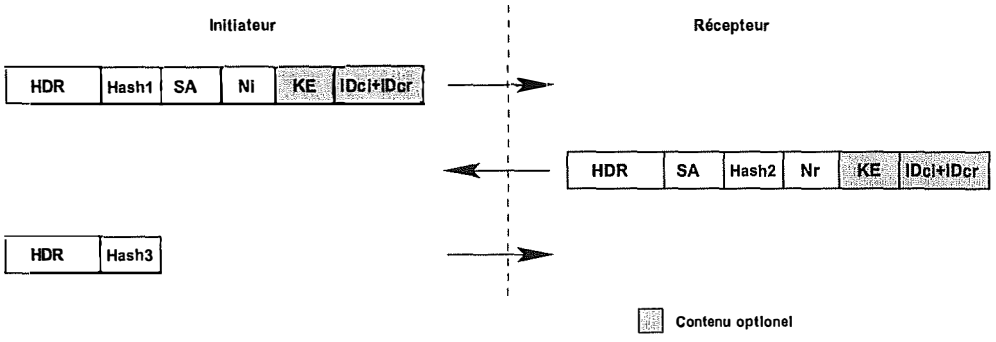



FIG. 1.17 – IKE : Quick Mode

Examinons l'échange illustré par la figure 1.17. Il est composé de trois messages. Ces messages seront tous protégés par la SA IKE établie en phase 1.

Dans le premier message, l'initiateur de la phase 2 (il peut s'agir de n'importe quel hôte ayant négocié la SA IKE) communique une ou plusieurs propositions de SA, avec une chaîne pseudo-aléatoire Ni. Un payload de type Hash placé juste après l'entête assure l'authentification du message. Si la Perfect Forward Secrecy est spécifiée, un payload d'échange de clef

fera également partie du message. Dans le cas où la ou les SA en négociation sont établies pour compte d'autres hôtes, les identitées des l'hôtes source et de destination seront jointes au message.

Le second message contient la réponse du récepteur, avec la ou les propositions de SA acceptées, un nombre pseudo-aléatoire et un Hash pour authentifier le message. Les payloads d'échange de clef et d'identité devront être présents si ils étaient présent dans le premier message.

Le troisième message conclu l'échange, en apportant la preuve que l'initiateur a bien reçu la réponse du récepteur et que la ou les SA sont bien établies.

Les formules de calcul des 3 hashs de l'échange, selon la même notation que celle décrite plus haut, sont :

$HASH1 = prf(SKEYID_a, MId|SA|Ni[|KE][|ID_{ci}|ID_{cr}])$
$HASH2 = prf(SKEYID_a, MId|Ni_b|SA|Nr[|KE][|ID_{ci}|ID_{cr}])$
$HASH3 = prf(SKEYID_a, 0|MId|Ni_b|Nr_b)$

A l'issue de la négociation, la nouvelle clef sera calculée différement suivant que la propriété de PFS est exigée ou non :

- Sans PFS il n'y a pas d'échange clef supplémentaire, et la formule de la clef est
  $CLEF = prf(SKEYID_d, protocole|SPI|Ni_b|Nr_b)$
- Avec PFS, la formule de la clef est
  $CLEF = prf(SKEYID_d, g(qm)^{xy}|protocole|SPI|Ni_b|Nr_b)$

où $g(qm)^{xy}$est le secret partagé résultant de l'échange de clef basé sur l'algorithme Diffie-Hellman.

## 1.4 IPsec - évolutions et perspectives

L'Internet Engineering Task Force continue les recherches pour améliorer les protocoles existants et prépare déjà la prochaine génération des protocoles IPsec. Le processus de développement est démarré depuis la fin de l'année 2001, et les cahiers des charges pour les nouvelles versions sont déjà disponibles sur le site de l'IETF.

Les travaux qui semblent les plus importants visent à produire une version 2 du protocole IKE, appelé dans les document IKE v2 ou "Son of IKE". La version 1 d'IKE comporte plusieurs défauts, parmi lesquels on cite en premier sa complexité. Les caractéristiques requises pour la future version ont déjà été écrites, et plusieurs propositions ont été faites par les différents groupes de chercheurs participant au développement. La date de soumission de la proposition de standard est fixée au mois de décembre 2002.

Nous n'en dirons pas plus sur ces évolutions futures, nous encourageons le lecteur intéressé à consulter les documents officiels disponibles sur le site web de l'IETF : http ://www.ietf.org/html.charte charter.html.

# Chapitre 2

# Implémentation IPsec OpenBSD

Dans ce chapitre nous allons examiner l'implémentation IPsec du système d'exploitation OpenBSD, qui sera utilisé pour la passerelle IPsec utilisée dans la solution exposée dans le chapitre 3.

OpenBSD est un système de type Unix dérivé de la distribution 4.4BSD-lite, comme c'est également le cas pour FreeBSD, NetBSD ou SunOS (cette liste n'est pas exhaustive). Le choix s'est porté sur OpenBSD pour la mise en oeuvre d'une passerelle VPN parce que le développement de ce projet est caractérisé par une approche proactive de la sécurité, comme le souligne le slogan "secure by default"[1]. Ce système s'est d'ailleurs taillé une solide réputation dans le milieu de la sécurité réseau. Le site officiel est référé en [20].

Nous allons examiner les différents éléments logiciels remplissant les différentes fonctions d'IPsec sur cette plate-forme, en faisant correspondre ces éléments aux composants de l'architecture théorique vue dans la section précédente.

## 2.1 Intégration avec la pile TCP/IP

Une grande partie de l'implémentation d'IPsec est intégrée au noyau du système et se fond dans la pile TCP/IP. nous sommes bel et bien dans le cas d'une intégration au système d'exploitation, et non ce qu'on appelle dans le jargon anglais un "Bump In The Stack"[2](abrégé par BITS). L'intégration de type BITS est utilisée dans les cas où les développeur n'ont pas accès au code de la pile réseau, et sont donc obligés d'utiliser les interface disponible avec la couche IP pour connecter leur implémentation qui viendra en général se placer après celle-ci. Dans le cas de l'implémentation OpenBSD l'intégration est totale, le code source de l'intégralité du système étant complètement ouvert et modifiable. Une intégration complète permet de mettre en oeuvre tous les modes spécifiés dans la définition des protocoles, alors que les implémentation de type BITS sont souvent limitées au mode d'encapsulation tunnel de par leur intégration partielle et leur position "en-dessous" de la couche IP, juste au dessus de la couche d'accès au réseau physique.

[1] en français "sécurisé par défaut", ce qui signifie concrètement que le code du système est revu pour corriger les failles potentielles, et que la configuration du système après l'installation de base est minimale quant aux services ouverts et accessibles de l'extérieur, n'offrant par conséquent que peu ou pas de prise aux attaques réseau.

[2] BITS, que l'on pourrait traduire par "bosse dans la pile". Cela par opposition à l'implémentation de type "Bump In The Wire" qui regroupe l'utilisation de matériel spécialisé qui se place en aval du système sur le réseau.

Reprenons le schéma général de l'architecture IPsec, nous allons faire correspondre les éléments de l'architecture théorique avec les éléments faisant partie de la pile réseau d'OpenBSD (figure 2.1). Sur ce schéma sont représentées les différents composants logiques de l'architecture IPsec, dans le contexte d'une implémentation TCP/IP classique. Les élément d'IPsec sont avec la couche IP, comme l'indique le cadre en pointillé. Les éléments propres à IPsec implémentés dans le noyau sont représentés en couleur. l'implémentation du protocole IKE est un programme qui tourne dans l'espace utilisateur, et ne fait pas partie du noyau du système. La base de donnée des politique de sécurité est un cas hybride : une partie des règles de politique de sécurité fait partie du noyau, et il y a également une série de règles utilisées par le démon IKE qui sont gérées par un programme séparé. L'interface spéciale *enc* n'est pas un composant d'IPsec au sens des spécifications, mais est un élément important de l'implémentation IPsec d'OpenBSD. Le programme de firewall (Packet Filter/NAT) fait quand à lui partie de la distribution standard du système, et nou l'avons mis sur ce schéma car il joue un rôle important dans la mise en oeuvre d'une passerelle basée sur IPsec.

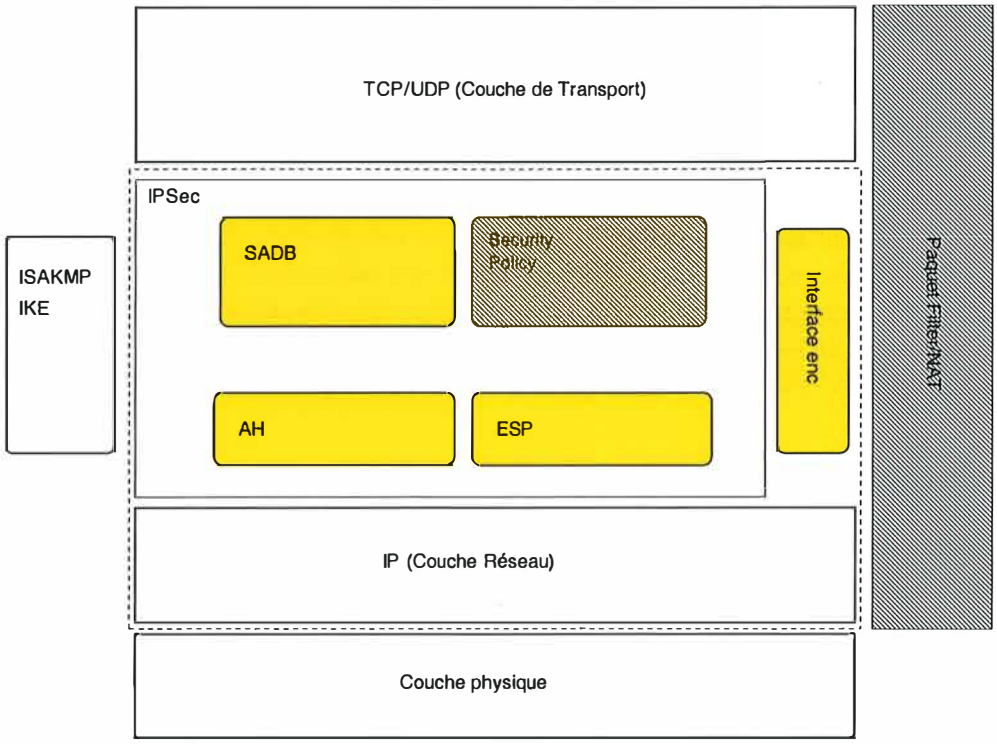



FIG. 2.1 – Architecture IPsec - Éléments de la pile TCP/IP

### 2.1.1 AH et ESP

Ces deux protocoles essentiels pour la communication par IPsec font partie du noyau, à côté (si l'on peut dire) des autres protocoles de la pile réseau. Comme nous l'avons déjà vu dans la partie précédente, le protocole AH offre les services d'intégrité des données et d'authentification, le protocole ESP offre, en plus de ces mêmes services, la confidentialité des données. Il y a cependant une différence entre l'intégrité offerte par AH qui couvre une partie

du contenu de l'en-tête IP, alors que ESP ne porte que sur les données transportées.

L'implémentation de ces protocoles sous OpenBSD peuvent être utilisés avec une série d'algorithmes de chiffrement et d'authentification, avec des longueurs de clef variables suivant les algorithmes. Le tableau 2.1 reprend la liste des algorithmes supportés :

| Algorithmes | Longueur de clef | Phases d'utilisation |
|---|---|---|
| MD5 | 56 bits | I, II |
| SHA | | I, II |
| RIPEMD | 160 bits | II |
| | | |
| DES | 56 bits | I, II |
| 3DES | 168 bits | I, II |
| BLF (blowfish) | variable - 160 bits recommandé | I, II |
| CAST | variable - 128 bits maximum (recommandé) | I, II |
| AES | | II |

TAB. 2.1 – Algorithmes d'authentification et de chiffrement supportés sous OpenBSD

Pour rappel, la spécification originale du protocole ESP impose au minimum le support de l'algorithme DES pour le chiffrement. Hors de nos jours cet algorithme n'est plus suffisamment résistant pour offrir une sécurité décente, il existe des machines spécialisées permettant de craquer le chiffrement DES en quelques heures, cela étant lié (entre autre) à la petite taille de clef utilisée. On recommande donc d'utiliser au minimum l'algorithme 3DES.

Les protocoles ESP et AH seront invoqués avec des paramètres liés aux différentes connexions à servir, ces paramètres étant contenus dans les SA correspondantes. On aura donc différents algorithmes à utiliser, avec différentes clefs.

### 2.1.2 SADB & SPD

Les paramètres à utiliser par AH et ESP seront lus dans la SADB qui se trouve elle aussi dans l'espace mémoire du noyau.

Il existe un utilitaire permettant de gérer la SADB manuellement, donnant donc la possibilité de créer et de détruire des SA.

Il est également possible de lire les SA contenues dans la SADB, en lisant le fichier /kern/ipsec. Ce fichier fait partie d'un système de fichier spécial donnant accès en lecture aux paramètres du noyau, il doit être monter avant de pouvoir y accéder. Examinons l'exemple de la figure 2.2 :

Cet exemple contient les informations des deux SA d'une même connexion, l'une sortante et l'autre entrante. Examinons les informations affichées :

- La première ligne relative à chaque SA reprend les éléments identifiant celle-ci de manière unique : les SPI, l'adresse de destination et le numéro du protocole (ESP = 50, AH = 51)
- l'âge de la SA. Nous voyons que ces deux SA ont été créées simultanément lors de la négociation de la connexion
- l'adresse IP source

```
Hashmask : 31, policy entries : 2
SPI = 3c408de7, Destination = 213.177.154.200, Sproto = 50
Established 307 seconds ago
Source = 193.1.1.1
Flags (00001186) = <tunneling>
Crypto ID : 1
xform = <IPsec ESP>
Encryption = <CAST-128>
Authentication = <HMAC-SHA1>
20743 bytes processed by this SA
Expirations :
Hard expiration(1) in 293 seconds
Soft expiration(1) in 233 seconds
Hard expiration after 4194304 bytes
Soft expiration after 3774873 bytes
SPI = a79c6dfb, Destination = 193.1.1.1, Sproto = 50
Established 307 seconds ago
Source = 213.177.154.200
Flags (00001186) = <tunneling>
Crypto ID : 2
xform = <IPsec ESP>
Encryption = <CAST-128>
Authentication = <HMAC-SHA1>
13576 bytes processed by this SA
Expirations :
Hard expiration(1) in 293 seconds
Soft expiration(1) in 233 seconds
Hard expiration after 4194304 bytes
Soft expiration after 3774873 bytes
```

FIG. 2.2 – exemple d'exécution de "cat /kernfs/ipsec"

- une série de drapeaux (flags) contenant des informations sur le mode de la SA. Dans notre exemple il s'agit de deux SA en mode tunnel
- Crypto ID
- le type de transformation, c'est-à-dire en fait le type de traitement appliqué aux données. Nous sommes bien en présence d'une connexion IPsec utilisant le protocole ESP
- ensuite nous trouvons les algorithmes de chiffrement et d'authentification/Intégrité
- le compteur du nombre de bytes transmis via la SA
- pour terminer sont affichées les limites souple et rigide d'expiration de la SA, en unité restante par rapport au moment d'affichage de ces informations.

Un fait surprenant dans les entrées de cette SADB : nous n'y trouvons aucune trace des SAs utilisées dans la phase 1 du protocole ISAKMP/IKE. Comment cela se fait-il ? Cela est simplement dû au fait qu'il est possible soit de créer les SA IPsec manuellement, soit de recourir à un autre protocole qu'ISAKMP pour la gestion des SA et des clefs, et donc sans utiliser de SA ISAKMP. Ces SAs ne sont donc pas gérées par l'implémentation IPsec de la pile

réseau, mais par le démon isakmpd dont nous parlerons en détail plus loin dans cette section.

*La SPD* (Security Policy Database) fait aussi partie du noyau. Cette table permet à la pile IP de déterminer si un paquet doit oui ou non être traité par IPsec, ou si il doit être transmis tel quel ou rejeté. Cette table est implémentée comme une table de routage modifiée et est accédée de la même manière, en effet la question est très similaire à celle du routage : "que dois-je faire de ce paquet ?". La réponse est trouvée en cherchant une entrée correspondant aux caractéristiques du paquet (adresse source, destination, port, protocole,...). Dans le cas où le paquet doit être traité par IPsec, et si une SA existe déjà, alors celle-ci sera employée. Si il n'existe pas de SA le paquet est rejeté et le processus de négociation d'une nouvelle SA est déclenché (dans le cas d'une gestion automatique des SA). Il est important de noter que cette SPD ne permet pas de déterminer si une SA peut être établie ou non, elle ne sert qu'au traitement appliqué à chaque paquet IP.

### 2.1.3 L'interface *enc*

La pile réseau BSD comporte une série d'interfaces qui ont pour but de faciliter les interactions avec des programmes externes comme par exemple des pilotes pour certains types de ligne particuliers ou pour certains protocoles (Ex : ppp, transmission sur ligne série, tunneling,...). Ces interfaces ont les mêmes particularités que les interfaces réseau ordinaires, elle offrent les mêmes services et il n'y a pas de périphérique correspondant dans le répertoire /dev.

L'une d'entre elles nous intéresse particulièrement : l'interface *enc*, appelée ainsi à cause du terme "encapsulation". Cette interface est similaire à l'interface loop (interface en boucle) par son implémentation, mais ne donne accès qu'en lecture aux paquets entrants à la sortie du traitement par IPsec, ainsi qu'aux paquets sortants avant leur traitement par le protocole ESP. C'est de là que vient le nom *enc*, parce que cette interface permet de surveiller les paquets traités par ESP avant encapsulation et après dé-encapsulation. Grâce à cela il est possible d'utiliser les facilités de filtrage d'OpenBSD pour appliquer des règles de firewall, ou simplement pour surveiller le trafic protégé par IPsec à l'aide d'outils comme tcpdump. Il suffit de spécifier l'interface enc comme l'interface sur laquelle appliquer les règles ou le dump. Si on veut utiliser un programme de type tcpdump, il faut d'abord activer l'interface avec la commande ifconfig. De cette façon les commandes :

```
#ifconfig enc0 up
#tcpdump -i enc0
```

affichera tout trafic entrant et sortant traité par IPsec sur le terminal.

La séquence de traitement d'un paquet traversant une passerelle OpenBSD est le suivant :

1. le paquet (encapsulé par IPsec) arrive via le réseau et entre par l'interface int1.
2. les règles de firewall/NAT portant sur le trafic entrant de l'interface int1 sont appliquées au paquet.
3. la couche IP reçoit le paquet, vérifie le protocole contenu grâce au drapeau *protocole* de l'entête. Si il s'agit du protocole AH ou ESP, c'est la sous-couche IPsec qui se charge de vérifier quel traitement appliquer dans la Security Policy Database, et le cas échéant quelle SA utiliser pour authentifier et déchiffrer le contenu du paquet (en fonction du protocole et du type de service fourni pour la connexion qui y est attachée). Ensuite le

paquet dé-encapsulé est renvoyé à l'entrée de la pile IP, jusqu'au moment où le paquet résultant est destiné à la couche transport ou doit être envoyé vers le noeud suivant si l'hôte agit comme passerelle. Dans ce dernier cas le paquet est traité comme un paquet sortant.

4. C'est au moment où le paquet est sorti du traitement IPsec qu'il est visible via l'interface enc. Il peut donc être filtré et/ou reporté dans le journal juste avant d'être passé à la suite du traitement.
5. à ce stade le paquet sera soit passé à la couche TCP/UDP si l'hôte est le destinataire final ; mais dans le cas d'une passerelle le paquet sera envoyé vers le LAN via l'interface int2. Il pourra éventuellement être destiné à être injecté dans un nouveau tunnel, dans ce cas il repassera par un traitement IPsec en tant que paquet sortant.
6. les règles de filtrage/NAT seront appliquées au paquet sortant sur l'interface int2 avant que celui-ci ne passe à la couche physique.

NB : en général le NAT est appliqué sur l'interface sortante, le plus souvent pour remplacer l'adresse IP source.

Voyons maintenant le cheminement d'un paquet sortant, à partir de son arrivée dans la couche IP :

1. La couche IP consulte la SPD pour déterminer quel traitement appliquer au paquet, en fonction de l'adresse IP et du port de destination. Si un traitement par IPsec doit être appliqué, et si une SA existe déjà, alors la sous-couche IPsec traite le paquet en fonction des paramètres spécifiés par la SA. Si il n'existe pas de SA, le paquet est mis dans la file d'attente et le processus de négociation d'une nouvelle SA (par le démon IKE) est déclenché. Si nous nous trouvons dans un cas de gestion manuelle des SA le paquet est rejeté.
2. Une fois le paquet traité par IPsec, il est ré-injecté à l'entrée de la sélection par la couche IP, et il pourra éventuellement subir plusieurs traitements successifs par la sous-couche IPsec, jusqu'à ce qu'il soit prêt à être envoyé vers la destination (qui peut avoir changé dans l'entête IP si on utilise un tunnel).

## 2.2 Gestion des SAs : le démon isakmpd

Le démon isakmpd est un programme tournant entièrement dans l'espace utilisateur, et qui prend en charge la gestion des SA et donc des clefs utilisées par les algorithmes de chiffrement dans le cadre des différentes connexions IPsec. C'est lui qui insère les SA dans la SADB qui est maintenue par le noyau. C'est lui aussi qui sera chargé de renouveler ces SA à l'expiration de leur durée de vie, et de détruire les SA lors de la fin d'une connexion. Il implémente le protocole IKE, qui est basé sur le *framework* ISAKMP. La description qui va suivre est inspirée de celle qu'en ont donné les auteurs du démon isakmpd dans [21].

### 2.2.1 Architecture d'isakmpd

Le démon isakmpd est implémenté sous forme d'une application pilotée par évènements et orientée message, c'est l'architecture qui semble le plus adaptée à la nature du protocole ISAKMP/IKE. Il est donc composé d'une boucle principale chargée de gérer des évènements tels que l'arrivée d'un message dans le cadre d'une négociation, ou l'expiration d'une SA. La

conception du démon avait aussi pour but une portabilité aisée sur différents systèmes de type Unix, ainsi que la possibilité d'être utilisé pour d'autres applications qu'IPsec[3].

L'architecture du démon isakmpd est modulaire. Parmi les composants certains servent à matérialiser les concepts du protocole IKE et fournissent les services qui permettent de les manipuler, façon orientée-objet. D'autres modules sont des abstractions des entités externes au démon avec lesquels il doit communiquer. Ces entités sont les composants qui sont propres au système d'exploitation et à l'application qui utilise les services du démon. De cette façon le coeur du programme demeure indépendant du système et même de l'application, le rendant ainsi portable et extensible à d'autres applications qu'IPsec. Dans le cas d'une extension future ou du portage vers une nouvelle plate-forme, seules les modules d'abstraction devront être adaptés. Il est à noter que tous les appels à des fonctions dépendantes du systèmes sont regroupées dans un module spécialement réservé à cet effet. D'autres modules sont purement fonctionnels, comme le module cryptographique et mathématique.

Passons les composants en revue, en commençant par les modules conceptuels :

**Message** composant qui fournit un type abstrait permettant de manipuler les messages ISAKMP, indexés en interne par type de payload. Les services exportés sont création/destruction, l'ajout de payload, le parsing, la validation et la recherche de contexte pour les messages entrants, l'enregistrement de fonctions de traitement post-envoi, les fonctions d'envoi de message indépendantes du transport utilisé et des fonctions de débogage.

**Exchange** composant qui implémente le moteur de gestion des échanges ISAKMP, échanges dont le but est d'établir de nouvelles SA. Ce module est donc assez naturellement implémenté sous la forme d'un automate fini générique, piloté par des scripts qui décrivent les différents échanges définis par ISAKMP. Ces scripts décrivent la séquence des messages composant un échange, et permettent de vérifier que tous les payloads nécessaires sont présents. Ce module exporte les services d'établissement d'échange en tant qu'initiateur, et de prise en charge d'échange "entrant", ainsi que de fonctions de recherche suivant plusieurs critères.

**SA** gère une SADB interne au démon. cette SADB comprend les SA qui seront envoyées à la SADB maintenue par IPsec dans le noyau, mais aussi les SAs relatives à la phase 1. Les entrées sont crées dés le début d'une négociation et enrichies au fur à mesure, et ne deviendront actives qu'une fois l'échange finalisé. Les données de la SA et celles propres au domaine d'interprétation (DOI) sont séparées, permettant une future évolution vers la gestion de SAs pour une autre application qu'IPsec. Les services exportés sont ceux relatifs à la gestion des SA (créations, destructions, recherche,...)

**Application** prend en charge la communication avec l'application pour laquelle isakmpd travaille, dans notre cas IPsec. Dans le cas d'OpenBSD ce module communique avec la couche IPsec via l'API *pf_key*, afin de créer, mettre à jour, grouper et enlever les SA dans la SADB du noyau.

**Timer** gère les évènements temporels tels que l'expiration des SA. Il s'agit en fait d'un système de timers permettant d'enregistrer des fonctions de callback pour le traitement devant être fait après un laps de temps donné. C'est donc ce module qui appellera la fonction de destruction d'une SA à son expiration.

[3]le démon implémente le protocole ISAKMP qui est un cadre général. Jusqu'à ce jour IPsec est la seule application à utiliser ISAKMP pour le protocole d'échange de clef IKE, mais cela pourrait changer un jour.

**Policy** prend en charge les requêtes à la politique de sécurité, autorisant ou non la création de nouvelles SA. Il n'y a qu'une seule méthode exportée permettant de faire les requêtes avec, pour paramètres, une proposition de SA, l'identité de l'hôte distant et les sélecteurs du paquet. C'est ce module qui servira d'interface avec le système de gestion de confiance Keynote (cfr section 2.3)

**Network** est une abstraction d'un transport, implémenté sous la forme d'une table de pointeurs vers les fonctions d'un transport particulier. Il exporte les fonctionalités permettant de "créer" ou de "détruire" un transport, ainsi que la gestion des descripteurs de fichier nécessaires pour les entrées/sorties. Il fournit aussi les méthodes d'envoi et de réception de message ISAKMP.

**Configuration** est une base de données reprenant les paramètres de configuration lus au démarrage du programme ou à la réception du signal HUP par le processus isakmpd. Chaque élément de la configuration est représenté sous la forme d'un triplet (section, tag, valeur), cfr. description du fichier de configuration, section 2.2.2. Ce module exporte des fonctionnalités typiques à une base de données (insertion, effacement, requête) et peut être accédé de manière interactive via le module User Interface, permettant des changements dynamiques de la configuration.

**User Interface** interpréteur de commande permettant le changement dynamique des données de configuration, le changement du niveau de débogage et la gestion des SA par l'administrateur.

**Authentification** supporte les deux méthodes d'authentification actuellement disponibles : par clef partagée ou par certificat X.509 (RSA). Trois fonctions sont exportées : la génération d'un secret partagé pour la dérivation des clefs (Diffie-Hellman), l'encodage d'un hash[4]avec clef pour l'authentification, et le décodage d'un hash avec clef pour la vérification de l'authenticité.

**Crypto & Math** regroupe toutes les fonctions relatives aux calculs cryptographiques utilisés par isakmpd.

**Dynamic Loader** permet de charger les librairies implémentant les algorithmes RSA installées sur le système, cela pour contourné le problème lié au brevet couvrant l'algorithme RSA aux USA. En effet, à cause de ce brevet il était interdit jusqu'il y a peu d'importer une implémentation non patentée aux Etats-Unis. Il fallait donc pouvoir supporter des implémentations différentes suivant qu'on se trouve sur le territoire américain ou non.

**Logging** permet, comme son nom l'indique, de logger les informations en provenance des différents modules, en fonction de leur niveau de débogage. Ce module permet aussi de gérer le changement dynamique du niveau de débogage des classes de logging.

**System-dependant** regroupe les fonctions dépendantes du système, à des fins de portabilité. Ce module pourra donc être changé en fonction du système sur lequel le démon isakmpd doit tourner.

### 2.2.2 Paramètres de configuration : le fichier isakmpd.conf

Le comportement de ce démon est très configurable et permet de spécifier de nombreux aspects du comportement lors des négociations. Tous les paramètres du démon isakmpd sont

[4]certains livres en français utilisent le mot "condensat" qui correspond au mot anglais "digest" ou "hash". Nous avons préféré utiliser le vocable anglais bien plus familier dans la littérature technique.

regroupés de manière structurée dans le fichier isakmpd.conf. Les paramètres relatifs à un même concept sont regroupés par section. Certains paramètres prennent une valeur finale, tandis que d'autres référencent une section. Nous obtenons ainsi une structure de fichier hiérarchique.

Chaque section commencera par un nom entre crochet, c'est ce nom qui sera éventuellement utilisé comme référence à cette section. Il y a un certain nombre de sections imposées et dont la présence est obligatoire, tandis que l'administrateur du VPN pourra en créer de nouvelles suivant les besoins. Ces nouvelles sections permettront de regrouper les paramètres définissant les règles de négociations, ainsi que les paramètres des différentes propositions acceptées pour les connexions. La description complète de ces paramètres serait trop longue pour être inclue ici. Nous joignons toutefois en annexe une copie de la page de manuel où chaque paramètre est décrit en détail.

En plus des paramètres définis dans le fichier isakmpd.conf, il existe une série de sections pré-définies spécifiant différentes "suites de transformation"[5]. La liste est très longue et nous ne la reprendrons pas ici. Le lecteur pourra se référer à la page de manuel d'isakmpd disponible en ligne sur [20] pour plus de détails sur ces sections pré-définies.

## 2.3 Gestion de la police de sécurité : Keynote

Keynote est un système de gestion de confiance[6] d'utilisation générale. Voici la définition d'un système de gestion de confiance donnée dans le RFC 2704 :

> "... une approche unifiée pour spécifier et interpréter des politiques de sécurité, des accréditations et les relations entre elles ; il permet l'autorisation directe d'actions critiques d'un point de vue sécurité. Un système de gestion de confiance constitue un mécanisme standard pour spécifier les politiques de sécurité d'une application et les accréditations".

Keynote définit :

- un langage permettant de spécifier les 'actions', c'est-à-dire les opérations potentiellement dangereuses d'un point de vue sécurité et que l'on veut contrôler
- un mécanisme pour identifier les *"principals"*, c'est-à-dire les entités autorisées à accomplir des actions
- un langage pour spécifier les règles de la politique de l'application qui expriment les conditions dans lesquelles un *principal* peut accomplir une action.
- un langage pour définir les 'accréditations' qui permettent aux *principals* de déléguer leur droit à d'autre principaux.
- Un moteur de vérification de conformité, qui détermine pour l'application cliente si une action est autorisée pour un *principal*, en fonction de la politique de sécurité et d'une série d'accréditations.

Dans le cadre d'IPsec, keynote est utilisé pour déterminer si oui ou non une SA peut être créée, cela en fonction des règles définies, de l'identité de l'hôte avec qui la SA est négociée, et des paramètres de la SA à créer (qui résultent de la négociation).

Comment se définissent ces règles et les différents éléments de la configuration de keynote ?

[5] traduction de l'anglais "transform suites", ensemble de paramètres relatifs à l'utilisation d'un algorithme de chiffrement ou d'authentification, ou de la combinaison des deux.

[6] Trust Management System

Tout d'abord, keynote fournit, comme sommet de la hiérarchie des *principals,* le *principal* "POLICY". C'est à partir de ce *"root principal"* qu'on délègue le droit d'effectuer des actions. Un *principal* est identifié soit par un mot de passe, chiffré ou non, soit par une clef ou un certificat. Dans le cas de l'utilisation d'un certificat (X.509) on pourra mettre soit toute la clef, soit le Distinguished Name du détenteur du certificat.

Les droits d'exécution sont soumis à la vérification de certaines conditions pour pouvoir être autorisés, ce sont ces conditions qui constituent les règles d'autorisation et donc la politique de sécurité.

Les conditions sont formulées sous la forme de prédicats sur la valeur d'attributs qui sont propres au champ de l'application qui fait appel aux services de keynote. Les conditions pour une assertion sont reliées par des opérateurs logiques ET et OU pour déterminer la valeur de l'assertion pour les valeurs d'attribut données comme contexte.

L'application adressera donc des requêtes au moteur de vérification de keynote pour savoir si, par exemple, "l'utilisateur A a-t-il le droit d'effectuer l'action *act* dans le contexte (a1=x, a2=y, a3=z) ?" Dans ce cas, si il existe pour l'action spécifiée une assertion telle que A est un utilisateur accrédité, et que les valeurs des attributs concordent avec celles acceptées dans la règle, alors keynote renverra la réponse qui correspond à cette règle.

Voyons la structure d'une assertion définie dans le langage keynote :

KeyNote-Version : 2
Authorizer : <public key or tag>
Licensees : <public key or tag expression>
Comment : <comment text>
Conditions : <logic predicates>

Le champ Authorizer spécifie le *principal* qui délègue ses droits au propriétaire de la clef publique ou du certificat spécifié dans le champ Licensees. Le prédicat logique spécifié dans le champ Conditions devra toutefois avoir été vérifié, sur base des informations fournies dans le contexte de la requête.

Une accréditation est une assertion keynote signée par une autorité de confiance. Dans le cas d'une PKI[7] basée sur les certificats X.509 c'est donc le CA (l'Autorité de Certification, ou une entité ayant délégation de pouvoir du CA) qui signera les accréditations, et c'est le root certificate (certificat principal) qui servira à en vérifier l'authenticité. La signature numérique garantissant aussi l'intégrité du contenu signé, on peut donc distribuer les règles de sécurité.

### 2.3.1 Utilisation de Keynote par isakmpd

L'intégration de keynote dans le démon isakmpd est simple, car les appels aux services du composant gérant la politique de sécurité se trouvent à deux endroits seulement. Il a donc suffi de remplacer le code de la fonction appelée pour faire les requêtes au moteur de vérification de keynote.

Dans le cas de l'utilisation de certificats X.509, il est possible de déléguer l'autorisation en utilisant une autorité de certification (CA). Le CA signera les certificats, certifiant ainsi leur authenticité. Dans la spécification de la politique de sécurité mise en oeuvre via keynote, on spécifie une règle où le *principal* POLICY délègue tout ou partie de ses droits au CA (en

[7]Public Key Infrastructure, infrastructure de clef publique, dont le but est de gérer et distribuer un ensemble de certificats sous le contrôle d'une autorité de certification.

mettant le *root certificate* comme Licensee). La clef public du CA contenue dans son certificat sera utilisée pour valider que les certificats issus du CA sont dignes de confiance.

Mais à ce stade, si on a délégué les droits à l'authorité de certification (CA), aucune règle n'existe qui donne le moindre droit aux détenteurs d'un certificat signé par cette CA. Le démon isakmpd se charge de combler cette lacune : les certificats contenus dans le répertoire [X509_Directory] mentionné dans le fichier isakmpd.conf seront utilisés pour créer automatiquement des accréditations, après avoir vérifier qu'ils sont bien dignes de confiance. La création d'accréditation sera aussi effectuée automatiquement pour les certificats reçu durant une négociation de SA grâce à l'envoi d'un payload de type Certificat.

Cela est très bien, mais imaginons la situation ou il y a des centaines d'utilisateurs, ayant chacun un certificat issu d'un même CA (directement ou par entités déléguées), et pouvant accéder à plusieurs passerelles IPsec selon des politiques différentes. Dans ce cas l'administration devient problématique si il faut spécifier des règles de politiques de sécurité pour chaque certificat sur les différentes passerelles... Tout cela peut être résolu facilement par l'utilisation des 'credentials' (accréditations). Une accréditation est une assertion keynote signée par autorité de certification. Cette signature garanti son intégrité et son authenticité, il n'est donc plus nécessaire de stocker toutes les règles au point de contrôle (la passerelle), et donc chaque machine/utilisateur ayant les droits de se connecter peut conserver ses accréditations, et les présenter au moment d'établir la connexion (négociation des SA). La passerelle vérifie alors authenticité des accréditations présentées et les intègre aux règles courantes de politique de sécurité.

Cette méthode permet d'alléger considérablement la charge d'administration, chaque hôte ayant accès aux services IPsec de la passerelle conservant lui-même ses accréditations.

La passerelle pourra obtenir les accréditations soit lors de l'échange qui a lieu pendant la négociation des SA, soit au près d'un serveur via un mécanisme externe au protocole IKE. Le mécanisme d'échange entre démons isakmpd existe, mais ce n'est hélas pas un standard très répandu et ne fonctionne qu'entre démons isakmpd (pas d'interopérabilité).

## 2.4 Composants annexes et outils

Il existe aussi certains composants qui ne font pas directement partie d'IPsec, mais qui sont utilisés dans le cadre de cette implémentation. Nous allons passer les principaux en revue.

### 2.4.1 OpenSSL

OpenSSL est une collection de librairies et d'outils qui permettent de gérer les connexions SSL et tout ce qui s'y rattache. OpenSSL fournit aussi les librairies de chiffrement utilisées par IPsec, aussi bien pour les opérations à clef publique que pour le chiffrement symétrique. C'est donc un composant clef de l'implémentation d'IPsec.

Parmi ses fonctionnalités, une autre qui nous intéresse est sa capacité à gérer les certificats X.509. C'est l'utilitaire en ligne de commande openssl qui nous permettra de créer la clef et le certificat de l'autorité de certification (CA root certificate), et ensuite de signer les certificats de la passerelle et des clients. Nous verrons le détail des commandes utilisées et de leur syntaxe dans l'étude de cas qui suit cette section. La documentation complète sur OpenSSL

### 2.4.2 ipsecadm

L'outil ipsecadm permet d'administrer manuellement la SADB, en créant et en détruisant les SA. Cela permet d'établir des connexions IPsec manuellement, sans devoir utiliser le démon isakmpd. Il faut toutefois savoir que certaines fonctionnalités comme la protection anti-rejeu et le renouvellement périodique des clefs (des SA) ne sont disponibles qu'avec l'utilisation d'un démon comme isakmpd pour la gestion automatique des SA.

Des exemples d'établissement de connexion IPsec manuelle sont donnés dans les page du manuel ipsecadm(8) et vpn(8). Ces pages de manuel sont disponibles en ligne sur [20].

### 2.4.3 Internet Paquet Filter - Firewall

OpenBSD inclus un système de filtrage de paquet très complet et offrant de nombreuses fonctionnalités.

Parmi celles-ci, les plus importantes pour nous sont le filtrage de paquet qui permet de restreindre le trafic TCP/IP sur base des adresses source et destination ainsi que des ports, en spécifiant des règles qui seront appliquées sur une interface donnée et dans un sens donné (*in* ou *out*). Ces règles peuvent être définies comme *stateful*, c'est à dire que pour une telle règle authorisant l'ouverture d'une connexion dans un sens, le trafic de retour sera automatiquement autorisé pour les connexions ouvertes grâce à la mémorisation de l'état des connexions.

Le filtrage sera utilisé pour restreindre le trafic aux seuls services utilisés par une passerelle IPsec, et interdire tout autre trafic.

Une autre fonctionnalité que nous utiliserons est la translation d'adresses, ou NAT. Celle-ci permet de convertir l'adresse de destination d'un paquet arrivant dans un sens afin qu'il soit envoyé vers une machine portant une adresse privée (par exemple), et d'effectuer la conversion inverse pour les paquets de la même connexion repartant dans l'autre sens. De cete façon les machines de l'extérieur communiquent avec des machines sur un réseau local derrière le firewall sans s'en rendre compte, et sans voir les adresses privées utilisées par celles-ci. Ces règles de conversion peuvent aussi convertir les ports. PLus d'explications seront données à ce sujet dan le chapitre suivant. La documentation détaillée est également disponible dans la page de manuel de l'application ipf sur le site d'OpenBSD [20], mais uniquement jusqu'à la version 2.9 du système.

# Chapitre 3

# Etude de cas : Société 3S

Le présent chapitre constitue l'élément principal de ce document. Les chapitres précédents visaient à introduire les concepts utilisés dans le cadre de la mise en oeuvre d'une solution concrète que nous allons maintenant décrire. Nous relaterons les étapes d'analyse, de configuration et de test qui ont mené à la solution actuellement en phase finale de test dans les installations de réseau de la compagnie Streamlined Solutions & Services.

Dans un premier temps nous décrirons le problème que le projet vise à résoudre, ainsi que les critères que la solution devra respecter, particulièrement en ce qui concerne le niveau de sécurité requis.

Nous passerons ensuite en revue l'état de la configuration du réseau au commencement du projet, avant d'étudier les différentes possibilités de d'implantation d'une passerelle d'accès sécurisé.

Nous verrons ensuite les différentes étapes de la configuration des éléments entrant dans la composition d'une solution complète, centrée autour de la passerelle d'accès IPsec.

La présentation des différents tests effectués afin de valider la solution constituera l'avant-dernière partie de ce chapitre. Nous verrons quels ont été les problèmes rencontrés au cours de ces tests, et comment nous avons choisi de les résoudre.

Nous cloturerons ce chapitre en évoquant l'avenir du projet, les prochaines étapes menant à son utilisation réelle par l'entreprise, ainsi que la possibilité d'utilisation de la même technologie pour sécuriser la liaison avec le siège de 3S au Sri-Lanka.

## 3.1 Présentation du problème

Streamlined Solutions & Services (en abrégé 3S) est une société de services informatiques basée à Luxembourg. Elle fournit principalement des solutions de clearing[1] dans les domaines de la finance et des télécoms.

La compagnie compte maintenant une cinquantaine d'employés. L'infrastructure informatique compte maintenant plus de 60 machines, parmi lesquelles on trouve les stations de travail, des serveurs de fichiers, des serveurs de test, ainsi que les serveurs web intranet, mail, base de données, etc. La quasi-totalité de ces machines tournent sous Windows. Un serveur de fichier permet aux employés de stocker et d'accéder à leurs documents et fichiers de travail, tout en assurant une sauvegarde régulière. Toutes ces ressources sont bien sûr accessibles à partir du

[1] Une explication de ce qu'est le clearing est disponible dans le glossaire, au début de ce document.

réseau local. Jusqu'ici la seule possibilité d'accès à distance était limitée à l'utilisation d'une seule connexion ISDN avec callback. Le callback consiste à recevoir l'appel de la machine voulant se connecter, mais cet appel n'est en fait qu'une demande de connexion ; le serveur coupe cette connexion et rappelle la machine distante sur le numéro qu'elle a utilisé pour faire sa connexion de demande. De cette façon le numéro de l'appelant est identifié et les coûts téléphoniques sont comptabilisés pour le serveur et facturés à l'entreprise.

La possibilité de se connecter à distance pour accéder aux ressources du réseau local est intéressante dans plusieurs scénarios :

- Accès à distance pour des raisons administratives, si par exemple l'administrateur doit procéder à des vérifications ou des interventions le soir de chez lui ou exceptionnellement pendant ses congés. Dans ce cas une connexion ISDN pourrait suffir.
- Accès à distance par les cadres pour accéder à certains documents, soit de chez eux, soit de leur ordinateur portable pendant un déplacement. La présence d'une ligne ISDN n'est pas garantie partout, et dans le cas où plusieurs accès sont nécessaires simultanément cela nécessite plusieurs lignes.
- Le cas du télé-travail peut s'avérer utile dans certaines situations particulières, même si c'est assez exceptionnel chez 3S. Dans ce cas une connexion ouverte en ISDN peut s'avérer coûteuse, surtout dans le cas d'une connexion internationale entre la Belgique et le Luxembourg par exemple.

Pour rappel, une connexion ISDN primaire offre un débit de donnée garanti de 64 ou 128 kilobits par seconde (kbps) suivant qu'on utilise un seul ou les deux canaux, dans les deux sens de communication, mais nécessite l'installation d'équippement spécial par l'opérateur du réseau téléphonique, et coûte plus cher à l'abonnement. Le tarif d'utilisation est similaire à celui du téléphone normal, on paye par unité de temps de communication. Donc l'ISDN est cher pour des connexions de longue durée. De plus comme quasiment toutes les connexions de ce type se font à partir de la Belgique, le coût est encore augmenté par le fait que la communication est internationale.

Internet offre plusieurs avantages par rapport à cela :

- multiples possibilités de connexions à l'internet, il peut s'agir d'une simple communication téléphonique locale grâce à un modem, d'une connexion ADSL, ou d'un accès directe via un réseau connecté à l'internet en permanence. Tous ces moyens d'accès n'offrent pas le même débit de donnée, mais il sont pratiquement disponibles partout. Ce n'est pas toujours le cas de l'ISDN puisqu'il nécessite un équipement spécial, et qui de plus en plus est délaissé au profit de l'ADSL pour l'Internet.
- dans un cas de connexion au réseau local de 3S pour de longues périodes comme dans le cas de télé-travail occasionel, une connexion ADSL à l'internet est sans conteste le moyen de connexion le moins cher et assurant le maximum de confort. En effet l'ADSL offre un débit égal ou supérieur à celui d'une connexion ISDN, est facturé en fonction de la quantité de données échangées avec un forfait assez élevé (de l'ordre de 15 GigaBytes par mois pour une connexion individuelle courante), et donc permet de travailler des journées entières, plusieurs jours d'affilée pour un coût fixe très bas. De plus en plus d'employés sont déjà équipés d'une telle connexion qui, à part l'achat d'un modem spécifique, coûte le même prix qu'un abonnement utilisant la ligne téléphonique standard.
- L'accès à L'internet se fait de manière locale, où que l'on se trouve dans le monde.

Par contre l'accès via l'Internet pose le problème de la sécurité et de la confidentialité des données. Il est en effet impératif que :

- seul les employés ayant accès à distance au réseau local puissent se connecter.
- les données et fichiers échangés ou accédés au cours d'une connexion distante ne doivent pas pouvoir être interceptés ou modifiés par des tiers.

L'utilisation d'Internet comme moyen d'accès nécessite donc l'emploi de méthode d'authentification et de protection de la confidentialité des données. Un autre point du cahier des charges est de permettre un accès aussi transparent que possible aux ressources du réseau. Les utilisateurs distant devront pouvoir accéder à leur fichiers, leur compte email, et devront également pouvoir utiliser des outils de contrôle à distance comme PC-Anywhere ou VNC.

Une technologie de type VPN (Virtual Private Network) semble d'emblée une bonne solution. Elle permettrait aux utilisateurs distant de se connecter au réseau local et d'en utiliser les ressources exactement comme si leur machine y était directement connectée.

Plusieurs technologies de type VPN existent, parmis celles-ci toutes ne répondent pas aux différents points de notre cahier des charges, particulièrement en ce qui concerne la transparence d'utilisation et la disponibilité et la facilité d'utilisation des solutions.

IPsec s'est présenté comme une solution possible permettant tout cela :

- transparence car le protocole se situe au niveau de la couche réseau (IP). Tout trafic réseau utilisant TCP/IP peut donc être véhiculé de manière sécurisée grâce à IPsec
- possibilités d'authentification forte basée sur l'utilisation de certificats
- protection de la confidentialité et de l'intégrité des donnés
- protocole supporté par l'industrie, et donc une grande disponibilité de solutions matérielles et logicielles.

### 3.1.1 Aperçu de la configuration du réseau de 3S

La configuration du réseau 3S peut être qualifiée de classique, et est représentée dans le schéma de la figure 3.1. Les deux sous-réseaux de la partie supérieure du schéma forment le réseau local sur lequel se trouvent les serveurs d'une part, et les stations de travail et environnements de test d'autre part. Les adresses utilisées sur ces sous-réseaux sont des adresses IP qui ne peuvent pas être routées sur l'Internet, et font partie d'une catégorie d'adresses spécialement dédiées à un usage privé par l'IANA[2]. Ces deux sous-réseaux sont reliés par un routeur, qui est également connecté au firewall.

Le firewall est une machine dédiée sur laquelle tourne le logiciel Firewall-1 produit par la société Checkpoint, spécialisée dans les solutions de sécurité réseau. Le but de ce firewall est de séparer le réseau local des réseaux extérieurs, tout en permettant certains accès bien précis. Il offre aussi des possibilités de conversion d'adresse, permettant ainsi qu'une machine interne ayant une adresse privée soit accessible de l'extérieur via une adresse publique.

Un autre sous-réseau spécial est connecté au firewall. Sur ce sous-réseau se trouvent les serveurs communiquant avec le monde extérieur (Internet). Certains de ces serveurs offrent un service au monde extérieur, comme par exemple le serveur web qui est accessible pour tous et donne des informations sur la compagnie. D'autres serveurs utilisent des ressources disponibles sur l'Internet, par exemple le serveur Proxy récupère le contenu de sites web pour le compte des machines du réseau local interne. Il fait office d'intermédiaire sécurisé en vérifiant le contenu des pages web pour y détecter d'éventuels virus. Un proxy similaire effectue cette vérification

[2]Internet Assigned Number Authority, l'autorité chargée de centraliser les numéros de protocoles, d'algorithmes, ainsi que les adresses IP réservées à un usage spécial. pour plus d'information, consultez le site le l'IANA : http ://www.iana.org

pour le compte du serveur de courrier électronique qui se trouve sur le réseau interne. Ce sous-réseau spécial est appelé *Zone Démilitarisée* (en anglais *Demilitarized Zone,* ou *DMZ*), il joue le rôle de zone de sécurité entre l'Internet et le monde extérieur, un peu comme la zone située entre les deux murailles d'une fortification. Tout trafic venant de ou allant vers ce sous-réseau passe par le firewall, qui n'autorise que certains types de protocoles bien précis, de ou vers certains des hôtes spécifiques. Par exemple le trafic smtp (envoi de courrier électronique) est autorisé uniquement entre le serveur de mail interne et le serveur proxy mail dans la DMZ d'une part, et entre le serveur proxy et l'Internet d'autre part.

Le lien entre le firewall et l'Internet se fait à travers un switch Ethernet qui relie les différents réseaux externes. Un routeur d'accès aux réseaux externes assure la connexion vers les réseaux privés de certains clients, et joue lui aussi un rôle de firewall en bloquant le trafic entre ces réseaux externes, n'autorisant que les communications de et vers le firewall de 3S. Un autre routeur est dédié à la connexion vers l'Internet, et n'autorise que les connexions de et vers les adresses publiques qui correspondent aux serveurs de la DMZ communiquant avec l'extérieur (et que le firewall converti vers les adresses privées de ces serveurs).

Les accès à distance se font par une connexion sur le routeur d'accès aux réseaux externes qui possède une interface ISDN. Dans ce cas la configuration du firewall doit être modifiée pour autoriser le trafic entre la machine distante et la ou les machines sur le réseau interne. L'autre cas d'accès à distance utilisé pour le télé-travail fut mis en oeuvre différemment, la connexion ISDN se faisant directement sur une machine du réseau interne équipée d'un modem. Afin de limiter les problèmes de sécurité liés à cette configuration (puisqu'un accès extérieur se fait alors sans passer par le firewall), l'appel de la machine distante ne servira qu'à démarrer la procédure de connexion, et sera coupé immédiatement par la machine sur le réseau local. Celle-ci rappellera ensuite automatiquement le numéro de l'employé utilisant l'accès à distance, ce numéro étant configuré manuellement sur la machine interne et ne dépendant pas du numéro d'appel utilisé pour la demande de connexion. Si donc une personne étrangère tentait une connexion sur cette ligne, elle ne provoquerait au pire qu'un appel automatique chez l'employé.

### 3.1.2 Projet d'implantation de la passerelle IPsec

Le projet d'installation d'une passerelle IPsec est le fruit d'une proposition personnelle. Suite à des recherches personnelles sur IPsec, j'en ai parlé à l'administrateur réseau de chez 3S qui m'a dit que cela pourrait être intéressant pour 3S. J'ai donc reçu le feu vert pour effectuer les tests et mettre au point une solution, à la condition de le faire en dehors de mes heures de travail normales puisque je suis moi-même développeur pour un des projets de la compagnie. Le fait que ce projet n'ai pas pour origine un besoin urgent de l'entreprise et que ce ne soit pas vraiment un projet officiel (même si il est fait avec l'accord de la direction) a bien sur un impact sur les solutions adoptées.

Le but du projet est d'offrir un accès aux ressources du LAN à travers l'Internet, de manière la plus transparente possible, tout en garantissant un haut niveau de sécurité. IPsec, étant situé au niveau de la couche réseau, offre une protection automatique et transparente, cela quelle que soit l'application TCP/IP utilisée pour accéder aux ressources du LAN.

Dans ce but, l'infrastructure mise en oeuvre consiste en :

- une passerelle IPsec comportant deux interfaces réseau, l'une connectée au réseau local, l'autre à l'Internet. L'accès à l'Internet se fera soit à travers le firewall déjà en place,

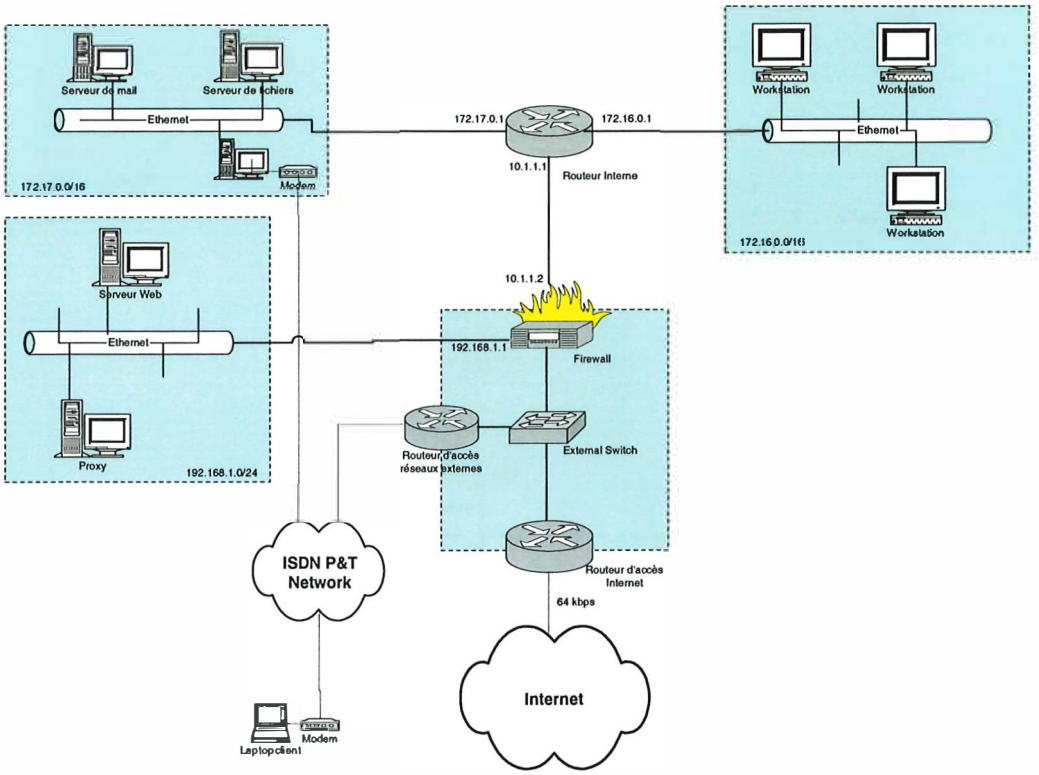



FIG. 3.1 – Schéma du réseau 3S

soit plus directement via le switch connecté au routeur servant de point d'accès.
- une machine de gestion des certificats. Celle-ci ne sera pas connectée au réseau, et servira uniquement à produire les certificats pour les différents acteurs (passerelle, clients,...). Nous en parlerons plus longuement dans la partie consacrée aux certificats.
- des machines clientes équipées d'un logiciel implémentant client VPN basé sur IPsec. Ces machines sont les machines personnelles des membres du personnel qui auront accès au réseau local via le VPN.

Une question importante est de savoir où la passerelle sera implantée. Il y a plusieurs possibilités :

- elle pourrait être installée dans la DMZ, derrière le firewall, avec les autres machines communicant avec l'Internet (serveur mail et web, proxies,...). L'accès Internet se fera via le firewall qui effectuera une translation d'adresse statique, afin de faire correspondre l'adresse IP utilisée par la passerelle dans la DMZ à une adresse publique accessible. Le firewall appliquera aussi un filtrage n'autorisant que les services nécessaires aux connexions IPsec et à l'administration. Cette implantation est illustrée par la figure 3.2.
- l'autre possibilité serait de la connecter directement à l'Internet, en parrallèle avec le firewall. La passerelle devra alors assurer toutes les mesures de sécurité vis-à-vis du monde extérieur, et il n'y aura pas de translation d'adresse du côté extérieur, l'interface externe utilisera une adresse publique. Cette implantation est illustrée par la figure 3.3.

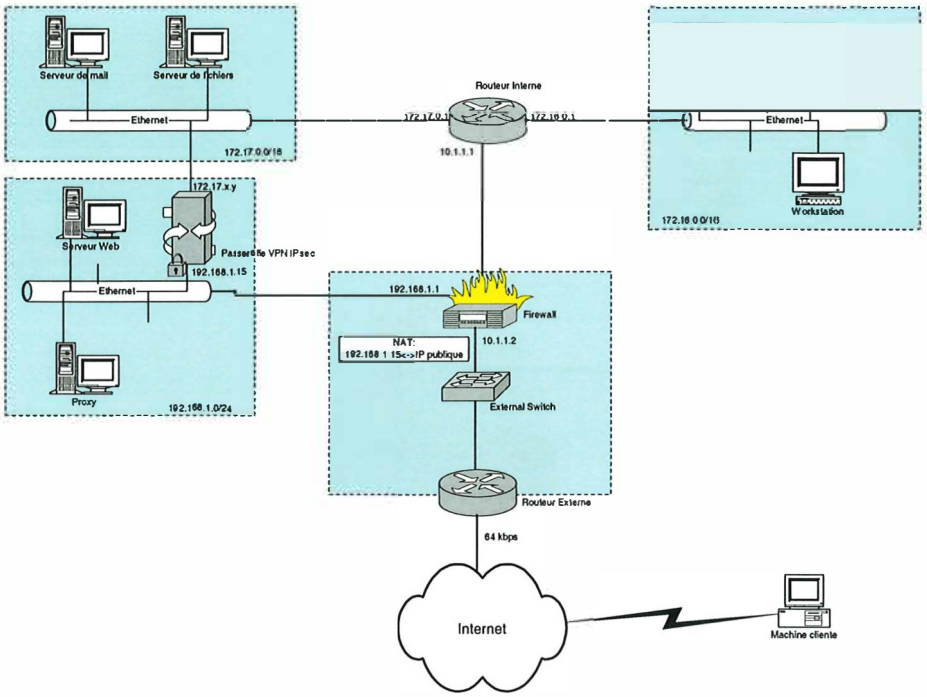



FIG. 3.2 – Implantation de la passerelle IPsec dans la DMZ

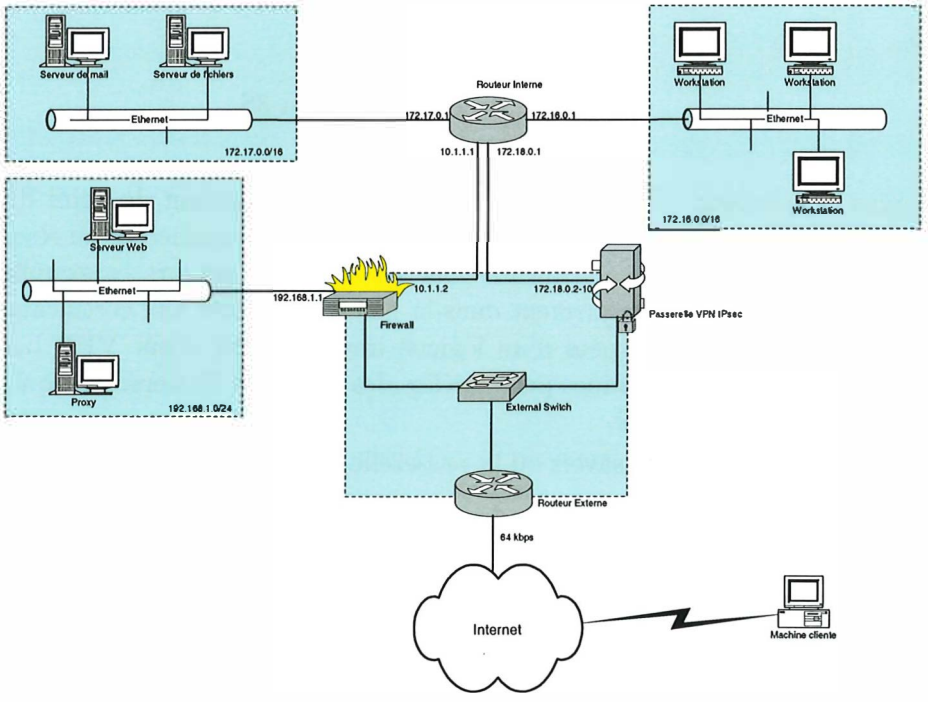



FIG. 3.3 – Implantation de la passerelle IPsec en parallèle avec le firewall

La connexion entre la passerelle et le réseau local pourra se faire :
- soit sur le routeur interne, en utilisant une ou plusieurs adresses privées sur un sous-réseau différent pour la passerelle, comme illustré dans la figure 3.3.
- soit directement sur l'un des deux sous-réseaux internes, celui hébergeant les serveurs semblent plus approprié dans la mesure où on peut considérer la passerelle comme un serveur. Cette connexion est illustrée dans la figure 3.2. Dans ce cas la passerelle utilisera une ou plusieurs adresses privées appartenant au sous-réseau auquel elle est connectée.

Dans les deux cas toutes les machines du réseau local, que ce soit les stations de travail ou les serveurs, seront accessibles, avec même la possibilité de ménager un accès vers les machines de la DMZ en passant par le firewall (pour faire de l'administration à distance). Toutes les connexions venant de l'extérieur à travers la passerelle IPsec auront leur adresse source convertie en une adresse locale cohérente avec celles utilisées sur le LAN, et qui pourra être routée sans problème de tous le points du réseau. D'éventuelles modification de la configuration du routeur seront peut-être nécessaires. Nous reviendrons sur la configuration de la conversion d'adresse plus loin.

Le choix du placement de la passerelle dans la DMZ semblait plus approprié, de cette façon la passerelle serait protégée par le firewall. Les problèmes rencontrés lors des tests de cette configuration nous ont finalement fait adopté l'implantation en parallèle. Plus de détails seront donnés dans la section concernant les tests.

## 3.2 Choix d'implémentation

Il existe de nombreuses implémentations d'IPsec, libres ou commerciales, intégrées à un système d'exploitation ou en tant que module matériel externe, ou encore en tant que module logiciel qui vient s'ajouter à un firewall par exemple. Ce dernier type de solution existe pour le firewall utilisé chez 3S (Checkpoint Firewall-1), mais le prix de la license est très élevé, et le fait que le projet soit non-prioritaire pour la companie (c'est-à-dire non-motivée par des raisons liées au business de l'entreprise) rend cette solution prohibitive.

C'est donc ce qui a orienté le choix d'expérimenter une solution libre. Il existe plusieurs projets visant à intégrer une implémentation IPsec dans différents systèmes d'exploitation de type Unix, citons le projet KAME (http ://www.kame.net) dont l'implémentation IPsec est intégrée à FreeBSD et NetBSD, L'implémentation inclue dans OpenBSD qui fut développée dans le cadre d'un projet financé par Ericson Radio Systems AB, et le projet FreeSwan (http ://www.freeswan.org) qui est un portage de l'implémentation d'OpenBSD vers les systèmes basés sur le noyau Linux. Tous ces projets étaient déja fonctionnels au moment du début de notre projet il y a environ un an et demi.

Le choix ne s'est pas basé sur le test de toutes ces implémentations (cela aurait pris trop de temps), d'autres critères sont également entrés en ligne de compte. La réputation du système OpenBSD a fortement influencé le choix. En effet OpenBSD est spécifiquement développé pour les applications de réseau, et a la sécurité pour objectif principal. OpenBSD n'avait jamais eu de trou de sécurité exploitable à distance dans l'installation par défaut, jusqu'à il y a peu ou un problème lié à OpenSSH pouvait être exploité sous certaines conditions. Il faut toutefois noter que le patch solutionnant ce problème fut publié avant la mise en ligne du code démontrant la faille.

Un autre avantage du système OpenBSD par rapport à beaucoup de distributions basées sur le noyau Linux est la sobriété de son installation de base. En effet dans les quelques expé-

riences personnelles d'installation de distributions Linux, l'installation par défaut comprend beaucoup trop de logiciels inutilisés, il faut faire le tri parmis des milliers de packages, avec finalement le risque de perdre le controle de ce qui est installé ou non. Cela comporte des risques au niveau de la sécurité, ou du moins demande plus de travail pour renforcer la sécurité du système (hardening). L'installation d'OpenBSD est minimale, seul ce qui est absolument nécessaire pour le fonctionnement du système est installé. Apartir de là on peut rajouter autant de logiciels que nécessaire, mais le contrôle est meilleur.

Autre point positif : le système OpenBSD inclus un très bon package de firewalling appelé ipf (pour Internet Packet Filter). Celui-ci offre des services de filtrage avec mémorisation des connexions (stateful firewalling), ainsi que de translation d'adresse (NAT) et de redirection. Les distributions d'OpenBSD à partir de la version 3.0 on abandonné ipf pour une autre suite, appelée pf (Packet Filter), et cela semble-t-il uniquement pour des raisons de divergeance d'opinion relative au licensing entre le développeur d'ipf (Daren Reed) et les gens de l'équipe OpenBSD. La version 2.9 que nous avons utilisée inclue toujours ipf et nous l'avons gardée car elle a fait ses preuves, et il semble que pf n'offre aucune fonctionnalité supplémentaire.

Tous ces points nous ont fait choisir OpenBSD comme plateforme pour la passerelle IPsec.

## 3.3 Mise en oeuvre

Nous allons maintenant parcourir les différentes étapes de la mise en oeuvre de l'infrastructure décrite dans les sections précédentes. La plus grande partie du texte portera sur la configuration de la passerelle. Nous exposerons les particularités de l'installation du système d'exploitation, les mesures de *hardening*[3] (configuration du noyau et des services, etc), la configuration du firewall et des règles de NAT, et, biensur, la configuration des composants IPsec, principalement celle d'isakmpd et du fichier de politique de sécurité keynote.

Nous verrons ensuite la gestion des certificats, ce qui inclue :

- création du certificat du CA (Certification Authority).
- création du certificat pour la passerelle.
- création des certificats pour les clients, avec procédure de demande.

Nous parlerons également du produit choisi pour la connexion du côté client, sa configuration, et le mode de déploiement, ce y compris la demande de certificat, jusqu'à ce que le client soit prêt à se connecter à travers la passerelle.

Nous verrons les mesures de surveillance prises sur la passerelle, afin de détecter les tentatives d'attaque en tout genre.

Nous terminerons en exposant les tests que nous avons effectué, les problèmess rencontrés au cours de ceux-ci, et les solutions que nous avons adoptées.

### 3.3.1 Configuration matérielle de la passerelle

Voici quelques information sur la configuration matérielle de la passerelle :

**CPU** Intel Pentium III 800Mhz

**Mémoire** 128MB SDRAM

**Disque dur** Interface IDE UDMA, 12GB

[3] ensemble de mesures dont le but est de réduire au maximum les chances de vulnérabilité aux attaques réseau.

**Interface réseau interne** modèle "DEC DECchip 21142/3" 10-100Mb/s

**Interface réseau externe** modèle "Intel 82557" 10-100Mb/s (Intégrée à la carte mère)

Comme nous pouvons le voir c'est une configuration de base utilisée en général pour une machine de bureau. Elle fera parfaitement l'affaire pour notre installation car nous n'avons pas de besoins spéciaux en puissance de calcul, surtout compte tenu de la bande passante de la ligne louée nous reliant à l'Internet (64Kb/s) : la passerelle n'aura jamais à traiter plus de 64Kb par seconde d'information, ce qui est très peu. D'après des tests effectués sur réseau local, elle serait en fait capable de soutenir une charge beaucoup plus élevée sans problème.

### 3.3.2 Installation du système, configuration

La distribution OpenBSD peut s'obtenir soit par l'achat des CD sur le site (www.openbsd.org), soit par téléchargement sur le site ftp. Il est également possible de télécharger une image de disquette bootable qui permettra de démarrer une installation via l'Internet, les différents packages étant récupérés sur le site ftp (ftp.openbsd.org). Nous avons opté pour la seconde solution, le débit d'une ligne ADSL permettant de récupérer le contenu des 2 CDroms en une nuit.

Tout au long des explications qui vont suivres, nous utiliserons des termes et des commandes auxquels les utilisateurs avancés et les administrateurs de système Unix sont familiers. Les lecteurs désireux de se familiariser à l'administration Unix peuvent se référer à [22].

L'installation est assez simple, les principales étapes étant le partitionnement du disque, le choix des packages à installer (principalement avec support X11 ou pas) et la création du compte root. Il faut également créer un compte utilisateur simple, qui sera utilisé pour les taches courantes, pour les tâches administratives il est préférable d'utiliser alors la commande *su*[4] à partir de ce compte plutôt que de se connecter directement avec le compte root. Un tel compte devra être ajouté au groupe *wheel* pour pouvoir faire un su root.

Pour notre partition nous avons découpé le disque comme suit :

/ la partition principale, de 50MB

**/usr** la partition contenant les applications utilisateur, et également l'arborescence des sources si des recompilations sont nécessaires. Taille : 1GB

**/home** la partition accueillant les répertoires attachés aux comptes des utilisateurs. Pour des raisons de confort nous lui avons attribué 1GB, bien qu'il n'y aura probablement pas plus de deux comptes sur cette machine (uniquement les comptes des administrateurs du système).

**/var** la partition contenant des variables, des informations temporaires,... mais aussi les fichiers de logging (journaux) du système. Nous avons attribué à cette partition tout le reste du disque (9GB), afin d'avoir beaucoup de place dans le cas où nous voudrions logger le trafic réseau dans son intégralité lors des tests.

Parmis les packages à installer, nous installons la base du système sans support de l'interface graphique X11, avec le compilateur car nous en aurons besoin pour recompiler le noyau.

Une fois l'installation de base terminée, il faut procéder à quelques modifications de la configuration pour endurcir le système (hardening). Celui-ci étant destiné à être connecté à

[4]La commande su permet à un utilisateur de prendre l'identité d'un autre utilisateur, après avoir fourni le mot de passe correspondant. Utilisée sans paramètre, l'utilisateur prend l'identité du super-utilisateur (root). l'ajout d'un "-" permet de charger le profil complet du compte cible.

l'internet, il faut être certain de verrouiller toute possibilité d'accès non-désiré, en désactivant tout ce qui n'est pas indispensable au système pour remplir sa fonction. Par exemple le démon *portmap* utilisé pour les requètes RPC (communication inter-process via le réseau) ainsi que le serveur de mail *sendmail* devront être désactivé dans le fichier de configuration /etc/rc.conf

Nous avons également procédé à une recompilation du noyau, afin de ne conserver que les pilotes de périfériques nécessaires pour le matériel présent sur notre machine, et pour supprimer le support de certains services non-nécessaires pour nos besoins, comme par exemple le support du protocole IPv6.

Afin d'activer la prise en charge des protocoles ESP et AH par le noyau, il faut d'une part que les options suivantes soient dé-commentées dans le fichier de configuration du noyau situé dans le répertoire /usr/src/sys/conf/ :

```
option GATEWAY
option INET
option IPSEC
option KEY
```

Dans l'ordre, ces options activent la prise en charge du *packet forwarding*[5], de la suite de protocoles TCP/IP, de la suite de protocoles IPsec et de l'interface Pf_key qui est utilisée par le démon isakmpd pour gérer les SA IPsec.

Il faut également que certaines propriétés soient activées, par la configuration des paramètres du noyau suivants, disponibles dans le fichier /etc/sysctl.conf :

```
net.inet.ip.forwarding=1
net.inet.esp.enable=1
net.inet.ah.enable=1
```

Ces paramètres peuvent aussi être configurés dynamiquement avec la commande sysctl, par exemple :

```
# sysctl -w net.inet.esp.enable=1
```

Le lecteur désireux d'en apprendre d'avantage sur la configuration du système OpenBSD pourra consulter le FAQ disponible sur le site d'OpenBSD [20].

D'autres mesure de durcissement pourraient être prises. Certains documents recommandent de retirer pûrement et simplement tout exécutable qui ne soit pas nécessaire à l'exécution des tâches de la passerelle. Ainsi, même si la passerelle se trouvait compromise, l'attaquant aurait très peu de ressources lui permettant d'exploiter la situation. L'idée est de déplacer ces exécutables non-nécessaires sur un support amovible (par exemple un cdrom), et de les remplacer par des liens symboliques vers le fichier réel sur ce support qui pourrait être monté en cas de besoin. Mais cette tâche est très lourde à mettre en oeuvre, car la liste des exécutables sans lesquels le système peut tourné prend du temps à établir, car il faut s'assurer de ne pas enlever trop de fichiers. Cete mesure n'a donc pas été prise pour l'instant.

Une autre mesure utile pour détecter une éventuelle intrusion consiste à calculer pour tous les fichiers exécutables une valeur de contrôle en utilisant l'algorithme de hashage md5. Toutes ces valeurs sont stockées sur un support accessible en lecture seule, et vérifiées régulièrement afin de certifier qu'aucun des fichiers protégés n'ait été modifié.

Le lecteur désireux d'en apprendre d'avantage sur le processus de hardening du système OpenBSD trouvera une série d'articles intéressants à l'adresse : http ://www.geodsoft.com

[5]la capacité qu'a une passerelle de faire passer les paquets IP d'une interface vers une autre est appelée en anglais *packet forwarding*.

### 3.3.3 Configuration réseau

La configuration réseau que nous allons décrire est celle en place actuellement, avec la passerelle implantée en parallèle du firewall.

#### Configuration de la passerelle

La passerelle comporte deux interfaces Ethernet de type standard et donc très bien supporté par le système.

L'interface utilisée pour la connexion du côté externe est configurée avec une adresse IP publique que nous ne dévoilerons pas dans ce document; cette adresse fait partie du sous-réseau d'adresses officielles attribuées à la société 3S. Elle est connectée à l'un des ports du switch externe, et supporte un débit maximum de 100Mb/s.

L'interface interne qui sera reliée au réseau local (directement ou via le routeur) sera configurée avec une adresse de type 172.17.5.x (dans le cas d'une connexion directe) ou 172.18.0.2 (si elle est connectée au routeur). Si nécessaire, cette interface pourrait être configurée avec plusieurs adresse IP (du même sous-réseau), afin de pouvoir satisfaire plus de connexions simultanées. Dans ce cas ces adresses doivent être configurées sur l'interface afin d'être utilisées ensuite par le mécanisme de translation d'adresse dont nous parlerons un peu plus loin.

La configuration d'une interface réseau se fait grâce à un fichier situé dans le répertoire /etc, et qui contient les adresses attribuées à une interface pour les différents protocoles supportés. Le fichier porte un nom indiquant à quelle interface il se rapporte, par exemple pour l'interface fictive if0 le nom du fichier sera /etc/hostname.if0.

Chaque ligne de ce fichier décrit la configuration d'une adresse, ou d'un alias. Un interface comporte une adresse IP principale et peut avoir des adresses IP secondaires appelées alias. La syntaxe d'une ligne est :

addr_family [alias] addr netmask broadcast_addr options

où :

**addr_family** est le type d'adresse. Dans le cas d'une adresse IPv4, ce paramètre vaut *inet*.

**alias** est un indicateur optionel pour signaler que la ligne courante définit un alias.

**addr** est l'adresse proprement dite, exprimée dans la notation décimale.

**netmask** est le masque de sous-réseau exprimé dans la notation décimale.

**broadcast_addr** est l'adresse de broadcast, c'est-à-dire l'adresse qui permet de transmettre des informations spéciales à tous les hôtes du sous-réseau. Ce paramètre est optionel.

**options** est une liste de paramètres optionels.

Le contenu du fichier de configuration de l'interface externe /etc/hostname.de0 est :

inet w.x.y.z 255.255.255.224 NONE

où w.x.y.z est l'adresse IP publique attribuée à la passerelle, et par laquelle les clients externes pourrons accèder au LAN via la passerelle.

Le contenu du fichier de configuration de l'interface interne /etc/hostname.fxp0 est :

inet 172.18.0.2 255.255.0.0 NONE
inet alias 172.18.0.3 255.255.255.255

Nous avons défini un alias, plus pour l'exemple que pour son utilité réelle dans la configuration actuelle.

#### Configuration des éléments du réseau (routeurs, firewall)

Comme nous l'avons mentionné dans la partie expliquant la configuration du réseau 3S, le routeur externe reliant le firewall et la passerelle à l'Internet assure un filtrage sur le trafic en provenance de l'Internet, afin que les tentatives de connexion directement sur l'interface du firewall et des autres routeurs soient rejetées, tout en laissant passer les connexions vers les serveurs dans la DMZ. Pour que la passerelle soit acessible, il faut donc modifier cette liste d'accès et y ajouter l'adresse IP publique de la passerelle.

Dans le cas de l'implantation de la passerelle dans la DMZ, il faudra aussi modifier la configuration du firewall, pour faire en sorte qu'il reçoive les connexion à destination de la passerelle et les fasse suivre vers la DMZ après avoir converti l'adresse de destination vers l'adresse privée utilisée par la passerelle. Cette modification consiste à ajouter l'adresse IP publique par laquelle les clients accèderont à la passerelle à l'interface externe du firewall. La prochaine section explique les modifications nécessaires pour la translation d'adresse.

### 3.3.4 Firewall et NAT

#### Côté externe

Quelle que soit l'implantation de la passerelle, seul le trafic suivant sera autorisé :

- le trafic utilisé pour les connexion ISAKMP lors de la négociation des clefs : protocole UDP, port 500, dans les deux sens (in et out)
- le trafic utilisé pour le tunnel une fois les SA établies : protocole ESP, port 500, in et out
- le trafic utilisé pour l'administration à distance via ssh : protocole tcp, port 22, in et out. On pourrait aussi utilisé une règle avec mémorisation de l'état (stateful) qui permettrait les trafic pour les demandes de connexion venant de l'extérieur, mais cela n'a pas vraiment d'intéret, le contrôle d'accès étant assuré par le protocole applicatif ssh.

Le serveur ssh est configuré pour vérifier que les adresse IP des hôtes qui se tentent de se connecter appartiennent bien à un domaine enregistré. Pour faire cette vérification il effectue des requètes DNS vers les serveurs de nom du fournisseur d'accès Internet de 3S, ces connexions peuvent également être autorisées : protocole UDP, vers et en provenance du port 53, uniquement pour les adresses des serveurs DNS.

Ces règles de filtrage seront appliquées soit par le firewall si la passerelle se trouve dans la DMZ, soit directement par le système de firewall intégré à la passerelle si celle-ci se trouve en parallèle au firewall.Le fichier de configuration commenté utilisé dans ce dernier cas est fourni en annexe.

En ce qui concerne les translations d'adresse, une règle sera nécessaire uniquement dans le cas d'implantation de la passerelle dans la DMZ. Dans ce cas le firewall devra convertir l'adresse de destination des paquets venant de l'extérieur, en remplaceant l'adresse publique de la passerelle (par l'adresse privée de celle-ci dans la DMZ. L'opération inverse devra être faite pour les paquets venant de la passerelle vers l'extérieur, l'adresse source devra être remplacée par l'adresse publique.

Le fichier de configuration des règles de filtrage se trouve en annexe. L'adresse IP publique de la passerelle a été volontairement changée, et remplacée par l'adresse privée 10.0.0.1.

### Côté interne

En ce qui concerne la connexion de la passerelle au réseau interne, que ce soit directement ou sur le routeur, il faut configurer une conversion d'adresse pour les connexions venant des machines clientes vers le réseau local. En effet, les machines connectées à travers l'Internet utiliseront des adresses publiques attribuées par leur fournisseur d'accès, et si nous laissons les connexions telles quelles il y aura un problème de routage des réponses que les machines locales tenteront d'envoyer. Cette situation est illustrée par la figure 3.4. Le problème est que le routeur interne n'est pas configuré pour acheminer le trafic directement destiné à l'extérieur. Ce routeur est configuré comme la passerelle par défaut des machines du LAN, c'est donc vers lui que celles-ci enverrons le trafic qui n'appartient pas à leur sous-réseau. Le routeur interne enverra ces paquets vers le firewall (car le firewall est la passerelle par défaut du routeur) qui les bloquera.

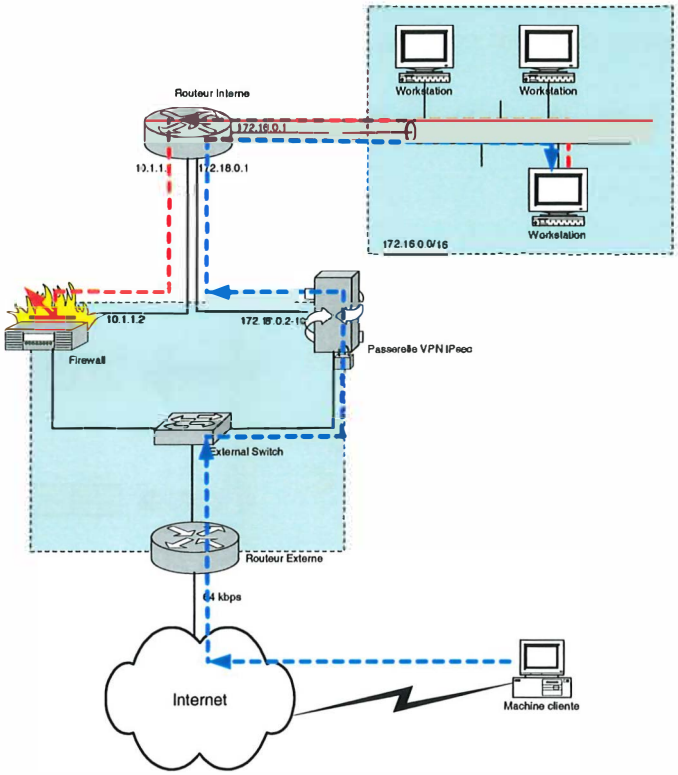



FIG. 3.4 – Problème de routage des adresses publiques sur le LAN

Pour solutionner ce problème, l'adresse source publique des paquets des connexion IPsec seront converties vers une des adresses de l'interface interne de la passerelle IPsec, donc une adresse privée routable sur le LAN. De cette façon il n'y aura pas de problème de routage, et on évite tout trafic portant une adresse publique sur le réseau privé. Pour effectuer cette conversion, on utilisera le module de NAT de la passerelle, en ajoutant une règle de *mapping* qui converti les adresses sources des paquets passant par l'interface interne de la passerelle en une adresse de l'interface (dans notre cas il s'agit d'une seule adresse).

La règle se trouve dans le fichier /etc/ipnat.rules, la voici :

map fxp0 0.0.0.0/0 -> 172.18.0.2/32

Cett règle définit un *mapping* entre toute adresse source passant par l'interface fxp0 (interface interne) à l'adresse 172.18.0.2 qui est l'adresse utilisée dans le cas d'une connexion sur le routeur. Il faut savoir que les règles de mapping sont toujours appliqués au moment où le paquet quitte la machine via l'interface spécifiée, et que la conversion inverse se fera automatiquement quand les paquets correspondant à cette connexion arrivent sur la même interface. Le mécanisme de translation d'adresse est illustré par la figure3.5.

Il y a un problème potentiel dans le cas d'un mapping d'adresse : que se passera-t-il si deux utilisateurs distant se connectent en même temps et établissent une connexion vers une machine du LAN en utilisant le même port TCP source ? Il y a un conflit, l'un des deux verra sa connexion perdue. La partie "portmap" ajoutée à la règle ci-dessous permet de régler ce type de problème : en effet non-seulement l'adresse source sera convertie, mais aussi le port tcp ou udp. De cette façon si le cas de deux connexion à partir du même port se présente, le mapping se fera bien vers la même adresse, mais sur deux ports différents. Les ports vers lesquels se fait la conversion sont compris dans l'intervalle donné dans la règle, ici entre 1025 et 65000.

map fxp0 0.0.0.0/0 -> 172.18.0.2/32 portmap tcp/udp 1025 :65000

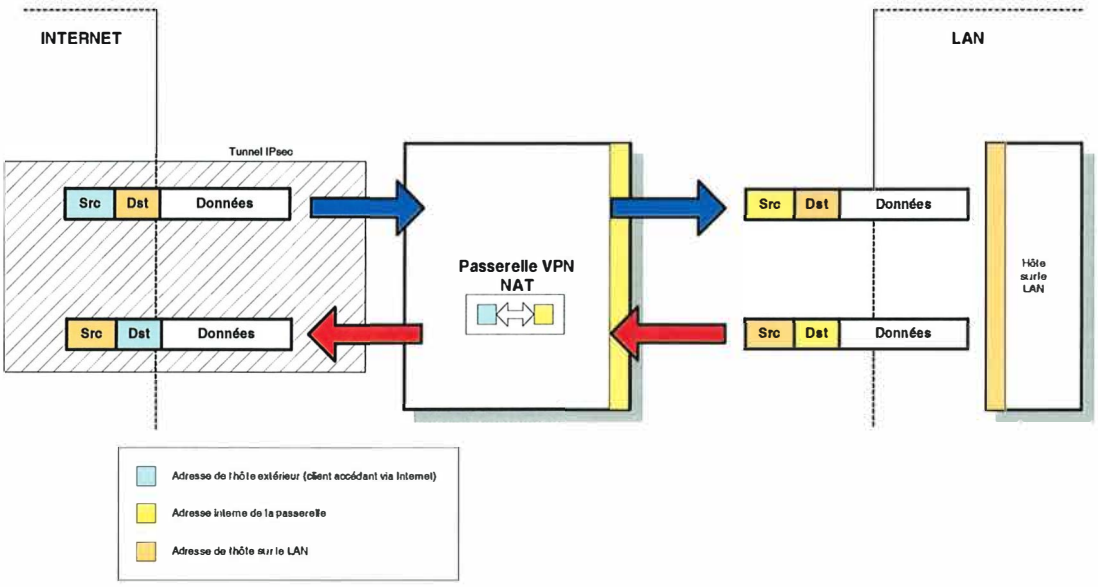



FIG. 3.5 – Mécanisme du NAT sur la passerelle IPsec

Une autre possibilité est de définir la correspondance vers une série d'adresses. Dans ce cas, si la série d'adresses est assez large, le problème de collision décrit plus haut n'aura pas lieu. Cependant la conversion est dynamique, et il n'es pas garanti que deux connexion venant de la même machine distante seront converties vers la même adresse interne. L'idéal serait de définir un mapping fixe entre un hôte distant et une adresse interne au moment où la connexion est établie ; hélas cela n'est pas possible avec cette implémentation. Les correspondances statiques (mapping) ne peuvent se faire que si les deux adresses sont connues à l'avance, hors l'adresse IP des hôte distant sont quasiment toujours dynamiques car allouées par le fournisseur d'accès de l'employé se connectant de chez lui.

### 3.3.5 Configuration ipsec : isakmpd.conf

Le démon isakmpd qui implémente le protocole IKE est configuré à l'aide du fichier /etc/isakmpd/isakmpd.conf. Ce fichier contient les propositions acceptées pour les phases I et II du protocole IKE, chaque proposition étant composée des algorithmes de chiffrement et d'authentification à utiliser, avec leur paramètres d'utilisation et les propriétés à respecter. Reportez-vous à la partie théorique où est décrit IKE pour plus d'information.

Le fichier de configuration complet utilisé pour notre passerelle figure en annexe. Résumons ici son contenu :

- Le premier paramètre *Listen-on* définit l'adresse de l'interface sur laquelle le démon isakmpd attend les demandes de connexion. L'adresse IP réelle à été enlevée du fichier pour des raisons de sécurité.
- Pour la phase I, le type d'échange avec protection d'identité ID-PROT est spécifié. Deux propositions sont spécifiées comme acceptables, l'une utilisant l'algorithme triple-DES (3DES) et l'autre l'algorithme CAST pour le chiffrement ; tous deux utilisent SHA comme algorithme d'authentification et de vérification d'intégrité, le mode d'authentification par signature RSA, et le mode utilisé par l'algorithme Diffie-Hellman est MODP-1024 (la taille du nombre premier n aura donc une taille de 1024 bits. Voir en annexe l'explication de l'algorithme Diffie-Hellman également [23, 19]). Chaque proposition fait également référence à une section définissant les durées de vie maximum des SA ISAKMP à 1 heure (3600 secondes, avec une limite souple de 1800 secondes et une limite rigide de 7200 secondes).
- Pour les phases II, Les utilisateurs ayant déjà accompli la phase I avec succès pourrons établir des SA IPsec pour acheminer le trafic vers les hôtes du sous-réseau défini dans la rubrique [3S-LAN], dont l'adresse est 172.16.0.0/12 (masque de sous-réseau : 255.240.0.0). Ce sous-réseau est celui qui inclus les deux sous-réseaux internes 3S.
  Le type d'échange Quick-Mode sera utilisé. Deux propositions sont définies comme acceptables, l'une à nouveau utilisant 3DES et l'autre CAST pour le chiffrement. Toutes deux spécifient l'utilisation du protocole ESP, le mode tunnel pour l'encapsulation, la propriété de *Perfect-Forward Secrecy* (qui entraîne une nouvel échange de clef) avec le groupe MODP-1024 pour l'échange de clef Diffie-Hellman. Deux types de durée de vie sont spécifiées : l'une sur le temps (600 secondes, limite souple=450secs, limite rigide=720secs), l'autre sur l'importance du trafic échangé (8MégaBytes, limite souple=4MB, limite rigide=10MB).
- La dernière section décrit les répertoires dans lesquels les certificats et la clef privée de la passerelle sont stockés.

La présence de deux propositions a en fait pour but d'observer la négociation au cours des tests. En effet, dans notre cas les clients qui se connecteront n'enverront qu'une seule proposition spécifiant l'algorithme CAST pour le chiffrement. Le choix des durées de vie pour les SA des deux phases s'est fait sur base d'exemples trouvés dans la documentation, et également sur base de tests. La SA de phase I n'est utilisée que pour négocier les SA IPsec, de plus c'est pendant la phase I que s'effectue l'authentification qui est une opération coûteuse. C'est pour cela que sa durée de vie est bien supérieure à celle des SA IPsec négociées par une phase II.

La durée de vie en terme de trafic pour les SA IPsec (phase II) est telle que, pour une ligne de 64kb utilisée à son rendement maximum 8MB seront transmis en 131 secondes. Cela dépendra fortement du type d'application utilisée à distance, le cas d'un transfert de fichier sera sans doute le plus gros consommateur de bande passante sur la ligne. Il serait intéressant

d'approfondir les recherches, pour déterminer à partir de quelle quantité de données transmise via un tunnel tel que spécifié une attaque par analyse cryptographique devient possible. Quoi qu'il en soit, une SA IPsec sera renouvelée toutes les 600 secondes, et ne sera jamais utilisée au delà des 720 secondes de limite rigide (ou au delà de 10MB transmis).

Le lecteur désireux d'en apprendre plus sur les différents algorithmes cryptographiques utilisés trouvera une source d'information très complète dans [1].

### 3.3.6 Gestion des certificats, PKI

Repassons tout d'abord en revue ce que sont les certificats, et à quoi ils servent. Le système de certificat repose sur l'utilisation de la cryptographie à clef publique, où chaque participant d'une communauté d'utilisateurs possède une pair de clef, l'une dite privée qu'il doit gardée secrète, et l'autre publique qu'il peut envoyer aux autres participants. Pour rappel, des données chiffrées avec la clef publique d'un participant ne pourra être déchiffré qu'avec la clef privée correspondante, ainsi le participant propriétaire de cette clef privée sera le seul à pouvoir déchiffrer les données. Selon le même principe, un participant pourra signer un ensemble de données, en chiffrant avec sa clef privée un hash[6] de ces données ; les autres participants qui possèdent la clef publique de ce membre peuvent alors vérifier que c'est bien le membre propriétaire de la clef privée correspondante qui à chiffré le hash, car ce qui est chiffré avec la clef privée ne peut être déchiffré qu'avec la clef publique correspondante.

Dans un tel système, les participants peuvent s'échanger leur clefs publiques entre eux, et si il existe une confiance suffisante entre les différents participant l'un pourra transmettre la clef publique d'un autre. Mais cela pose des problèmes dans les communautés où les participants sont très nombreux et ne se connaissent pas nécessairement. Pour régler ce problème, tous les membres de cette communauté peuvent décider de faire confiance à un membre particulier, qui servira de référence pour tous, et que l'on appelle "Autorité de Certification" (en anglais "*Certification Authority*", ou *CA*). Son rôle sera d'assurer qu'une clef publique est bien authentique, et qu'elle appartient bien au participant que l'on croit.

Comment l'Autorité de Certification assure-t-elle son rôle, et comment un participant peut-il être certain de l'authenticité d'une clef publique ? Pour assurer l'authenticité de la clef publique d'un participant, l'Autorité de Certification signera (avec sa clef privée) un ensemble de données contenant la clef publique du participant, plus des informations sur ce participant, notamment sur son identité. Ces informations sont structurées de manière hiérarchique, et reflètent une structure en arbre. L'ensemble de ces informations forment ce qu'on appelle le *Distinguished Name* (en abrégé *DN*), et suffit à identifier le participant. L'ensemble signé (la clef publique étant incluse) constitue ce qu'on appelle un *Certificat.* Pour permettre aux participants de vérifier qu'un certificat est bien issu de l'Autorité de Certification, ils utilisent la clef publique de celle-ci, contenue dans un certificat spécial appelé *Root Certificate* dans la terminologie anglophone. ce Root Certificate est le certificat du CA, qu'il aura signé avec sa propre clef. Ce certificat est la clef de voûte du système, il doit être disponible pour tous les participant, accessible à un endroit où on est certain de son authenticité. Et bien sur, comme c'est la clef privée du CA qui sert à signer tous les certificats du système il est crucial que celle-ci reste secrète, elle doit être protégée en conséquence.

Lorsqu'un nouveau participant désire obtenir un certificat, il envoie une requête de certificat au CA. Cette requête est composée de la clef publique et des informations nécessaires

[6] résultat de l'application d'une fonction de hashage à l'ensemble de données

à l'identification du participant, l'ensemble est signé par le participant demandeur. Le CA y ajoutera les informations relatives à la gestion de l'ensemble des certificats (par exemple un numéro de série sera attribué à chaque certificat issu), les information de sa propre identité, et signera le tout, qui devient un certificat issu de ce CA. Un certificat contient donc les données du participant, sa clef publique, ce sous-ensemble étant signé par le participant ; ensuite on trouve les informations d'identification du CA, et la signature de ce dernier sur l'ensemble des données.

Pour la distribution des certificats, il est possible d'utiliser ce qu'on appelle un *directory* : il s'agit d'une base de donnée hiérarchique, optimisée pour les accès en lecture. Le type de directory le plus approprié dans le cas des certificats X.509 est celui répondant à la norme LDAP (Lightweight Directory Access Protocol) qui est une version allégée de la norme X.500. La structure arborescente de ces directories correspond en effet à la suite de champs d'information du *Distinguished Name* des certificats X.509. Ce n'est pas un hasard bien sur, la structure des certificats X.509 a été conçue spécialement pour cela. Ainsi donc le Distinguished Name indique le chemin d'accès vers le certificat du participant dans un directory LDAP. Nous n'utiliserons pas de directory pour notre configuration car le faible nombre de participants ne le justifie pas.

Pour notre configuration, nous aurons besoin d'une autorité de certification, qui authentifiera les certificats de chacune des machines participant au système : la passerelle IPsec et les clients se connectant à distance.

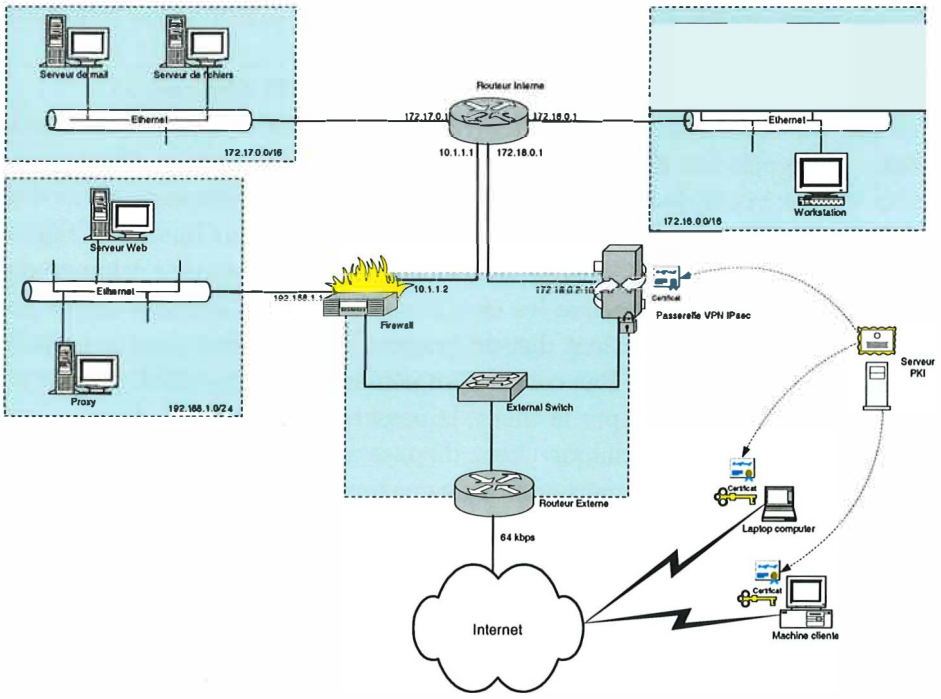



FIG. 3.6 – Passerelle et certificats

Une machine spécifiquement dédiée à la gestion des certificats est utilisée dans notre infrastructure. Cette machine contient sur son disque la clef privée de l'Autorité de Certification,

il est donc capital qu'elle soit en sécurité tant d'un point de vue physique que d'un point de vue réseau. Pour cette raison cette machine n'est pas connectée sur le réseau, les transferts de donnée nécessaires pour le traitement des requêtes de certificats se feront sur disquette. Cette machine est un PC similaire par ses caractéristiques matérielles à la machine utilisée pour la passerelle. C'est également le système OpenBSD qui est installé dessus. Dans la mesure où cette machine n'est pas connectée au réseau aucune mesure spéciale n'a été prise pour en renforcer la sécurité, si ce n'est bien sur une mot de passe solide pour chaque compte utilisé pour la gestion des certificats. La clef privée du CA est elle-même chiffrée, un mot de passe (qui est en fait la clef de chiffrement) est nécessaire pour la déchiffrer à chaque utilisation. Les outils OpenSSL sont utilisés pour accomplir toutes les opérations de gestion des certificats.

Pour les utilisateurs autorisés à se connecter à distance via IPsec, un certificat sera créer. Les certificats sont personnels, attachés à l'identité de l'utilisateur, pas de la machine à partir de laquelle a lieu la connexion. Cependant, comme dans notre cas un utilisateur se connectera toujours à partir de la même machine, nous parlerons de machine dans la suite de cette section. Gardons simplement à l'esprit que quand une machine se connecte via IPsec, il s'agit en fait de l'utilisateur de cette machine qui s'authentifie grâce à son certificat. La seule exception à cette règle est le cas de la passerelle : elle possède son propre certificat en tant que machine.

Chaque machine impliquée dans l'infrastructure IPsec (passerelle, machine cliente) sera en possession de deux certificats au moins :

- une copie du Root Certificate, qui permet de vérifier l'authenticité des autres certificats. Sur la passerelle, il joue aussi un rôle dans la politique de sécurité spécifiée dans le fichier isakmpd.policy dont il était question plus haut : notre politique autorise tout possesseur d'un certificat issu du CA.
- son propre certificat, qu'il pourra transmettre lors de la négociation.
- une copie du certificat de l'interlocuteur sera nécessaire lors de la l'authentification pendant la négociation d'une SA ISAKMP : celui de la machine cliente qui tente de se connecter dans le cas de la passerelle, ou celui de la passerelle dans le cas d'une machine cliente. Ce certificat permettra d'authentifier l'interlocuteur lors de la négociation.
  Le protocole IKE prévoit la transmission des certificats lors de l'échange de la phase I, il n'est donc pas obligatoire que les certificats des clients distants soient stockés sur le disque de la passerelle : le client distant enverra son certificat dans le troisième message de l'échange de la phase I. Par contre l'implémentation OpenBSD ignore le *payload* de requête de certificat envoyé par le client, la passerelle ne transmet donc pas son certificat. Il est donc nécessaire que chaque client dispose à l'avance d'une copie de ce certificat.

Lors de l'authentification par la méthode des signature du protocole IKE qui a lieu pendant la phase I, La passerelle pourra authentifié le client grâce au certificat de ce dernier dont elle détient une copie (soit installée à l'avance, soit acquise lors de l'échange). Ce certificat aura d'abord été vérifié à l'aide du Root Certificate. L'authentification se fait grâce à la clef publique contenue dans le certificat, selon la méthode décrite dans le protocole IKE (voir partie théorique).

De son côté le client authentifiera la passerelle en utilisant exactement la même méthode, lors de la phase I. De cette façon le client est certains de l'identité de la passerelle à laquelle il se connecte, et la passerelle est certaine de l'identité du client qui se connecte, et peut alors autoriser la négociation des SA IPsec au cours de la phase II.

Dans le cas où un certificat viendrait à être volé (si par exemple un employé se fait dérober son ordinateur portable qui contient ses clefs et certificats sur le disque), le certificat devra être

*révoqué*. Dans une infrastructure avec serveur de certificat LDAP et de nombreux utilisateurs utilisant les certificats entre eux, un mécanisme de *Revocation List* est utilisé, pour signaler quels sont les certificats qui ne sont plus dignes de confiance. Cette liste est émise par l'Autorité de Certification et mise à disposition de tous sur le serveur de certificat.

Dans notre cas nous n'utilisons pas de serveur de certificat LDAP, mais ce n'est pas grâve car la gestion de la politique de sécurité est centralisée sur la passerelle IPsec. Pour révoquer un certificat explicitement dans la politique de sécurité, il faut ajouter une condition sur le *Distinguished Name* de l'"*Authorizer*" de la règle, qui indique que les connexions dont le distinguished name du client distant correspond à celui du certificat révoqué doivent être rejetées. Cette condition est spécifiée dans le fichier isakmpd.policy, suivant la syntaxe des expressions régulières, il est donc possible de ne spécifier que certains éléments du DN, comme par exemple le Common Name, dans la condition.

En outre, les certificats ont une validité d'un an, il doivent donc être renouvelés au bout des 365 jours de leur validité. La durée de validité peut être spécifiée en fonction des besoins : un employé pourra être autorisé à utiliser l'infrastructure IPsec pour se connecter au réseau pendant une période donnée, en lui délivrant un certificat dont la période de validité correspond à la période d'autorisation.

#### 3.3.6.1 Création du root certificate

Afin de créer un certificat (que ce soit le root certificate ou un autre), il faut d'abord créer la paire de clefs publique-privée RSA. Ensuite il faut générer une requête de certificat.

La commande suivante génère une paire de clefs RSA de 1024 bits, et la stoque dans le fichier /etc/ssl/private/ca.key après l'avoir chiffrée avec l'algorithme triple-DES, avec le mot de passe monMotDePasse :

```
openssl genrsa -out /etc/ssl/private/ca.key -des3 -passout monMotDePasse 1024
```

La commande ci-dessous génère la demande de certificat. Elle prend en paramètre le fichier de clef pour lequel le certificat devra être créé (ca.key) et le nom du ficher qui contiendra la requête de certificat (ca.csr) :

```
openssl req -new -key /etc/ssl/private/ca.key -out /etc/ssl/private/ca.csr
```

Cette commande est interactive, les éléments du distinguished name seront demandés : localité, ville, unité organisationelle, common name, etc. Une partie de ces paramètres peuvent être configurés dans le fichier /etc/ssl/openssl.cnf. Le mot de passe permettant de déchiffrer la clef privée sera également demandé.

La dernière étape consiste à signer la requête en utilisant le même fichier de clef que celui qui a servi à générer la requête. La clef du root certificate est utilisée pour signer tous les certificats, et cela y compris le root certificate. La commande suivante est utilisée :

```
openssl x509 -req -days 365 -in /etc/ssl/private/ca.csr -signkey /etc/ssl/private/ca.key
-out /etc/ssl/ca.crt
```

Le résultat de cette commande produit le root certificate contenu dans le fichier /etc/ssl/ca.crt. Ce fichier sera distribué aux différents participants (passerelle, machines clientes) afin qu'ils puissent vérifier l'authenticité que les certificats des autres parties sont bien issu de cette même Autorité de Certification.

#### 3.3.6.2 Création des certificats de la passerelle et des clients

Chaque participant (passerelle, utilisateurs distants) doit générer une paire de clef RSA et une requête de certificat à adresser à l'Autorité de Certification pour que celle-ci puisse générer le certificat.

Dans le cas de la passerelle qui utilise elle aussi le système OpenBSD, la génération des clefs et du certificat se fera de la même façon que celle décrite pour le root certificate. Il ne faut toutefois pas chiffrer la clef privée, car le démon isakmpd devra pouvoir y accéder. La commande de génération de clef sera donc la suivante :

```
openssl genrsa -out /etc/isakmpd/private/local.key 1024
```

Le fichier /etc/isakmpd/private/local.key devra avoir des droits en lecture seulement par le super-utilisateur (root), et ne pas être accessible par les autres utilisateurs. Le démon isakmpd vérifiera que les droits sont ainsi paramètrés et refusera de démarrer si ce n'est pas le cas.

La commande de génération de la requête de certificat sera :

```
openssl req -new -key /etc/isakmpd/private/local.key -out /etc/isakmpd/private/10.0.0.1.csr
```

Dans cette exemple le nom du ficher de requête corresond à une adresse IP, celle de la passerelle. C'est une bonne idée, ainsi l'adresse IP qui doit être ajoutée comme information dans le certificat de la passerelle est connue par l'Autorité de Certification.

Les utilisateurs distants qui se connecterons à la passerelle doivent eux aussi générer leurs clefs et leur requête. La procédure est identique et dépend du logiciel client IPsec utilisé, nous verrons cela dans la section dédiée au déploiement des clients.

La requête de certificat d'un participant pourra être transmise par mail pour être traitée par l'Autorité de Certification qui produira le certificat. Pour simplifier les opérations, un petit script qui regroupe les différentes commandes OpenSSL nécessaires pour la création d'un certificat à partir d'une requête de certificat peut être utilisé. Ce script se trouve en annexe. Il contient essentiellement les deux commandes suivantes :

```
openssl x509 -req -days 365 -in <request_file> -CA <CA_cert> -CAkey <CA_key>
    -CAcreateserial -out <newCertFile.crt>
```

Cette commande permet de générer un certificat x509 à partir d'une requête de certificat émise par la passerelle ou par un client. Cette commande prend différents paramètres :

- x509 -req : spécifie le mode de traitement des certificats x509, ici le traitement d'une requête
- -days 365 : spécifie une durée de vie d'un an
- -in <request_file> : indique le chemin et le nom du fichier de requête de certificat
- -CA <CA_cert> : indique le chemein et le nom du fichier de certificat de CA (le *Root Certificate*)
- -CAkey <CA_key> : indique le chemin et le nom du fichier contenant la clef privée de l'Atutorité de Certification.
- -CAcreateserial : indique qu'un numéro de série doit être créé et intégré aux données du certificat
- -out <newCertFile.crt> : indique le chemin et le nom de fichier du nouveau certificat. Ce nom de fichier se termine habituellement par l'extension ".crt".

La commande suivante permet d'ajouter à un certificat x509 une extension qui contiendra l'identité utilisée dans le processus d'identification du protocole ISAKMP. Cette extension

s'appelle "*Subject Alternative Name*". Dans le cas d'un certificat pour un client distant l'identité sera l'adresse email, et la commande sera :

certpatch -t ufqdn -i <email> -k <CA_key> <newCertFile.crt> <newCertFile.crt>

- -t ufqdn : précise le type d'identité, où "ufqdn" signifie "user fully qualified domain name".
- -i <email> : contient les données d'identité à encoder dans le "*Subject Alternative Name*", ici l'adresse email de l'utilisateur distant.
- -k <CA_key> : indique le chemin et le nom du fichier contenant la clef privée de l'Autorité de Certification.
- les deux derniers paramètres indique le fichier d'entrée et celui de sortie. Dans notre cas ce sont les deux même nom puisque on veut modifier le certificat créé par la commande vue précédemment.

Dans le cas de la passerelle, le subject Alternate Name sera l'adresse IP de l'interface externe (celle par laquelle les clients se connectent) :

certpatch -t ip -i <IP_address> -k <CA_key> <newCertFile.crt> <newCertFile.crt>

- -t ip : précise le type d'identité, dans ce cas une adresse IP. Ce paramètre peut ne pas être indiqué car le type ip est celui par défaut.
- -i <IP_address> : contient les données d'identité à encoder dans le "*Subject Alternate Name*", ici l'adresse IP de la passerelle.

Une remarque importante concernant l'adresse IP figurant comme Subject Alternate Name (SAN) dans le certificat de la passerelle : cete adresse doit correspondre à l'adresse IP qui sera envoyée dans le payload d'identité (ID) lors de la négociation. Ce qui veut dire que si la passerelle est connectée derrière un firewall effectuant une translation d'adresse, c'est l'adresse privée de la passerelle qui devra figurer comme SAN dans son certificat, car c'est l'adresse privée qu'elle enverra comme information d'identité. C'est donc bien l'adresse configurée pour l'interface externe de la passerelle qui devra figurer comme SAN dans le certificat.

Pour plus d'informations sur les outils OpenSSL, le lecteur pourra consulter [24, 25].

### 3.3.7 Authentification, politique de sécurité : isakmpd.policy

La politique de sécurité, qui permettra de décider si il y a lieux de poursuivre la négociation d'une SA dés réception du premier message sur base du contenu du payload SA, est spécifiée dans le fichier /etc/isakmpd.policy. Ce fichier contient un ensemble d'assertions dont les conditions seront vérifiées par rapport au contexte de chaque négociation de SA. Lorsque le démon isakmpd adresse un requête au moteur de vérification de politique keynote, il fournit en paramètre le contexte de la SA, qui est une liste d'attributs avec leur valeur. Chaque assertion contient un ensemble de conditions sur la valeur de certains attributs, et si keynote trouve une assertion qui est vérifiée pour le contexte fourni, il donnera son feu vert pour que la négociation continue.

Le fichier isakmpd.policy qui est en place sur la passerelle est semblable à celui représenté ici :

```
Keynote-version : 2
Comment : Authentication based on X.509 certificate

Authorizer : "POLICY"
Licensees : "CA"

Authorizer : "CA"
Licensees : "DN :/C=LU/L=Luxembourg/O=3S/OU=Network Security VPN/CN=CA Root
Certificate/Email=clandrain@3s-cbos.com"
Conditions : app_domain == "IPsec policy"
&& phase_1 == "main"
&& esp_present == "yes"
&& esp_enc_alg != "null"
&& pfs == "yes"
&& esp_encapsulation == "tunnel"
-> "true";
```

Cette politique de sécurité délègue le droit de se connecter en IPsec à tous les clients dont le certificat est issu du *Root Certificate* dont le *Distinguished Name* figure comme valeur du champ *Licensees*. Elle vérifie aussi les conditions suivantes :

- l'application est bien IPsec. Cela peut surprendre, mais c'est tout simplement parce que le système de gestion de confiance Keynote peut être utilisé pour d'autres applications.
- le mode principal (*main mode*) a été utilisé pour établir la SA négociée au cours de la phase I, et pas le mode agressif.
- le protocole à utiliser est ESP.
- l'algorithme de chiffrement n'est pas l'algorithme "null", qui est en fait un moyen d'appliquer un algorithme vide à des fin de diagnostic et de test, mais qui ne devrait jamais être utilisé en situation réelle.
- la connexion se fait avec le mode d'encapsulation *tunnel*, c'est-à-dire que les paquets IP complets sont emballés dans de nouveaux paquets IP après avoir été traités par ESP.
- la propriété de *Perfect Forward Secrecy* est requise, c'est-à-dire qu'un nouvel échange de clef est effectué pour chaque SA IPsec négociée, rendant ainsi les clefs de ces SA indépendantes de celles négociées en phase I.

Si toutes ces conditions sont réunies pour une SA, keynote donne le feu vert pour poursuivre la négociation ; sinon la négociation se terminera par l'envoi d'une notification à la machine client pour lui signifer qu'aucune des propositions faites dans le *payload* SA n'est acceptable.

### 3.3.8 administration, monitoring

Afin de pouvoir administrer la passerelle à distance de manière sécurisée, nous utilisons le protocole SSH (Secure SHell) version 2, avec authentification par clef publique utilisant l'algorithme DSA. Le protocole SSH est en fait la version sécurisée de rsh (Remote Shell), construit sur base de SSL. Grâce à cela nous avons un accès de type terminal à distance, avec une authentification forte (Clef public de 1024 bits de long) dont tous le trafic est chiffré. Toutes les tâches administratives peuvent être effectuées via cet accès. Les lecteurs désireux d'en apprendre d'avantage sur la configuration et l'utilisation de SSH peuvent se référer à [26], ou à [27] où les pages de manuel peuvent être lues en ligne.

Afin de surveiller les tentatives d'accès sur la passerelle, plusieurs mesures on été prises. Tout d'abord le démon isakmpd est configuré pour conserver une trace complète de tous les échanges de négociation. Cette trace consiste à stocker les paquets de négociation dans un format binaire lisible par tcpdump. de cette façon toute tentative de connexion IPsec laissera une trace, et pourra donc être détectée et analysée à posteriori.

En plus de cela le logiciel de détection d'intrusion Snort est installé. Le but est d'observer les activités générales d'un point de vue de sécurité, comme les port scanning et les tentatives de connexion en toute généralité, pour cela les règles de détection par défaut on été installées. Cela provoque biensur beaucoup de fausses alertes, mais le but n'est pas de détecter les attaques ciblées sur isakmpd mais plutôt une surveillance générale. Des informations détaillées sur Snort sont disponible sur la site officiel du projet, voir [28].

Il y a quelque temps de cela une faille de sécurité exploitable à distance fut signalée par l'équipe de développement de OpenSSH pour le serveur SSH pour les versions jusque et incluant la 3.3 (version installée sur la passerelle). Le patch fut disponibles 2 jours plus tard, avant même que l'*exploit* (le code source permettant d'exploiter la faille) soit publié sur l'Internet. La mise à jour qui consistait en la mise à jour des source de SSH grâce à un patch, la recompilation et l'installation des nouveaux fichier exécutables, plus une modification des paramètres de configuration, fut effectuée à distance sans aucun problème. Les quelques semaines qui suivirent, nous avons observer plusieurs tentatives d'utilisation de l'exploit de la faille de SSH. Le fichier journal (log) du module de détection de scanning de ports indiquait des coups de sonde répétés sur le port tcp 22 (port utilisé par SSH), et le journal du démon sshd indiquait lui aussi des connexions faites à partir d'outils de détection de la version de SSH. Aucun de ces tentatives ne donna lieu à une intrusion.

En dehors de cet exploit qui concernait directement la passerelle, de nombreux scannings sont observés régulièrement, à la recherche d'applications comportant des failles de sécurité connues. L'Internet est un milieu hostile, une surveillance constante est nécessaire afin de détecter un éventuel problème et réagir promptement. Il faut également se tenir au courant des actualités en matière de sécurité afin de maintenir le système à jour quand de nouvelles failles sont détectées. Pour cela il existe de nombreux site spécialisés, ainsi que différentes listes de distribution par courrier électronique auxquelles il est possible de s'abonner pour recevoir les nouvelles directement dans sa boite aux lettres électronique.

Profitons de cette occasion pour signaler le site d'un projet intéressant : le projet HoneyNet. Celui-ci consiste à installer des systèmes comportant des failles connues, dans le but d'observer les techniques utilisées par les cyber-criminels pour les exploiter. Plus d'information sur le site du projet [29].

### 3.3.9 Clients : Configuration et déploiement

Afin de permettre aux utilisateurs distant de se connecter via la passerelle IPsec, il fallait choisir une implémentation de programme client IPsec qui pourrait fonctionner sur le système d'exploitation utilisé par ceux-ci, c'est-à-dire les différentes versions du système Windows de Microsoft. Après quelques recherches nous avons choisi de tester le logiciel PGPnet VPN client de Network Associate, d'une part parce que la réputation des logiciels de la suite PGP n'est plus à faire dans le domaine du chiffrement par clef publique, et ensuite parce que ce logiciel offre toutes les fonctionnalités dont nous avons besoin :

- connexion sécurisée IPsec à travers une passerelle
- authentification basée sur l'utilisation de certificats X509

- support de différents algorithmes standards également supportés par la passerelle IPsec OpenBSD.

Un autre point important qui a orienté le choix est que ce logiciel avait déjà été testé dans des configurations centrées sur une passerelle OpenBSD.

Le logiciel PGPnet fait partie d'une suite de logiciels de sécurité incluant plusieurs applications distribuées séparément mais dont la configuration est intégrée. Parmis les autres application on trouve PGPmail qui permet le chiffrement et la signature du courrier électronique, PGPfire qui est un firewall personnel destiné à protéger un ordinateur individuel connecté à l'internet, et PGPdisk offrant la possibilité de créer un disque virtuel sécurisé par chiffrement. Chaque application peut être configurée individuellement, avec certains paramètre partagés par tous les logiciels de la suite. L'application supplémentaire d'administration PGPadmin est également fournie. Cette application permet de créer un fichier exécutable d'installation pour toute ou partie de la suite, et de préconfigurer les paramètres qui pourront être verrouillés pour l'utilisateur final. Cela permet dans notre cas de configurer les paramètres liés à IPsec, également d'inclure le root certificate et celui de la passerelle, de telle sorte que l'utilisateur n'aura plus à s'en soucier. Certaines actions comme la création des clefs et de la requête de certificat peuvent également être déclenchée automatiquement lors de la première exécution de l'application. Ces possibilités rendent le déploiement du logiciel facilement réalisable par les employés eux-même sur leur machine individuelle à domicile.

Les paramètres de configurations devront correspondre exactement avec ceux de la passerelle, cela afin que les propositions des SA puissent correspondre.

Des figures illustrant la configuration des clients sont disponibles en annexe.

### 3.3.10 Tests, problèmes rencontrés, solutions

#### 3.3.10.1 Tests préliminaires sur réseau Local

Le but des tests sur réseau local était de vérifier si le système pouvait fonctionner, avec la possibilité d'expérimenter librement sans se soucier de l'aspect sécurité dans un premier temps. Les tout premiers essais eurent pour but d'établir la première connexion, de se familiariser avec la configuration, d'observer les effet d'un changement des paramètres sur le fonctionnement de la connexion, etc. Ces premiers tests étaient donc de la pure expérimentation, sans autre but que de mieux connaître le fonctionnement de l'ensemble. Il n'y a pas grand chose à dire sur ces premiers essais, la configuration était simplement constitués de deux machine de type PC, l'une configurée avec OpenBSD en tant qu'hôte IPsec acceptant les connexions, et l'autre en tant que client fonctionnant sous Windows et utilisant PGPnet pour établir la connexion.

Nous allons parler plus longuement des tests effectués eux aussi sur réseau local, mais ayant pour but d'établir la faisabilité de la solution d'implantation envisagée pour la passerelle d'une part, et la transparence d'utilisation d'autre part. Pour cela la configuration illustrée par la figure 3.7 fut utilisée.

Le principe du test est simple : la passerelle IPsec est interposée entre ma machine de travail habituelle (la machine sur laquelle j'effectue le développement dans le cadre de mon travail chez 3S) et le réseau local sur lequel se trouvent les autres machines de travail. La passerelle est configurée pour bloquer tout trafic non-protégé par IPsec. Le logiciel PGPnet est installé et configuré sur ma machine, l'idée étant de tester si il est possible d'effectuer toutes les tâches habituelles et d'accéder aux ressources de la même façon que par une connexion directe au LAN.

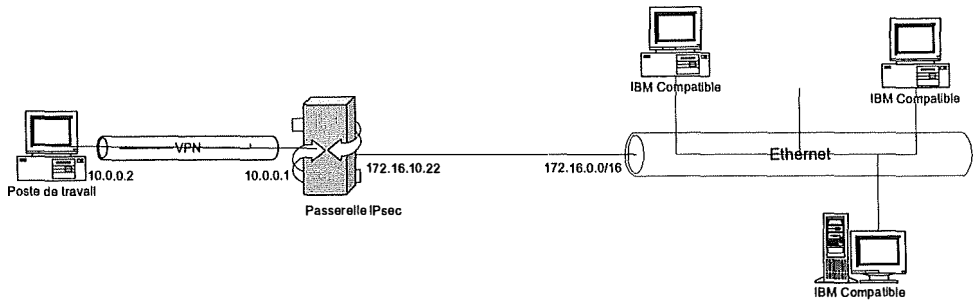



FIG. 3.7 – Configuration de test sur réseau local

J'ai donc passé plusieurs jours à travailler sur mon PC connecté par IPsec au réseau local. le lien entre la station de travail et la passerelle utilisait un sous-réseau privé différent de celui utilisé sur le LAN de 3S, la passerelle effectuait une translation d'adresse vers un alias configuré sur l'interface la reliant au LAN qui correspond à une adresse IP similaire à celle utilisée habituellement par ma machine. Dans l'ensemble tout s'est passé sans problème. Les applications utilisées habituellement dans le cadre du travail journalier et faisant appel au réseau sont :

- le programme de messagerie électronique MS Outlook connecté à un serveur MS Exchange.
- le partage de fichier de Microsoft (protocole SMB)
- le contrôle de machine à distance grâce à un programme de type PC-Anywhere (produit par la société Symantec).

Globalement toutes ces applications passent sans problème à travers le tunnel IPsec, et les ressources sont accessible sans difficultés particulières. Il y a cependant trois différences par rapport à une connexion directe sur le LAN lors de l'utilisation :

- lors du logon, quand l'utilisateur entre son nom et son mot de passe, le tunnel IPsec n'est pas encore ouvert. En effet, il faut que l'utilisateur se connecte d'abord pour pouvoir ouvrir la connexion IPsec. Comme dans le cas de notre LAN l'authentification se fait sur le contrôleur de domaine MS Windows NT, et que celui-ci est inaccessible, l'authentification échoue si le profile d'utilisateur ne se trouve pas déjà sur la machine de travail. Il est également possible d'utilisé un compte local sur la machine de travail, mais alors une authentification sera nécessaire lors du premier accès à une ressources du domaine NT. Lors des accès suivants le nom d'utilisateur et le mot de passe qui auront été stockés dans une mémoire cache interne seront utilisés.
- Lors du premier accès sur le LAN à travers le tunnel, la machine NT publie son nom de machine et son adresse IP auprès du serveur WINS[7], c'est-à-dire le serveur de noms de machine sur les réseaux Microsoft. Cette publication se fait via le protocole NetBios qui est encapsulé par tcp. Le problème qui se pose ici est que l'adresse publiée par la machine de travail derrière la passerelle est son adresse IP locale. Par conséquent le serveur de nom renvoie l'adresse 10.0.0.2 dans notre cas, et cette adresse n'est pas accessible par les autres machines du LAN qui enverront toutes les requêtes destinée à la machine de test vers la passerelle par défaut (le routeur interne reliant les deux parties du LAN

[7]Windows Internet Naming Service

et le firewall), qui elle soit les rejeterra, soit les enverra au firewall qui les rejettera. PAr conséquent les machines du LAN ne peuvent pas accéder à la machine connectée à travers les tunnel, même si elles la voient apparaitre dans la fenêtre "voisinage réseau" de leur système microsoft.
Une solution possible serait d'installer sur la passerelle un programme de type proxy, qui convertirait les paquets contenant l'annonce de nom NetBios en remplaçant l'adresse annoncée (l'adresse IP de la machine de travail) par celle de la passerelle utilisée pour le mapping d'adresse. Le serveur WINS recevrait alors avec le nom de la machine de travail l'adresse de la passerelle.
Mais cette solution ne fonctionnerait qu'avec un mapping statique entre l'adresse de la machine de travaille et celle de la passerelle par le module de translation d'adresse. Hors nous utilisons un mapping dynamique, avec lequel seul les connexions venant de la machine située derrière la passerelle sont mappées.
Dans notre scénario d'utilisation, il n'est pas nécessaire que les machines sur le LAN puisse se connecter aux machines accédant à distance via la passerelle IPsec. Nous n'avons donc pas poussé les investigation plus loin, puisque ce problème n'est pas un obstacle à notre projet.

- Un autre problème lié au mapping dynamique se produit avec les connexions de longue durée, par exemple dans le cas de l'utilisation du programme PC-Anywhere qui se connecte à un serveur de contrôle distant sur une machine du LAN. La connexion s'établit sans problème, et lors d'une utilisation normale continue il n'y a aucun problème. Toutefois dans le cas où la connexion ouverte reste inactive pendant un certain temps, c'est-à-dire sans qu'aucun échange d'information n'ait lieu entre les deux machines, le mapping de port effectué par la passerelle sera réinitialisé. Ce qui aura pour conséquence que lorsque la machine distante voudra reprendre les activités via la connexion un mapping se fera via un nouveau port source, différent de celui utilisé lors de la connexion initiale, et donc la connexion échoue.
  Ce phénomène a été observé plusieurs fois lors de nos tests, il faut cependant noter que le lapse de temps nécessaire à la réinitialisation est assez long, et une utilisation classique ne pose aucun problème. Dans le cas où le travail devait être interrompu pendant une période inhabituelle, il suffit d'interrompre la connexion Pc-Anywhere et d'en ouvrir une nouvelle plutard.

#### 3.3.10.2 Test d'attaque de type Denial of Service

Durant les recherches sur l'implémentation IPsec choisie, j'ai essayé de trouver des rapports ou des exploits[8] concernant des vulnérabilités du démon isakmpd et des protocoles ESP et AH. La seule source d'information réelle à ce sujet, incluant un exploit, est l'article [13] écrit par William A. Simpson, l'un des deux concepteur d'un autre protocole d'échange de clef appelé Photuris. Photuris est un protocole ayant le même rôle que IKE dans l'utilisation d'IPsec, mais se voulant plus simple dans sa conception. Le protocole Photuris est décrit dans [12].

Le problème pour lequel un exploit est fourni est appelé "Cookie Crumb Attack", et repose sur le fait que lors de la réception du premier message d'une négociation IKE en phase 1 le ré-

[8]Un exploit, dans le vocabulaire de la sécurité informatique, est un script ou un programme, en général disponible sur l'Internet, permettant de tester une faille de sécurité sur un système informatique. La question de publier ou non de tels programmes est assez controversée parmi les spécialistes en sécurité informatique, car ils peuvent être utilisés à des fins de test ou pour des attaques réelles.

cepteur créé un état en mémoire, c'est-à-dire qu'il stocke le cookie reçu de l'initiateur ainsi que l'adresse de celui-ci. Il est donc possible, en générant une grande quantité de fausses requètes de connexion IKE avec de fausses adresses source, de provoquer une grande consomation de ressources sur la machine réceptrice. Un mécanisme de "garbage collection" qui récupère la mémoire allouée après un certain lapse de temps est présent dans l'implémentation du démon isakmpd utilisé par OpenBSD, mais selon l'auteur de cet article un tel mécanisme n'est pas suffisant.

Nous avons donc voulu tester la vulnérabilité de notre passerelle vis-à-vis de ce type d'attaque. Pour cela nous avons connecté une machine OpenBSD sur le réseau, du côté de l'interface acceptant les connexions IKE (IPsec), et nous avons lancé plusieurs instances de ce programme contre la passerelle. Le code de l'exploit a du être légèrement modifié, afin que les demandes de connexion générées contiennent une proposition de SA acceptable selon notre politique de sécurité, sans cela l'attaque ne pourrait pas fonctionner.

Sur la passerelle, le démon isakmpd était configuré avec l'option de logging des messages échangés au cours de la négociation, afin de pouvoir vérifier que les fausses demandes de connexion sont bien reçues.

Un extrait du fichier de logging des paquets de négociation décodé par tcpdump montre deux messages échangés pendant l'attaque. Le premier vient de la machine attaquante, on le reconnait à son adresse source 10.78.13.246 (adresse générée de manière aléatoire) et le fait qu'il soit adressé à la passerelle (adresse 192.168.1.3 pour ces tests). On voit aussi que le message est bien traité par la passerelle qui répond en acceptant la proposition par un payload de proposition contenant le même payload de transformation que celui contenu dans le premier message. On observe aussi le mécanisme de cookies, le premier message ne contient que celui de l'initiateur, le second message contient également celui du récepteur. Bien sûr dans le cadre des tests il y a des centaines d'échange de ce type par seconde.

Pendant l'attaque l'utilitaire *top* était utilisé pour surveiller l'état des ressources sur la passerelle. Pendant le test nous avons observé une augmentation continue de la mémoire consommée par le processus du démon isakmpd, jusqu'à atteindre aux environs de 60MB au bout de 3 minutes d'attaque. Cela ne correspond pas exactement à ce qui figure dans l'article, mais il est probable que les tests effectués à l'époque de sa publication furent effectués sur des machines moins puissantes, et que dans notre cas le mécanisme de garbage collecting étant plus rapide a retardé l'augmentation de la consommation mémoire de manière plus efficace, sans toutefois parvenir à stopper l'augmentation. Un fois l'attaque terminée on observe que le démon recycle la mémoire allouée par l'attaque, les évènement de timeout de fausses tentatives de connexion apparaissent dans les messages de débogage. Cependant il semble que la mémoire allouée pendant l'attaque ne soit pas rendue au système d'exploitation, et reste donc allouée au processus isakmpd.

La consommation du temps de calcul du processeur atteint quasiment les 100% d'occupation par le processus isakmpd pendant la durée de l'attaque. Un test fut effectué en lançant l'attaque alors qu'une connexion IPsec était ouverte. Très peu de temps après le début de l'attaque la connexion fut perdue, et ne pû pas être restaurée avant que l'attaque ne cesse. L'attaque en déni de service est donc réellement efficace.

Aucune solution technique n'existe pour parer à ce genre d'attaque. En effet il ne s'agit pas d'une faille dans l'implémentation du protocole ISAKMP, mais bien un défaut dans le protocole lui-même. Il est impossible de pouvoir identifier les fausses requêtes de connexion des vrais, et donc impossible de les bloquer avant leur arrivée au processus isakmpd par

exemple.

Il faut toutefois relativiser. En effet ces tests on été menés sur un réseau local à 10Mb/s de bande passante. La ligne reliant le réseau 3S à l'Internet est un ligne à 64Kb/s, L'impact d'une attaque de ce type à travers une telle ligne sera sans doute sans effet réel sur la passerelle qui dispose de ressources de calcul suffisantes pour traiter les fausses requêtes assez rapidement sans provoquer de surcharge, car le débit d'arrivée des fausses requêtes ne sera pas suffisant. Le problème dans ce cas sera probablement la saturation de la ligne louée, ce qui est un tout autre problème, bien que dans ce cas le déni de service risque d'être étendu à tous les autres services utilisant la ligne louée.

Ce problème du protocole ISAKMP sera sans doute résolu dans une future version du protocole IKE actuellement en développement.

#### 3.3.10.3 Tests "in vivo"

La phase de test suivante est la connexion de la passerelle IPsec à l'Internet, afin de tester si les connexions IPsec fonctionnent bien. La passerelle n'est cependant pas connectée au réseau local.

Dans un premier temps la passerelle fut connectée au réseau de la DMZ, derrière le firewall. ce dernier effectue une translation d'adresse statique, afin que les connexions faites à partir de l'Internet vers une adresse publique soient transmisent à la passerelle qui utilise une adresse privée comme les autres machines de la DMZ. Ce type de connexion n'a pas fonctionné, les paquets entrants arrivaient bien jusqu'à la passerelle, mais celle-ci ne parvenait pas à envoyer les réponses, un message d'erreur relatif au checksum des paquet UDP apparaissait en examinant le traffic réseau à l'aide de tcpdump. Malgré des recherches la raison de ce problème n'a pas pu être trouvée à ce jour. Peut-être est-ce lié à la translation d'adresse par le firewall, mais c'est une pûre hypothèse.

Nous sommes donc passé à l'autre configuration, avec la passerelle connectée en parallèle au firewall. Dans ce cas il n'y a plus de translation d'adresse jusqu'à la passerelle, celle-ci est directement connectée à l'Internet de la même façon que le firewall. Afin de reproduire une situation d'utilisation similaire à celle d'une connexion au réseau local de 3S à travers la passerelle, un PC tournant sous Windows NT est connecté à l'interface interne de la passerelle et simulera une machine de LAN. Cette configuration est illustrée par la figure 3.8.

Cette phase de test a deux objectifs :

1. tester la résistance de la passerelle IPsec à l'environement hostile d'Internet.
2. Tester que les connexions IPsec se déroulent bien, et offrent l'accès aux ressources de la machine NT de manière transparente.

Le premier objectif a pu être vérifié par une surveillance quotidienne des activités réseau reportées par le logiciel de détection d'intrusion Snort, et par les information contenues dans le fichier journal /var/log/authlog, qui contient principalement des informations en provenance du démon sshd qui gère les connexions SSH utilisées pour l'administration à distance. De nombreuses tentatives de connexion furent observées au cours des 3 mois de connexion permanente à l'Internet, particulièrement après l'annonce d'une vulnérabilité dans le serveur OpenSSH. Cette vulnérabilité fut heureusement signalée et corrigée avant la publication de l'exploit correspondant. Le patch fourni sur le site d'OpenSSH fut appliqué le jour même de sa disponibilité.

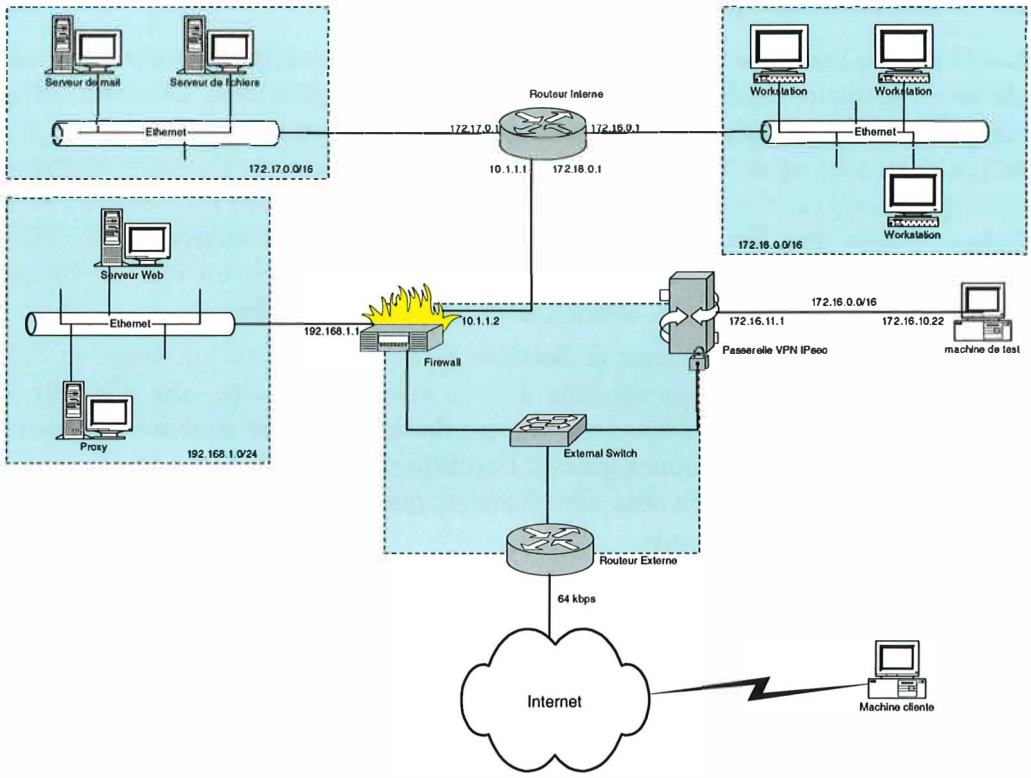



FIG. 3.8 – Configuration de test "in vivo"

Les fichiers *alert* et *portscan.log* générés par Snort permettent de voir les différents scanning de ports qui ont lieu plusieurs fois par jour sur différents services. Le firewall interne du système OpenBSD tel que nous l'avons configuré bloque tout le trafic non-destiné aux services SSH ou ISAKMP qui sont les seuls services activés sur la machine. La plupart des informations fournies par Snort sont donc simplement intéressante pour observer toutes les tentatives de récupération d'information ou d'intrusion auxquelles une machine connectée en permanence sur l'Internet est sujette.

Les activités au niveau du protocole IKE peuvent être surveillées grâce au fichier dans lequel le démon isakmpd reporte systèmatiquement les messages échangés au cours des négociations IKE. Ces informations sont lisibles grâce à tcpdump, un utilitaire destiné à observer le trafic TCP/IP sur une interface réseau, éventuellement stocké dans un fichier. Jusqu'ici aucune tentative de connexion IKE n'a été observée.

#### 3.3.10.4 Exemple de log d'une connexion IPsec

Le lecteur trouvera en annexe les différentes traces collectées par le logiciel client et la passerelle au cours d'une connection à la machine de test à travers l'Internet. Ces logs seront commentés pour en facilité la lecture.

### 3.3.11 Status du projet, perspectives futures

Au moment de l'écriture de ces lignes le projet d'infrastructure d'accès sécurisé au réseau local de 3S est toujours dans la phase de test décrit un peu plus haut. Les tests ont jusqu'ici été concluants. Il y a toutefois un problème de confiance assez compréhensible de la part de la direction de la compagnie vis-à-vis de la sécurité de l'installation. Un audit externe par une companie spécialisée en sécurité des réseaux devrait idéalement être effectué, afin de certifier que l'infrastructure est fiable. Une fois que la fiabilité de la solution sera établie, elle pourra être utilisée par les cadres et les membres du personnel assurant un rôle d'administration technique pour l'accès sécurisé aux ressources du réseau local de 3S.

La société Streamlined Solutions & Services possède également un siège au Sri-Lanka. Une passerelle IPsec pourrait être installée dans le réseau de ce siège, afin d'établir un VPN entre les deux réseaux pour protéger les échanges de données. L'administration peut se faire entièrement à partir du Luxembourg grâce à l'utilisation de SSH.

Il n'y a aucun projet dans ce sens actuellement, mais cela est possible maintenant que le savoir-faire en la matière est acquis.

# Conclusion

Le but essentiel que le projet décrit dans ce document a eu à résoudre est l'absence de sécurité d'une connexion utilisant l'Internet pour accéder aux ressources d'une entreprise. Ces ressources, dans le cas d'une société de services en informatique comme Streamlined Solutions & Services, constituent un avoir important de l'entreprise, et doivent donc être protégées contre l'appropriation ou la détérioration par des tiers.

L'utilisation des protocoles de la suite IPsec, par la mise en oeuvre d'une passerelle sécurisée et d'un système d'authentification par certificat, permet l'accès distant via l'Internet tout en garantissant un niveau de sécurité répondant aux exigences de l'industrie les plus élevées.

Dans ce document nous avons exposé les étapes qui ont permis d'aboutir à un tel résultat. Tout d'abord nous avons vu comment les problèmes liés à l'implantation d'une telle passerelle dans un réseau existant ont été résolu. Plus particulièrement, les problèmes du choix de l'emplacement de la passerelle sur le réseau ainsi que ceux liés au routage interne ont pu être adressés par une analyse adéquate des mécanismes du réseau.

Ensuite nous avons décrit les tests qui ont été menés afin de valider la solution. Cette validation fut fonctionnelle d'une part, pour s'assurer de la transparence d'accès aux ressources à travers l'accès sécurisé. L'autre aspect qui fut validé fut la robustesse et la fiabilité de la solution déployée, cela en mettant l'élément principal de la solution qu'est la passerelle dans une situation réelle de fonctionnement, c'est à dire directement connectée à l'Internet. Par une surveillance quotidienne, nous avons observé que la passerelle est résistante aux conditions hostiles de l'Internet.

La solution décrite semble techniquement prête à être mise en service réel. Il faut toutefois garder à l'esprit que la sécurisation informatique est un problème difficile et complexe, particulièrement dans un cas comme le nôtre où les biens d'une entreprise pourraient courrir un risque. Un audit externe par une société spécialisée devra donc être effectué afin de certifier la validité de la solution. De plus, une surveillance régulière des installations par l'administrateur restera toujours nécessaire, des systèmes de surveillance automatiques peuvent aider dans cette tâche.

Au cours du projet nous avons également pu découvrir à quel point toute machine connectée à l'Internet est exposée en permanence à des tentatives d'intrusion en tout genre. La sécurité informatique est un enjeu très important, et il est très probable qu'à l'avenir l'utilisation de technologies de type IPsec fera partie du quotidien.

# Bibliographie


[1] B. Schneier, *Applied Cryptography Second Edition : Protocols, Algorithms, and Source Code in C.* John Wiley & Son, Inc, 1996.

[2] W. Odom, *Préparation à la certification CCNA.* CampusPress France, 1999.

[3] "Iana assigned port numbers." http ://www.iana.org/assignments/port-numbers.

[4] J. Postel, "User datagram protocol." RFC0768, Aout 1980.

[5] J. Postel, "Transmission control protocol." RFC0793, Septembre 1981.

[6] G. Cizault, *IPv6, Théorie et pratique.* 2ème ed., 1999.

[7] M. A. Miller, *Implementing IPv6.* M&T, second ed., 1999.

[8] D. C. Plummer, "Ethernet address resolution protocol : Or converting network protocol addresses to 48 bits ethernet address for transmission on ethernet hardware." RFC826, Novembre 1982.

[9] J. Postel, "Internet control message protocol." RFC0792, Septembre 1981.

[10] S. M. Bellovin, "Security problems in tcp/ip protocol suite," *Computer Communications Review*, pp. 32–48, Avril 1989.

[11] C. R. Davis, *IPSec, Securing VPNs.* ISBN 0-07-212757-0, Osborne/Mc Graw Hill, 2001.

[12] P. Karn and W. Simpson, "Photuris : Session-key management protocol." RFC 2522, Mars 1999.

[13] W. A. Simpson, "Ike/isakmp considered harmful," tech. rep., http ://www.usenix.org/publications/login/1999-12/features/harmful.html, Décembre 1999.

[14] S. Kent and P. Atkinson, "Security architecture for the internet protocol." RFC2401, Novembre 1998.

[15] S. Kent and R. Atkinson, "Ip encapsulating security payload (esp)." RFC 2406, Novembre 1998.

[16] S. Kent and R. Atkinson, "Ip authentication header." RFC 2402, Novembre 1998.

[17] D. Piper, "The internet ip security domain of interpretation for isakmp." RFC 2407, Novembre 1998.

[18] M. S. D. Maughan, M. Schertler and J. Turner, "Internet security association and key management protocol (isakmp)." RFC 2408, Novembre 1998.

[19] D. Harkins and D. Carrel, "The internet key exchange (ike)." RFC 2409, Novembre 1998.

[20] OpenBSD project, http ://www.openbsd.org, *OpenBSD Official Web Site.*

[21] N. Hallqvist and A. D. Keromytis, "Implementing internet key exchange (ike)," tech. rep., USENIX, 2000.

[22] *Unix System Administration Handbook, third edition.* 2000.

[23] H. Orman, "The oakley key determination protocol." RFC2412, Novembre 1998.

[24] OpenSSL project, http ://www.openssl.org, *OpenSSL Official Web Site.*

[25] J. Granstam, *How to use X.509v3 certificates for authentication with isakmpd.* http ://hem.passagen.se/hojg/isakmpd, 2000.

[26] D. J. Barrett and R. E. Silverman, *SSH, The Secure Shell, The definitive Guide.* O'Reilly, 2001.

[27] OpenSSH project, http ://www.openssh.org, *OpenSSH Official Web Site.*

[28] Snort project, http ://www.snort.org, *Snort Project Official Web Site.*

[29] The HoneyNet project, http ://www.honeynet.org, *The HoneyNet project Web Site.*

# Annexes

# Annexe A

# Algorithme d'échange de clef Diffie-Hellman

La description qui vous est fournie ici est une traduction des pages 513 à 514 du livre "Applied Cryptography" de Bruce Schneier [1].

---

Diffie-Hellman fut le premier algorithme à clef publique jamais inventé, et remonte à 1976. Il tire sa sécurité de la difficulté de calculer un logarithme discret dans un champ fini, alors qu'il est très facile de calculer une exponentiation dans le même champ fini. Diffie-Hellman peut être utilisé pour la distribution de clefs - Alice et Bob[1] peuvent utiliser cet algorithme pour générer une clef secrète - mais il ne peut pas être utilisé pour chiffrer et déchiffrer des messages.

Les mathématiques utilisées sont simples. Tout d'abord Alice et Bob conviennent d'un grand nombre premier, n et g, tel que g est premier modulo n. Ces deux entiers ne doivent pas être secrets ; Alice et Bob peuvent le négocier en utilisant n'importe quel canal de communication non sécurisé. Ces nombres peuvent même être commun à un groupe d'utilisateur. Ca n'a pas d'importance.

La suite du protocole se déroule comme suit :

1. Alice choisit un grand nombre premier aléatoire x and envoie à Bob
   $X = g^x \bmod n$
2. Bob choisit un grand nombre premier aléatoire y et envoie à Alice
   $Y = g^y \bmod n$
3. Alice calcule
   $k = Y^x \bmod n$
4. Bob calcule
   $k' = X^y \bmod n$

$k$ et $k'$ sont égal à $g^{xy} \bmod n$. Aucune personne à l'écoute sur le canal d'échange ne peut calculer cette valeur ; ils ne connaissent que n, g, X et Y. A moins de pouvoir calculer le

[1] Alice et Bob sont les personnages traditionnellement utilisés dans les exemples explicatifs des algorithmes de sécurité. Alice et Bob sont les utilisateurs du mécanisme de sécurité, Malory est un attaquant qui essaye de déjouer le mécanisme.

logarithme discret et de récupérer x ou y, ils ne peuvent résoudre le problème. Donc, k est la clef secrète qu'Alice et Bob ont calculé indépendemment l'un de l'autre.

Le choix de g et n peut avoir un impact substanciel sur la sécurité de ce système. Le nombre $(n-1)/2$ devrait aussi être un nombre premier. Et plus important encore, n devrait être grand : la sécurité du système est basée sur la difficulté de factoriser des nombres de la même taille que n. Vous pouvez choisir n'importe quel g tel que g est premier modulo n ; il n'y a pas de raison de ne pas choisir le plus petit g possible - en général un nombre à un chiffre. (Et en fait, g ne doit pas nécessairement être premier ; il doit juste générer un grand sous-groupe du groupe multiplicatif modulo n.)

# Annexe B

# Fichiers de configuration de la passerelle

Les fichiers de configurationdonnés ici sont ceux de l'infrastructure de test décrite par la figure 3.8 en page 89.

## B.1 /etc/isakmpd.conf

```
# configuration for the isakmpd ISAKMP/Oakley (aka IKE) daemon.
#
# The network topology of the example net is like this:
#
# Internet - GateKeep [w.x.y.z] - 172.16.0.0/12 (LAN)
#

[General]
Listen-on=        w.x.y.z

[Phase 1]
Default=          ROAMINGUSER

[ROAMINGUSER]
Phase=                  1
Transport=        udp
Configuration=          Default-main-mode

[Default-main-mode]
DOI=              IPSEC
EXCHANGE_TYPE=          ID_PROT
Transforms=       CAST-SHA, 3DES-SHA

# Phase 1 Proposals
########################
#  AUTHENTICATION_METHOD = RSA_SIG forces the use of x509 certificates.

[3DES-SHA]
ENCRYPTION_ALGORITHM=   3DES_CBC
HASH_ALGORITHM=         SHA
AUTHENTICATION_METHOD=  RSA_SIG
GROUP_DESCRIPTION=      MODP_1024
Life=             LIFE_3600_SECS

[CAST-SHA]
ENCRYPTION_ALGORYTHM=   CAST_CBC
HASH_ALGORITHM=         SHA
AUTHENTICATION_METHOD=  RSA_SIG
GROUP_DESCRIPTION=      MODP_1024
Life=             LIFE_3600_SECS


# Phase 2
######################
#

[Phase 2]
Passive-connections=    VPN-3S-ROAMINGUSER

[VPN-3S-ROAMINGUSER]
Phase=                  2
ISAKMP-peer=            ROAMINGUSER
Configuration=          Default-quick-mode
Local-ID=         3S-LAN

[3S-LAN]
ID-type=          IPV4_ADDR_SUBNET
Network=          172.16.0.0
Netmask=          255.240.0.0

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

# Phase 2 Proposals
######################
#

[QM-ESP-PFS-SUITE]
Protocols=        QM-ESP-PFS
```

```
[QM-ESP-PFS]
PROTOCOL_ID=            IPSEC_ESP
Transforms=       QM-ESP-CAST-SHA-PFS-XF, QM-ESP-3DES-SHA-PFS-XF

[QM-ESP-3DES-SHA-PFS-XF]
TRANSFORM_ID=           3DES
ENCAPSULATION_MODE=     TUNNEL
AUTHENTICATION_ALGORITHM=     HMAC_SHA
GROUP_DESCRIPTION=      MODP_1024
Life=             LIFE_600_SECS, LIFE_4_MEG

[QM-ESP-CAST-SHA-PFS-XF]
TRANSFORM_ID=           CAST
ENCAPSULATION_MODE=     TUNNEL
AUTHENTICATION_ALGORITHM=     HMAC_SHA
GROUP_DESCRIPTION=      MODP_1024
Life=             LIFE_600_SECS, LIFE_4_MEG


# Lifetime definitions
########################
#

[LIFE_600_SECS]
LIFE_TYPE=        SECONDS
LIFE_DURATION=          600,450:720

[LIFE_3600_SECS]
LIFE_TYPE=        SECONDS
LIFE_DURATION=          3600,1800:7200

[LIFE_4_MEG]
LIFE_TYPE=        MEGABYTES
#LIFE_DURATION=         4,1:6
LIFE_DURATION=          8,4,10

# x509 locations
####################
#

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

## B.2 /etc/isakmpd.policy

```
Keynote-version: 2
Comment:    Authentication based on X.509 certificate
Authorizer: "POLICY"
Licensees:  "CA"

Authorizer: "CA"
Licensees:  "DN:/C=LU/L=Luxembourg/O=3S/OU=Network Security VPN/CN=CA Root Certificate/Email=clandrain@3s-cbos.com"
Conditions: app_domain == "IPsec policy"
            && phase_1 == "main"
            && esp_present == "yes"
            && esp_enc_alg != "null"
            && pfs == "yes"
            && esp_encapsulation == "tunnel"
            -> "true";
```

## B.3 /etc/ipf.rules

```
#     $OpenBSD: ipf.rules,v 1.6 1997/11/04 08:39:32 deraadt Exp $
#
# IP filtering rules.  See the ipf(5) man page for more
# information on the format of this file, and /usr/share/ipf
# for example configuration files.
#
# edit the ipfilter= line in /etc/rc.conf to enable IP filtering
#


#First we block everything
block in on de0 all

#Now we drop all packets with non-routable IPs
block in quick on de0 from 0.0.0.0/7 to any
block in quick on de0 from 2.0.0.0/8 to any
block in quick on de0 from 5.0.0.0/8 to any
block in quick on de0 from 10.0.0.0/8 to any
block in quick on de0 from 23.0.0.0/8 to any
block in quick on de0 from 27.0.0.0/8 to any
block in quick on de0 from 31.0.0.0/8 to any
block in quick on de0 from 69.0.0.0/8 to any
block in quick on de0 from 70.0.0.0/7 to any
block in quick on de0 from 72.0.0.0/5 to any
block in quick on de0 from 82.0.0.0/7 to any
block in quick on de0 from 84.0.0.0/6 to any
block in quick on de0 from 88.0.0.0/5 to any
block in quick on de0 from 96.0.0.0/3 to any
block in quick on de0 from 127.0.0.0/8 to any
block in quick on de0 from 128.0.0.0/16 to any
block in quick on de0 from 128.66.0.0/16 to any
block in quick on de0 from 169.254.0.0/16 to any
block in quick on de0 from 172.16.0.0/16 to any
block in quick on de0 from 191.255.0.0/16 to any
block in quick on de0 from 192.0.0.0/19 to any
block in quick on de0 from 192.0.48.0/20 to any
block in quick on de0 from 192.0.64.0/18 to any
block in quick on de0 from 192.0.128.0/17 to any
block in quick on de0 from 192.168.0.0/16 to any
block in quick on de0 from 197.0.0.0/8 to any
block in quick on de0 from 201.0.0.0/8 to any
block in quick on de0 from 204.152.64.0/23 to any
block in quick on de0 from 224.0.0.0/3 to any


#Let's allow ESP and SSH
pass in quick on de0 proto udp from any to w.x.y.z/32 port = 500
pass in quick on de0 proto esp from any to w.x.y.z/32
pass in quick on de0 proto tcp from any to w.x.y.z/32 port = 22 flags S keep state
keep frags

# Allow DNS connections
pass in quick on de0 proto udp from 194.154.192.1/32 port = 53 to w.x.y.z/32
pass in quick on de0 proto udp from 194.154.192.101/32 port = 53 to w.x.y.z/32
pass in quick on de0 proto udp from 194.154.192.102/32 port = 53 to w.x.y.z/32

pass out quick on de0 proto udp from w.x.y.z/32 to 194.154.192.1 port = 53
pass out quick on de0 proto udp from w.x.y.z/32 to 194.154.192.101 port = 53
pass out quick on de0 proto udp from w.x.y.z/32 to 194.154.192.102 port = 53


# Allow ntp connections
pass out quick on de0 proto udp from w.x.y.z/32 to 195.13.23.5 port = 123
pass out quick on de0 proto udp from w.x.y.z/32 to 195.13.1.153 port = 123
pass in quick on de0 proto udp from 195.13.23.5 port = 123 to w.x.y.z/32
pass in quick on de0 proto udp from 195.13.1.153 port = 123 to w.x.y.z/32


#
block out on de0 all
```

```
block out quick on de0 from 172.0.0.0/8 to any
block out quick on de0 from 172.0.0.0/8 to 0.0.0.0/7
block out quick on de0 from 172.0.0.0/8 to 2.0.0.0/8
block out quick on de0 from 172.0.0.0/8 to 5.0.0.0/8
block out quick on de0 from 172.0.0.0/8 to 10.0.0.0/8
block out quick on de0 from 172.0.0.0/8 to 23.0.0.0/8
block out quick on de0 from 172.0.0.0/8 to 27.0.0.0/8
block out quick on de0 from 172.0.0.0/8 to 31.0.0.0/8
block out quick on de0 from 172.0.0.0/8 to 69.0.0.0/8
block out quick on de0 from 172.0.0.0/8 to 70.0.0.0/7
block out quick on de0 from 172.0.0.0/8 to 72.0.0.0/5
block out quick on de0 from 172.0.0.0/8 to 82.0.0.0/7
block out quick on de0 from 172.0.0.0/8 to 84.0.0.0/6
block out quick on de0 from 172.0.0.0/8 to 88.0.0.0/5
block out quick on de0 from 172.0.0.0/8 to 96.0.0.0/3
block out quick on de0 from 172.0.0.0/8 to 127.0.0.0/8
block out quick on de0 from 172.0.0.0/8 to 128.0.0.0/16
block out quick on de0 from 172.0.0.0/8 to 128.66.0.0/16
block out quick on de0 from 172.0.0.0/8 to 169.254.0.0/16
block out quick on de0 from 172.0.0.0/8 to 172.16.0.0/16
block out quick on de0 from 172.0.0.0/8 to 191.255.0.0/16
block out quick on de0 from 172.0.0.0/8 to 192.0.0.0/19
block out quick on de0 from 172.0.0.0/8 to 192.0.48.0/20
block out quick on de0 from 172.0.0.0/8 to 192.0.64.0/18
block out quick on de0 from 172.0.0.0/8 to 192.0.128.0/17
block out quick on de0 from 172.0.0.0/8 to 192.168.0.0/16
block out quick on de0 from 172.0.0.0/8 to 197.0.0.0/8
block out quick on de0 from 172.0.0.0/8 to 201.0.0.0/8
block out quick on de0 from 172.0.0.0/8 to 204.152.64.0/23
block out quick on de0 from 172.0.0.0/8 to 224.0.0.0/3

#Let's allow the outgoing traffic for ESP
pass out quick on de0 proto udp from w.x.y.z/32 to any port = 500
pass out quick on de0 proto esp from w.x.y.z/32 to any
```

## B.4 /etc/ipnat.rules

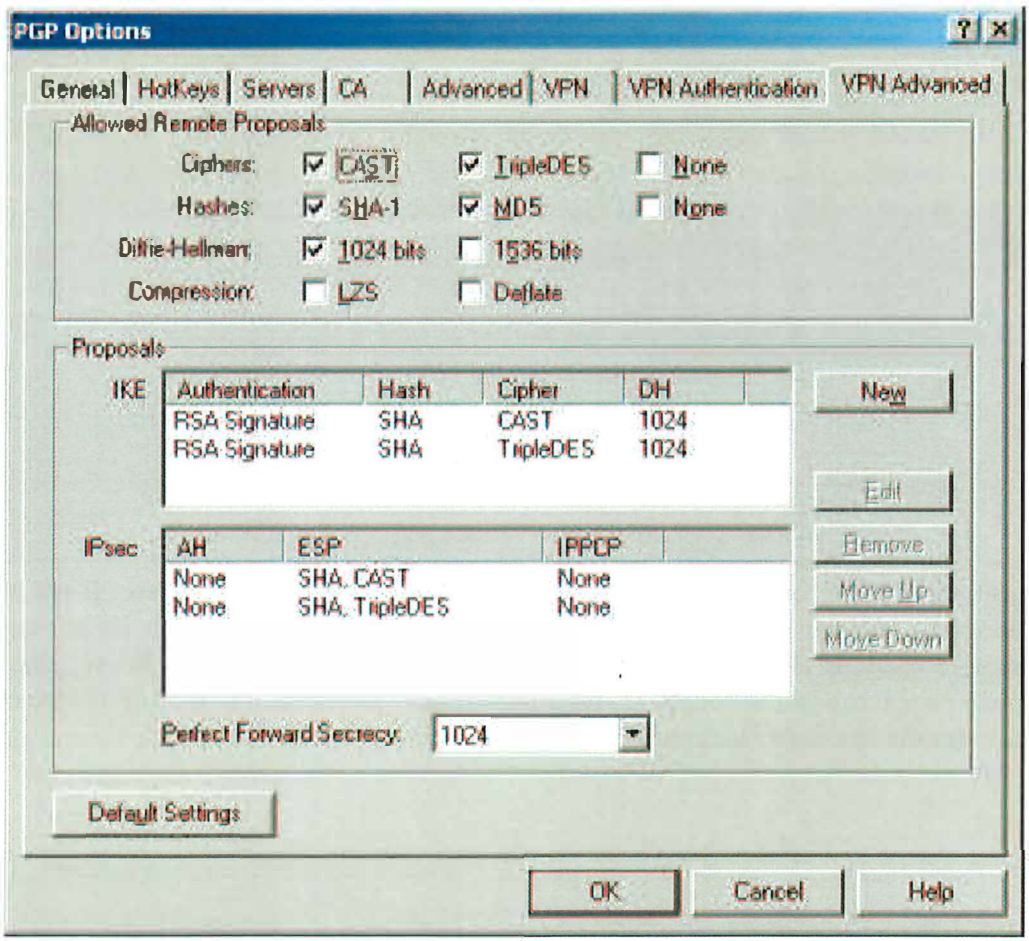
```
# $OpenBSD: ipnat.rules,v 1.2 1999/05/08 16:33:10 jason Exp $
#
# See /usr/share/ipf/nat.1 for examples.
# edit the ipnat= line in /etc/rc.conf to enable Network Address Translation


map fxp0 0.0.0.0/0 -> 172.16.11.1/32 portmap tcp/udp 1025:65000
```

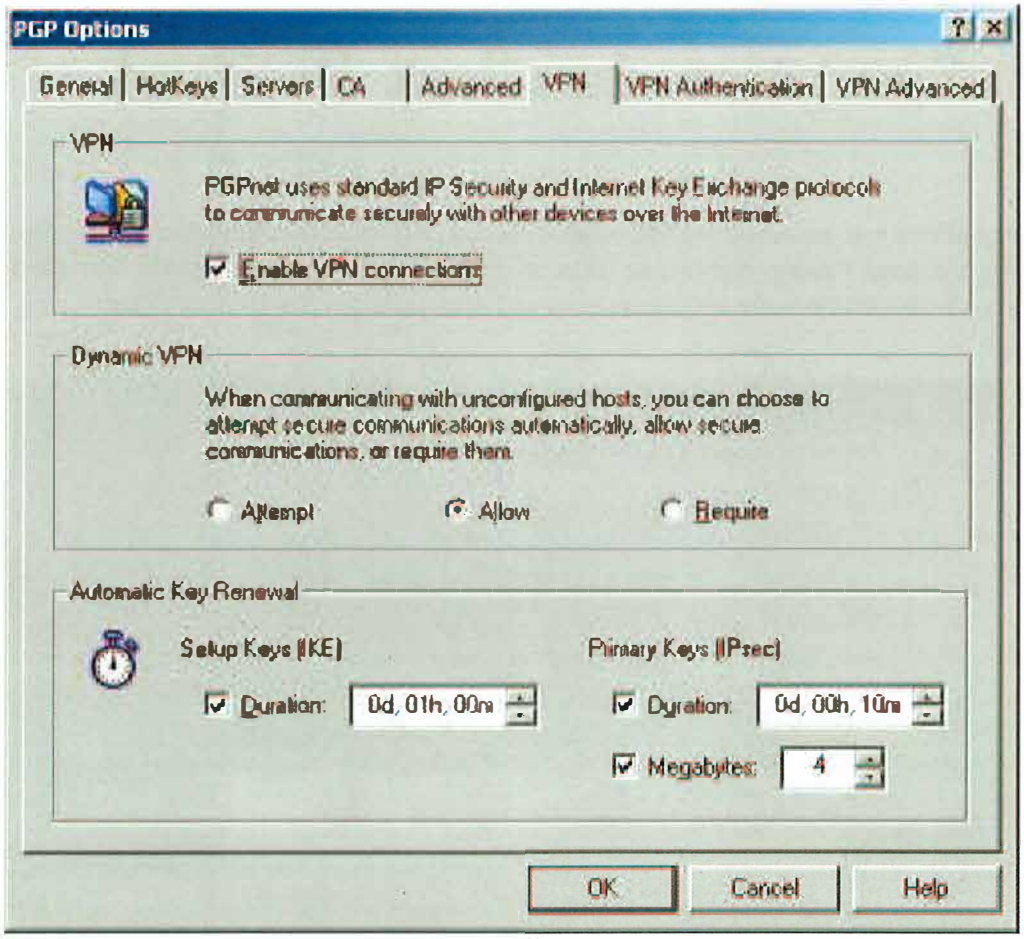
# Annexe C

# Configuration du client PGPnet

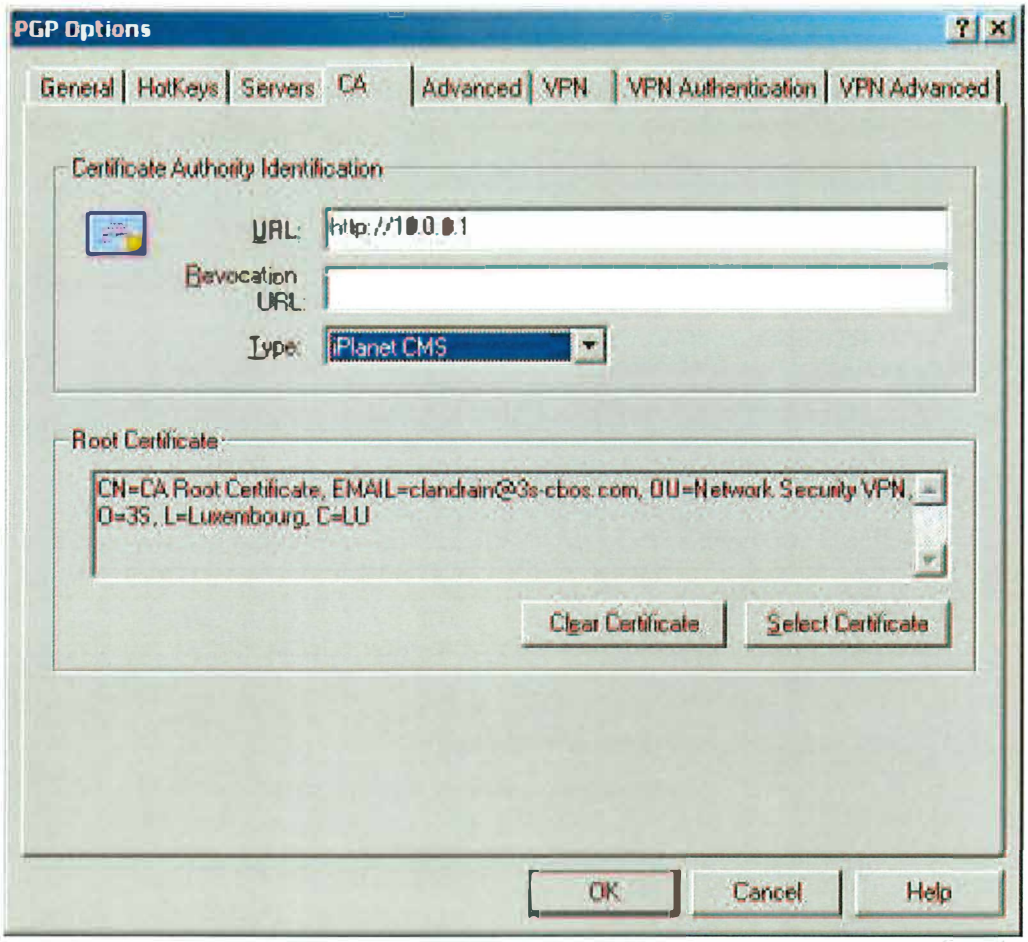
Nous allons voir ensemble les captures d'écrans illustrant la configuration des clients PGPnet. Comme nous l'avons mentionné dans le chapitre 3 cette configuration doit correspondre à celle de la passerelle pour les paramètres utilisées pour les SA IKE et IPsec.



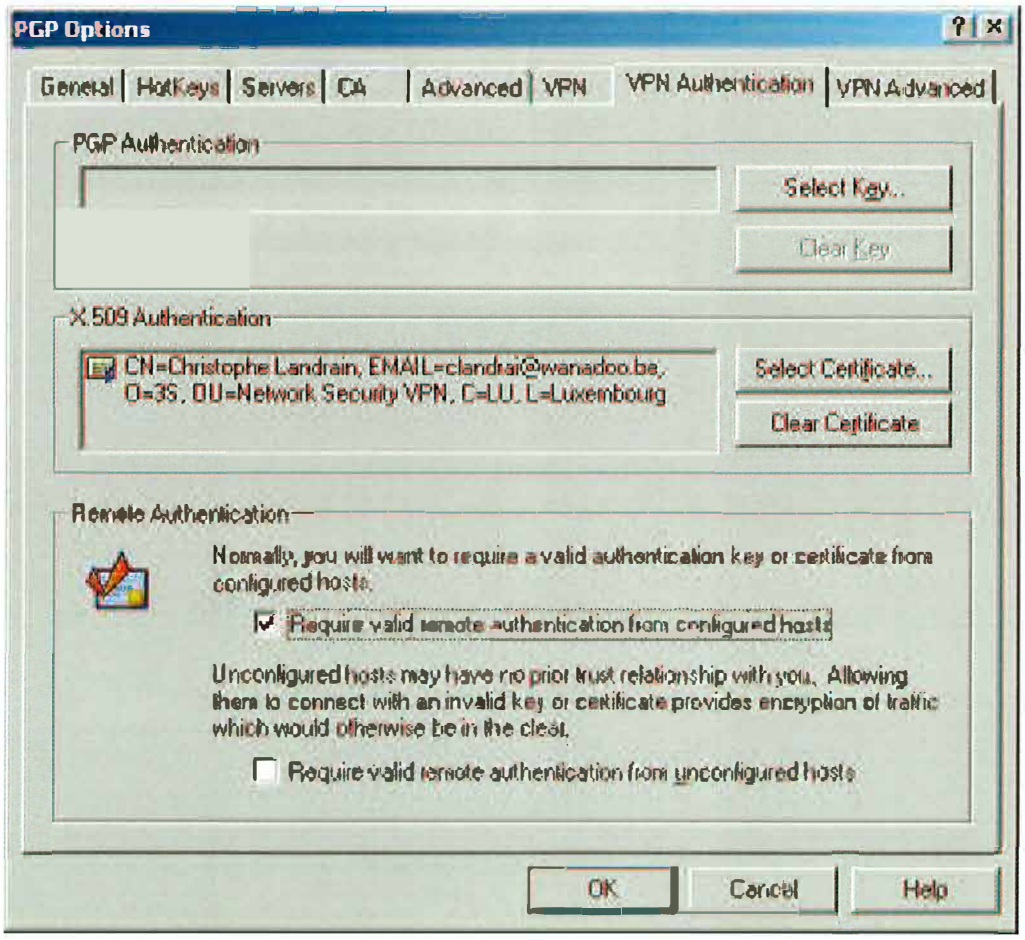

Ce premier écran montre la configuration des différentes propositions acceptables pour IKE et IPsec. L'ordre des propositions détermine la préférence lors de la négociation.



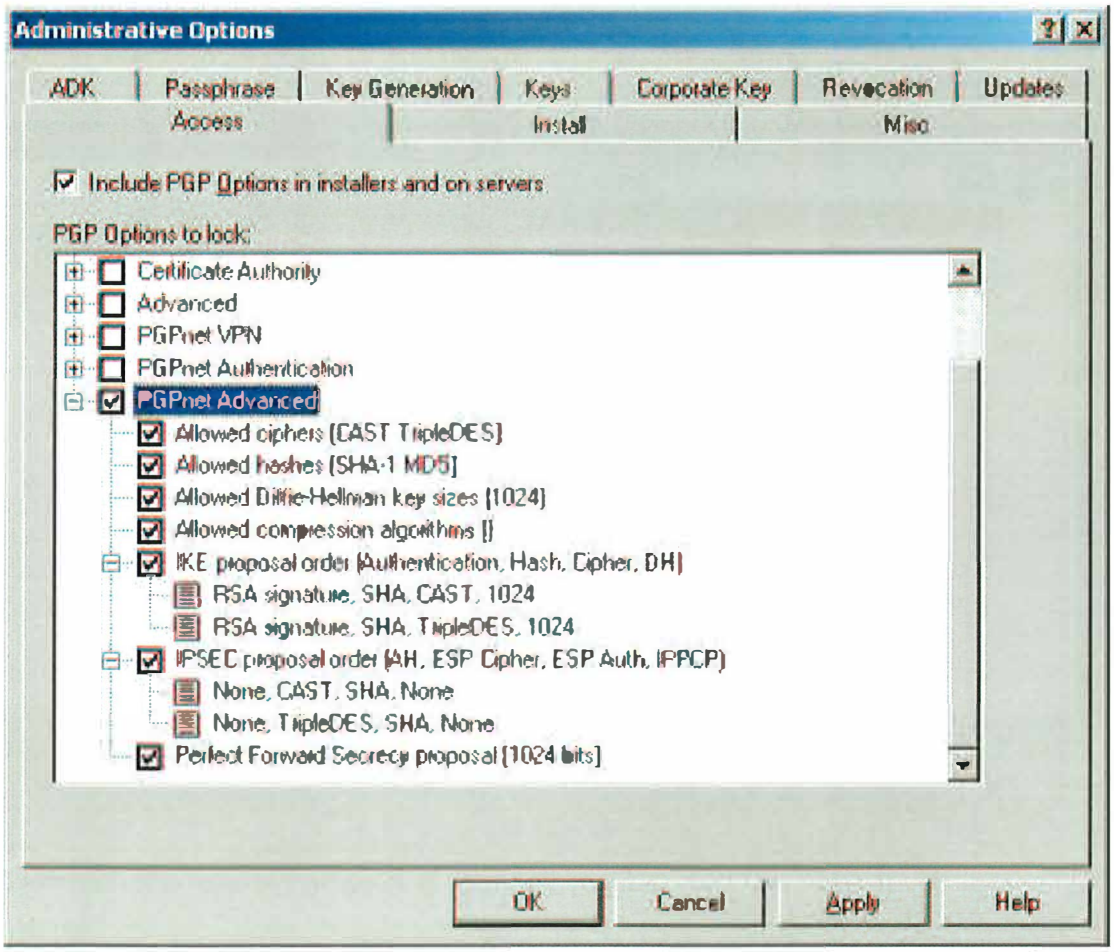

Dans cet écran sont configurées les durées de vie pour les SA IKE et IPsec. Il est impératif que les valeur choisies correspondent à celles spécifiées pour la passerelle. Nous remarquons que pour le client PGPnet il n'est possible que de configurer une seule limite, alors que la passerelle prend une limite souple et une limite rigide. La limite du client doit être comprise dans l'intervalle des deux limites de la passerelle.



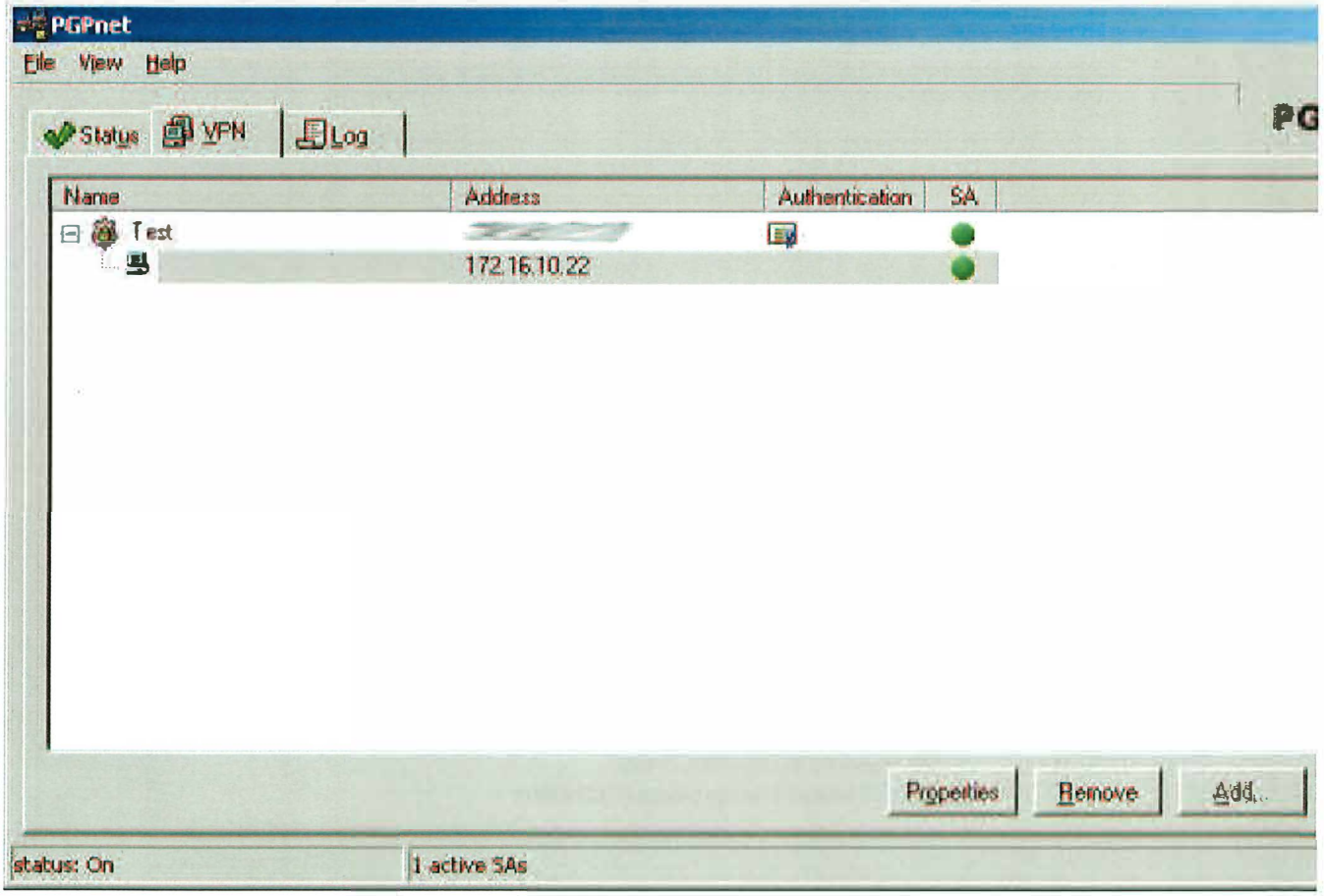

C'est dans cet écran qu'est paramètré le certificat du CA qui aura préalablement été importé et signé. On voit apparaitre les éléments du Distinguished Name du Root Certificate. Les autres informations de cet écran ne sont pas important pour nous, sauf peut-être le type d'autorité de certification. Nous n'avons pas testé avec les autres valeurs.




L'authentification de la passerelle se fera grâce à son certificat qui aura lui aussi été importé et signé au préalable.




Le logiciel PGPadmin permet de créer un exécutable d'installation automatique et de verrouiller les paramètres qui auront été préconfigurés.




C'est à partir de cet écran que l'on configure l'adresse de la passerelle ainsi que celle des hôtes qui seront accédés à travers elle.
Cette capture d'écran a été prise alors qu'une connection IPsec était ouverte. On peut le voir grâce aux boules vertes qui indiquent que des SA ont été établies pour communiquer avec ces hôtes.

# Annexe D

# Script de génération de certificat

```
#!/bin/ksh

if ( ! test $1 || ! test $2 )
then
      echo usage:
      echo        $0 req_filename email
        exit
fi

CA_key="/etc/ssl/private/ca.key"
CA_cert="/etc/ssl/ca.crt"
currentDir=`pwd`
cert_dir=$currentDir/certs

OpenSSL_bin=/usr/sbin/openssl
CertPatch_bin=/usr/sbin/certpatch

echo processing Certificate request $1

#Test that the file exists
if ( ! test -r $1 )
then
      echo the file $1 does not exist!
      echo exiting...
      exit
fi

# First compute the filename of the new certificate
newCertFile=${1%%.csr}
echo $newCertFile.crt

echo Signing...
$OpenSSL_bin x509 -req -days 365 -in $1 -CA $CA_cert -CAkey $CA_key \
-CAcreateserial -out $cert_dir/$newCertFile.crt

#Testing if the cert file was well generated
if ( ! test -r $cert_dir/$newCertFile.crt )
then
      echo The new certificate file could not be found
      echo problem while generating $newCertFile.crt
      echo exiting...
      exit
fi

echo Patching...
$CertPatch_bin -i ufqdn -i $2 -k $CA_key $cert_dir/$newCertFile.crt \
$cert_dir/$newCertFile.crt

echo Copying .pem file
cp $cert_dir/$newCertFile.crt $cert_dir/$newCertFile.pem

echo Done!
```

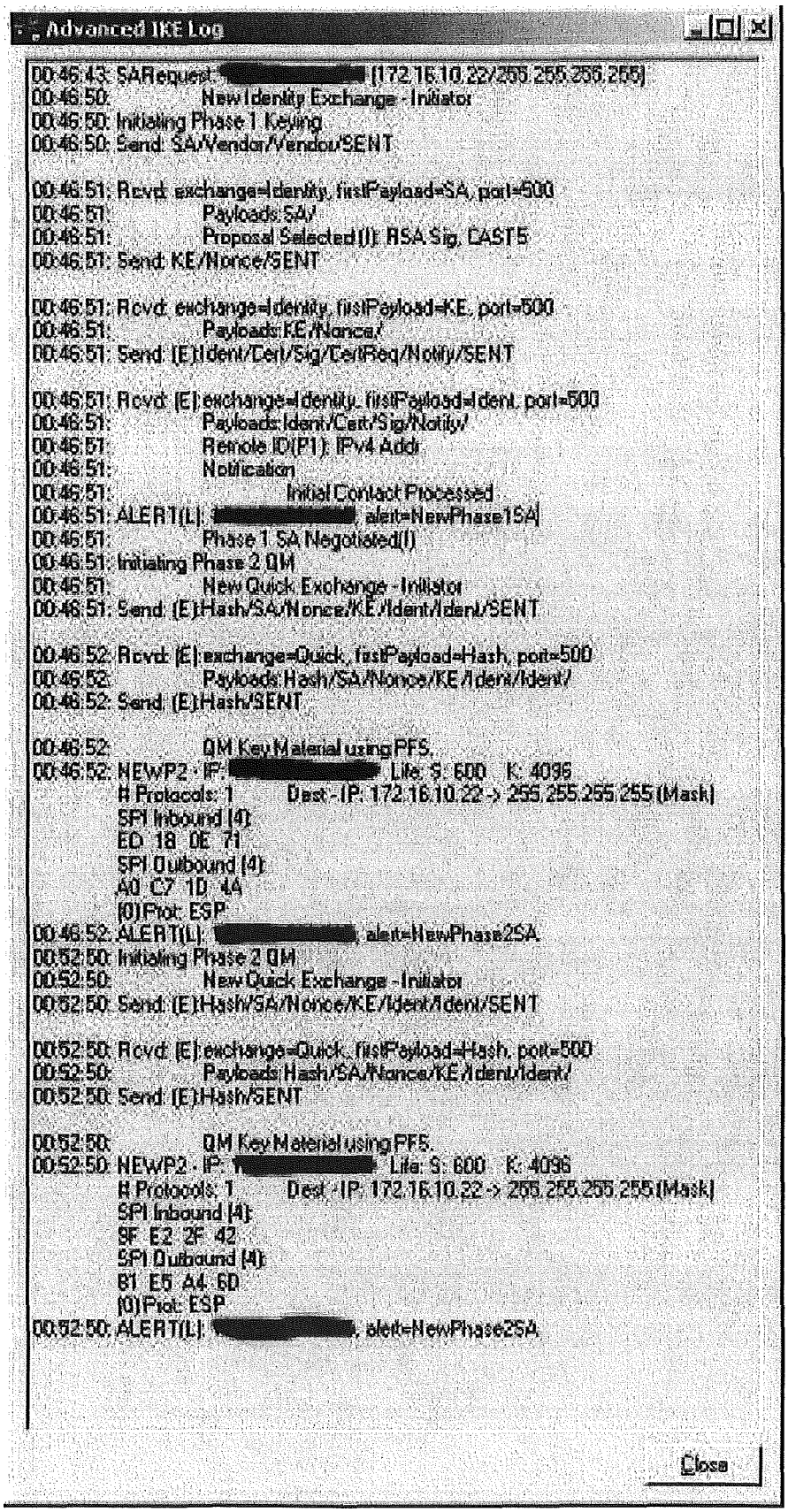
# Annexe E

# Logs d'une connexion IPsec

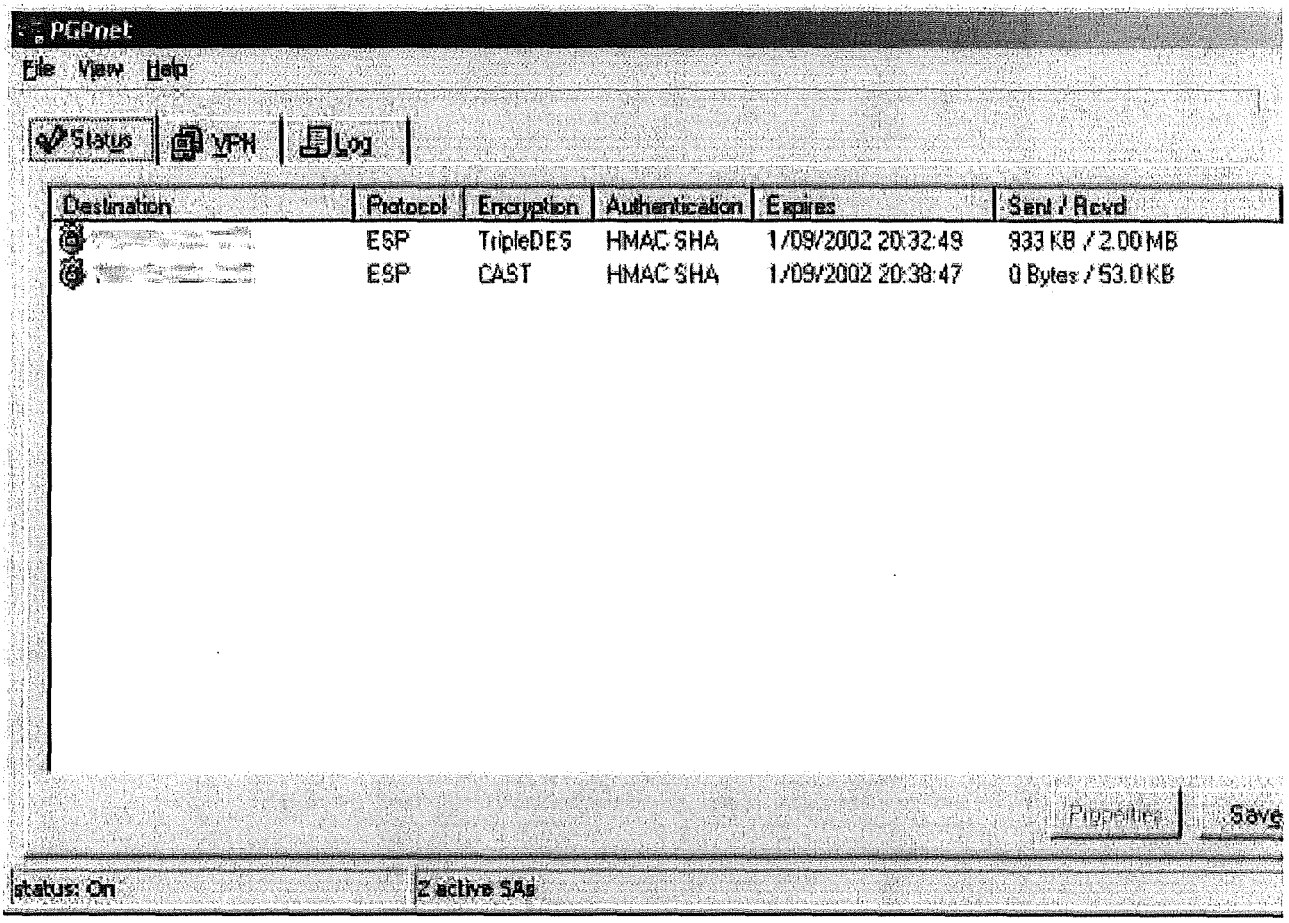
## E.1 Logs sur la machine cliente

## E.2 Logs sur la passerelle

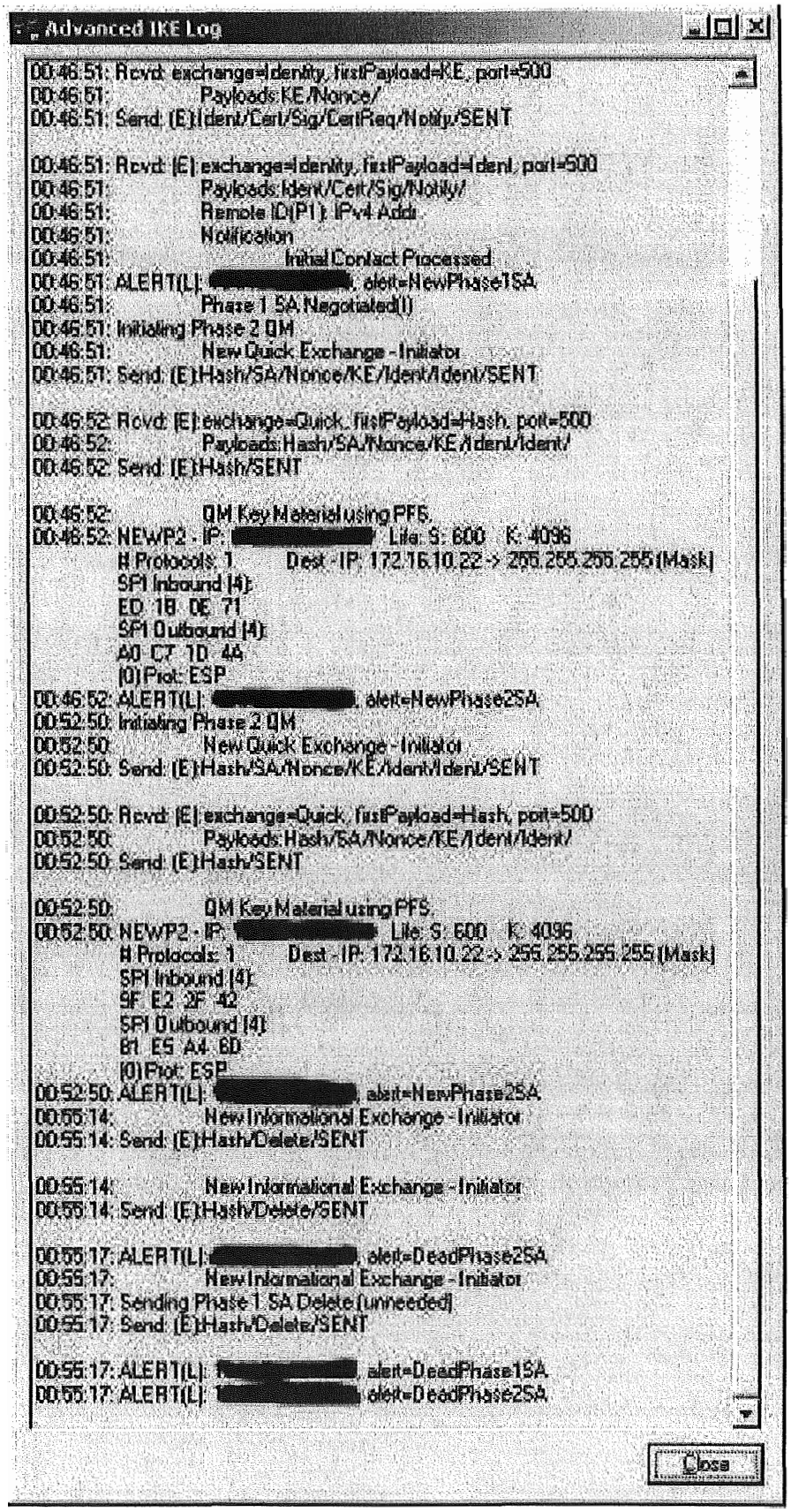
Le log par la passerelle des échanges correspondant à la session de test décrite plus haut se trouve à la page suivante.




Voici la trace produite sur le client pendant l'établissement d'une SA. On y retrouve les messages échangés au cours de la phase 1 dans un premier temps, puis pendant la phase 2 qui permet d'établir la SA IPsec.




L'écran de status nous montre ici l'existence de deux SA IPsec vers la passerelle. Si on regarde la zone indiquant la quantité de données reçue et envoyée on s'aperçoit que nous nous trouvons dans le cas où une limite souple de durrée de vie à été atteinte et une nouvelle SA a été négociée par la passerelle. Et en effet la limite souple de la passerelle est bien de 2MB. Les deux SA peuvent coexister sans problème. Nous voyons sur cet écran que la passerelle utilise la nouvelle SA pour envoyer, mais que le client continue à utiliser l'ancienne car pour lui la limite souple (4MB) n'est pas encore atteinte.




Cette trace est celle générée pendant la négociation d'une nouvelle SA. On voit qu'à l'heure 00 :52 :50 le client à démarrer une nouvelle négociation de phase 2 à l'issue de laquelle une nouvelle paire de SA IPsec a été créée. Cette trace contient également les message de notification échangés lors de la coupure de la connection en fin de session.

```
20:41:10.869745 213.177.154.233.500 > w.x.y.z.500:  [udp sum ok] isakmp v1.0
exchange ID_PROT
      cookie: 406ad6b9f8dea5b0->0000000000000000 msgid: 00000000 len: 148
      payload: SA len: 84 DOI: 1(IPSEC) situation: IDENTITY_ONLY
          payload: PROPOSAL len: 72 proposal: 1 proto: ISAKMP spisz: 0 xforms: 2
              payload: TRANSFORM len: 32
                  transform: 1 ID: ISAKMP
                      attribute ENCRYPTION_ALGORITHM = CAST_CBC
                      attribute HASH_ALGORITHM = SHA
                      attribute AUTHENTICATION_METHOD = RSA_SIG
                      attribute GROUP_DESCRIPTION = MODP_1024
                      attribute LIFE_TYPE = SECONDS
                      attribute LIFE_DURATION = 3600
              payload: TRANSFORM len: 32
                  transform: 2 ID: ISAKMP
                      attribute ENCRYPTION_ALGORITHM = 3DES_CBC
                      attribute HASH_ALGORITHM = SHA
                      attribute AUTHENTICATION_METHOD = RSA_SIG
                      attribute GROUP_DESCRIPTION = MODP_1024
                      attribute LIFE_TYPE = SECONDS
                      attribute LIFE_DURATION = 3600
      payload: VENDOR len: 16
      payload: VENDOR len: 20 [ttl 0] (id 1)
20:41:10.870527 w.x.y.z.500 > 213.177.154.233.500:  [udp sum ok] isakmp v1.0
exchange ID_PROT
      cookie: 406ad6b9f8dea5b0->3598dbb0a34c04aa msgid: 00000000 len: 80
      payload: SA len: 52 DOI: 1(IPSEC) situation: IDENTITY_ONLY
          payload: PROPOSAL len: 40 proposal: 1 proto: ISAKMP spisz: 0 xforms: 1
              payload: TRANSFORM len: 32
                  transform: 1 ID: ISAKMP
                      attribute ENCRYPTION_ALGORITHM = CAST_CBC
                      attribute HASH_ALGORITHM = SHA
                      attribute AUTHENTICATION_METHOD = RSA_SIG
                      attribute GROUP_DESCRIPTION = MODP_1024
                      attribute LIFE_TYPE = SECONDS
                      attribute LIFE_DURATION = 3600 [ttl 0] (id 2)
20:41:10.976351 213.177.154.233.500 > w.x.y.z.500:  [udp sum ok] isakmp v1.0
exchange ID_PROT
      cookie: 406ad6b9f8dea5b0->3598dbb0a34c04aa msgid: 00000000 len: 196
      payload: KEY_EXCH len: 132
      payload: NONCE len: 36 [ttl 0] (id 3)
20:41:11.045715 w.x.y.z.500 > 213.177.154.233.500:  [udp sum ok] isakmp v1.0
exchange ID_PROT
      cookie: 406ad6b9f8dea5b0->3598dbb0a34c04aa msgid: 00000000 len: 196
      payload: KEY_EXCH len: 132
      payload: NONCE len: 36 [ttl 0] (id 4)
20:41:11.349034 213.177.154.233.500 > w.x.y.z.500:  [udp sum ok] isakmp v1.0
exchange ID_PROT
      cookie: 406ad6b9f8dea5b0->3598dbb0a34c04aa msgid: 00000000 len: 1036
      payload: ID len: 155 type: DER_ASN1_DN = "(not shown)"
      payload: CERT len: 697
      payload: SIG len: 132
      payload: CERTREQUEST len: 5
      payload: NOTIFICATION len: 12
          notification: INITIAL CONTACT (0000000000000000->0000000000000000) [ttl
0] (id 5)
20:41:11.383712 w.x.y.z.500 > 213.177.154.233.500:  [udp sum ok] isakmp v1.0
exchange ID_PROT
      cookie: 406ad6b9f8dea5b0->3598dbb0a34c04aa msgid: 00000000 len: 891
      payload: ID len: 12 type: IPV4_ADDR = w.x.y.z
      payload: CERT len: 691
      payload: SIG len: 132
      payload: NOTIFICATION len: 28
          notification: INITIAL CONTACT (406ad6b9f8dea5b0->3598dbb0a34c04aa) [ttl
0] (id 6)
20:41:11.660785 213.177.154.233.500 > w.x.y.z.500:  [udp sum ok] isakmp v1.0
exchange QUICK_MODE
      cookie: 406ad6b9f8dea5b0->3598dbb0a34c04aa msgid: f35dbf48 len: 356
      payload: HASH len: 24
      payload: SA len: 108 DOI: 1(IPSEC) situation: IDENTITY_ONLY
          payload: PROPOSAL len: 48 proposal: 1 proto: IPSEC_ESP spisz: 4 xforms: 1
```

```
SPI: 0xc8ab0078
              payload: TRANSFORM len: 36
                  transform: 1 ID: CAST
                      attribute AUTHENTICATION_ALGORITHM = HMAC_SHA
                      attribute ENCAPSULATION_MODE = TUNNEL
                      attribute GROUP_DESCRIPTION = 2
                      attribute LIFE_TYPE = KILOBYTES
                      attribute LIFE_DURATION = 4096
                      attribute LIFE_TYPE = SECONDS
                      attribute LIFE_DURATION = 600
          payload: PROPOSAL len: 48 proposal: 2 proto: IPSEC_ESP spisz: 4 xforms: 1
SPI: 0x4d5fb920
              payload: TRANSFORM len: 36
                  transform: 1 ID: 3DES
                      attribute AUTHENTICATION_ALGORITHM = HMAC_SHA
                      attribute ENCAPSULATION_MODE = TUNNEL
                      attribute GROUP_DESCRIPTION = 2
                      attribute LIFE_TYPE = KILOBYTES
                      attribute LIFE_DURATION = 4096
                      attribute LIFE_TYPE = SECONDS
                      attribute LIFE_DURATION = 600
      payload: NONCE len: 36
      payload: KEY_EXCH len: 132
      payload: ID len: 12 type: IPV4_ADDR = 192.168.1.1
      payload: ID len: 12 type: IPV4_ADDR = 172.16.10.22 [ttl 0] (id 7)
20:41:11.823398 w.x.y.z.500 > 213.177.154.233.500:  [udp sum ok] isakmp v1.0
exchange QUICK_MODE
      cookie: 406ad6b9f8dea5b0->3598dbb0a34c04aa msgid: f35dbf48 len: 304
      payload: HASH len: 24
      payload: SA len: 60 DOI: 1(IPSEC) situation: IDENTITY_ONLY
          payload: PROPOSAL len: 48 proposal: 2 proto: IPSEC_ESP spisz: 4 xforms: 1
SPI: 0xffb3c9f0
              payload: TRANSFORM len: 36
                  transform: 1 ID: 3DES
                      attribute AUTHENTICATION_ALGORITHM = HMAC_SHA
                      attribute ENCAPSULATION_MODE = TUNNEL
                      attribute GROUP_DESCRIPTION = 2
                      attribute LIFE_TYPE = KILOBYTES
                      attribute LIFE_DURATION = 4096
                      attribute LIFE_TYPE = SECONDS
                      attribute LIFE_DURATION = 600
      payload: NONCE len: 36
      payload: KEY_EXCH len: 132
      payload: ID len: 12 type: IPV4_ADDR = 192.168.1.1
      payload: ID len: 12 type: IPV4_ADDR = 172.16.10.22 [ttl 0] (id 8)
20:41:11.930414 213.177.154.233.500 > w.x.y.z.500:  [udp sum ok] isakmp v1.0
exchange QUICK_MODE
      cookie: 406ad6b9f8dea5b0->3598dbb0a34c04aa msgid: f35dbf48 len: 52
      payload: HASH len: 24 [ttl 0] (id 9)
20:42:38.989541 213.177.154.233.500 > w.x.y.z.500:  [udp sum ok] isakmp v1.0
exchange INFO
      cookie: 406ad6b9f8dea5b0->3598dbb0a34c04aa msgid: d84d38dc len: 68
      payload: HASH len: 24
      payload: DELETE len: 16 [ttl 0] (id 10)
20:42:42.008960 213.177.154.233.500 > w.x.y.z.500:  [udp sum ok] isakmp v1.0
exchange INFO
      cookie: 406ad6b9f8dea5b0->3598dbb0a34c04aa msgid: 395aecc0 len: 84
      payload: HASH len: 24
      payload: DELETE len: 28 [ttl 0] (id 11)
```

# Annexe F

# Page de manuel isakmpd.conf

ISAKMPD.CONF(5) OpenBSD Programmer's Manual ISAKMPD.CONF(5)

## NAME

**isakmpd.conf** - configuration file for isakmpd

## DESCRIPTION

**isakmpd.conf** is the configuration file for the **isakmpd** daemon managing security association and key management for the IPSEC layer of the kernel's networking stack.

The file is of a well known type of format called .INI style, named after the suffix used by an overrated windowing environment for its configuration files. This format consists of sections, each beginning with a line looking like:

[Section name]
Between the brackets is the name of the section following this section header. Inside a section many tag/value pairs can be stored, each one looking like:

Tag=Value
If the value needs more space than fits on a single line it's possible to continue it on the next by ending the first with a backslash character immediately before the newline character. This method can extend a value for an arbitrary amount of lines.

Comments can be put anywhere in the file by using a hash mark (`#'). Then the comment goes on to the end of the line.

Often the right-hand side values consist of other section names. This results in a tree structure. Some values are treated as a list of several scalar values, such lists always use comma as the separator. Some values are formatted like this: X,Y:Z, which is an offer/accept syntax, where X is a value we offer and Y:Z is a range of accepted values, inclusive.

### Auto-generated parts of the configuration

Some predefined section names are recognized by the daemon, voiding the need to fully specify the Main Mode transforms and Quick Mode suites, protocols and transforms.

For Main Mode:
*{DES,BLF,3DES,CAST}-{MD5,SHA}[-{DSS,RSA_SIG}]*

For Quick Mode:
*QM-{ESP,AH}[-TRP]-{DES,3DES,CAST,BLF,AES}[-{MD5,SHA,RIPEMD}][-PFS]-SUITE*

Example 1: 3DES-SHA means; 3DES encryption, SHA hash, and authorization by pre-shared keys. Example 2: QM-ESP-3DES-SHA-PFS-SUITE means; ESP protocol, 3DES encryption, SHA hash, and use Perfect Forward Security.

There are two predefined lifetimes used. The Main Mode lifetime, named LIFE_MAIN_MODE, currently defaults to one hour (minimum 60 seconds, maximum 1 day). The Quick Mode lifetime, LIFE_QUICK_MODE, defaults to 20 minutes (minimum 60 seconds, maximum 1 day).

In addition, the predefinitions include some default values for the special sections "General", and "X509-certificates". These values are presented in the example below.

Note that using the predefined section names imply some minor limitations. There are currently no predefined ESP+AH Quick Mode suites, and the Diffie-Hellman group description for Main Mode (and Quick Mode with PFS) is set to DH group 1 (MODP_768) when using MD5 hash, and DH group 2 (MODP_1024) when using SHA hash.

All autogenerated values can be overridden by manual entries by using the same section and tag names in the configuration file.

In particular, the default phase 1 (Main or Aggressive Mode) and phase 2 (Quick Mode) lifetimes can be overridden by these tags under the "General" section;

```
[General]
Default-phase-1-lifetime=       3600,60:86400
Default-phase-2-lifetime=       1200,60:86400
```

Also, the default Phase 1 ID can be set by creating a <Phase1-ID> section, as shown below, and adding this tag under the "General" section;

```
[General]
Default-phase-1-ID=             Phase1-ID-name

[Phase1-ID-name]
ID-type=                        USER_FQDN
Name=                           foo@bar.com
```

**Roots**

*General* Generic global configuration parameters

*Policy-file* The name of the file that contains KeyNote(4) policies. The default is "/etc/isakmpd/isakmpd.policy".

*Retransmits* How many times should a message be retransmitted before giving up.

*Check-interval* The interval between watchdog checks of connections we want up at all times.

*Exchange-max-time* How many seconds should an exchange maximally take to setup before we give up.

*Listen-on* A list of IP-addresses OK to listen on. This list is used as a filter for the set of addresses the interfaces configured provides. This means that we won't see if an address given here does not exist on this host, and thus no error is given for that case.

*Shared-SADB* If this tag is defined, whatever the value is, some semantics of **isakmpd.conf** are changed so that multiple instances can run on top of one SADB and setup SAs with eachother. Specifically this means replay protection will not be asked for, and errors that can occur when updating an SA with its parameters a 2nd time will be ignored.

*Phase 1* ISAKMP SA negotiation parameter root

*IP-address* A name of the ISAKMP peer at the given IP-address.

*Default* A name of the default ISAKMP peer. Incoming Phase 1 connections from other IP-addresses will use this peer name.

This name is used as the section name for further information to be found. Look at <ISAKMP-peer> below.

*Phase 2* IPsec SA negotiation parameter root

*Connections* A list of directed IPSec "connection" names that should be brought up automatically, either on first use if the system supports it, or at startup of the daemon. These names are section names where further information can be found. Look at <IPSec-connection> below. Normally any connection mentioned here are treated as part of the "Passive-connection" list we present below, however there is a flag: "Active-only" that disables this be-

haviour. This too is mentioned in the <IPSec-connection> section, in the "Flags" tag.

*Passive-connections*
A list of IPSec "connection" names we recognize and accept initiations for. These names are section names where further information can be found. Look at <IPSec-connection> below. Currently only the Local-ID and Remote-ID tags are looked at in those sections, as they are matched against the IDs given by the initiator.

*KeyNote*

*Credential-directory*
A directory containing directories named after IDs (IP addresses, ``user@domain'', or hostnames) that contain files named ``credentials'' and ``private_key''.

The credentials file contains keynote(4) credentials that are sent to a remote IKE daemon when we use the associated ID, or credentials that we may want to consider when doing an exchange with a remote IKE daemon that uses that ID. Note that, in the former case, the last credential in the file MUST contain our public key in its Licensees field. More than one credentials may exist in the file. They are separated by whitelines (the format is essentially the same as that of the policy file). The credentials are of the same format as the policies described in isakmpd.policy(5). The only difference is that the Authorizer field contains a public key, and the assertion is signed. Signed assertions can be generated using the keynote(1) utility.

The private_key file contains the private RSA key we use for authentication. If the directory (and the files) exist, they take precedence over X509-based authentication.

*X509-Certificates*

*Ca-directory* A directory containing PEM certificates of certification authorities that we trust to sign other certificates. Note that for a CA to be really trusted, it needs to be somehow referred to by policy, in isakmpd.policy(5). The certificates in this directory are used for the actual X.509 authentication and for cross-referencing policies that refer to Distinguished Names (DNs). Keeping a separate directory (as opposed to integrating policies and X.509 CA certificates) allows for maintenance of a list of "well known" CAs without actually having to trust all (or any) of them.

*Cert-directory*
A directory containing PEM certificates that we trust to be valid. These certificates are used in preference to those passed in messages and are required to have a SubjectAltName extension.

*Accept-self-signed*
If this tag is defined, whatever the value is, certificates that do not originate from a trusted CA but are self-signed will be accepted.

*Private-key* The private key matching the public key of our certificate (which should be in the "Cert-directory", and have a subjectAltName matching our ID, so far that is our IP-address).

**Referred-to sections**

*_ISAKMP-peer_* Parameters for negotiation with an ISAKMP peer

*Phase* The constant 1, as ISAKMP-peers and IPSec-connections really are handled by the same code inside isakmpd.

*Transport* The name of the transport protocol, defaults to UDP.

*Port* In case of UDP, the UDP port number to send to. This is optional, the default value is 500 which is the IANA-registered number for ISAKMP.

*Local-address* The Local IP-address to use, if we are multi-homed, or have aliases.

*Address* If existent, the IP-address of the peer.

*Configuration* The name of the ISAKMP-configuration section to use. Look at <ISAKMP-configuration> below.

*Authentication* If existent, authentication data for this specific peer. In the case of preshared key, this is the key value itself.

*ID* If existent, the name of the section that describes the local client ID that we should present to our peer. If not present, it defaults to the address of the local interface we are sending packets over to the remote daemon. Look at <Phase1-ID> below.

*Remote-ID* If existent, the name of the section that describes the remote client ID we expect the remote daemon to send us. If not present, it defaults to the address of the remote daemon. Look at <Phase1-ID> below.

*Flags* A comma-separated list of flags controlling the further handling of the ISAKMP SA. Currently there are no specific ISAKMP SA flags defined.

*_Phase1-ID_*

*ID-type* The ID type as given by the RFCs. For Phase 1 this is currently IPV4_ADDR, IPV4_ADDR_SUBNET, FQDN, USER_FQDN, or KEY_ID.

*Address* If the ID-type is IPV4_ADDR, this tag should exist and be an IP-address.

*Network* If the ID-type is IPV4_ADDR_SUBNET this tag should exist and be a network address.

*Netmask* If the ID-type is IPV4_ADDR_SUBNET this tag should exist and be a network subnet mask.

*Name* If the ID-type is FQDN, USER_FQDN, or KEY_ID, this tag should exist and contain a domain name, user@domain, or other identifying string respectively.

*_ISAKMP-configuration_*

*DOI* The domain of interpretation as given by the RFCs. Normally IPSEC. If unspecified, defaults to IPSEC.

*EXCHANGE_TYPE*
The exchange type as given by the RFCs. For main mode this is ID_PROT and for aggressive mode it is AGGRESSIVE.

*Transforms* A list of proposed transforms to use for protecting the ISAKMP traffic. These are actually names for sections further describing the transforms. Look at <ISAKMP-transform> below.

*_ISAKMP-transform_*

*ENCRYPTION_ALGORITHM*
The encryption algorithm as the RFCs name it, or ANY to denote that any encryption algorithm proposed will be accepted.

*KEY_LENGTH* For encryption algorithms with variable key length, this is where the offered/accepted keylengths are described. The value is of the offer-accept kind described above.

*HASH_ALGORITHM*
The hash algorithm as the RFCs name it, or ANY.

*AUTHENTICATION_METHOD*
The authentication method as the RFCs name it, or ANY.

*GROUP_DESCRIPTION*
The group used for Diffie-Hellman exponentiations, or ANY. The name are symbolic, like MODP_768, MODP_1024, EC_155 and EC_185.

*PRF* The algorithm to use for the keyed pseudo-random function (used for key derivation and authentication in Phase 1), or ANY.

*Life* A list of lifetime descriptions, or ANY. In the former case, each element is in itself a name of the section that defines the lifetime. Look at <Lifetime> below. If it is set to ANY, then any type of proposed lifetime type and value will be accepted.

*_Lifetime_*

*LIFE_TYPE* SECONDS or KILOBYTES depending on the type of the duration. Notice that this field may NOT be set to ANY.

*LIFE_DURATION*
An offer/accept kind of value, see above. Can also be set to ANY.

*_IPSec-connection_*

*Phase* The constant 2, as ISAKMP-peers and IPSec-connections really are handled by the same code inside isakmpd.

*ISAKMP-peer* The name of the ISAKMP-peer which to talk to in order to set up this connection. The value is the name of an <ISAKMP-peer> section. See above.

*Configuration*
The name of the IPSec-configuration section to use. Look at <IPSec-configuration> below.

*Local-ID* If existent, the name of the section that describes the optional local client ID that we should present to our peer. It is also used when we act as responders to find out what <IPSec-connection> we are dealing with. Look at <IPSec-ID> below.

*Remote-ID* If existent, the name of the section that describes the optional remote client ID that we should present to our peer. It is also used when we act as responders to find out what <IPSec-connection> we are dealing with. Look at <IPSec-ID> below.

*Flags* A comma-separated list of flags controlling the further handling of the IPSec SA. Currently only one flag is defined:

*Active-only* If this flag is given and this <IPSec-connection> is part of the phase 2 connections we automatically keep up, it will not automatically be used for accepting connections from the peer.

*_IPSec-configuration_*

*DOI* The domain of interpretation as given by the RFCs. Normally IPSEC. If unspecified, defaults to IPSEC.

*EXCHANGE_TYPE*
The exchange type as given by the RFCs. For quick mode this is QUICK_MODE.

*Suites* A list of protection suites (bundles of protocols) usable for protecting the IP traffic. Each of the list elements is a name of an <IPSec-suite> section. See below.

*_IPSec-suite_*

*Protocols* A list of the protocols included in this protection suite. Each of the list elements is a name of an <IPSec-protocol> section. See below.

*_IPSec-protocol_*

*PROTOCOL_ID* The protocol as given by the RFCs. Acceptable values today are IPSEC_AH and IPSEC_ESP.

*Transforms* A list of transforms usable for implementing the protocol. Each of the list elements is a name of an <IPSec-transform> section. See below.

*ReplayWindow* The size of the window used for replay protection. This is normally left alone. Look at the **ESP** and **AH** RFCs for a better description.

*_IPSec-transform_*

*TRANSFORM_ID* The transform ID as given by the RFCs.

*ENCAPSULATION_MODE*
The encapsulation mode as given by the RFCs. This means TRANSPORT or TUNNEL.

*AUTHENTICATION_ALGORITHM*
The optional authentication algorithm in the case of this being an ESP transform.

*GROUP_DESCRIPTION*
An optional (provides PFS if present) Diffie-Hellman group description. The values are the same as GROUP_DESCRIPTION's in <ISAKMP-transform> sections shown above.

*Life* List of lifetimes, each element is a <Life-time> section name.

*_IPSec-ID_*

*ID-type* The ID type as given by the RFCs. For IPSec this is currently IPV4_ADDR or IPV4_ADDR_SUBNET.

*Address* If the ID-type is IPV4_ADDR, this tag should exist and be an IP-address.

*Network* If the ID-type is IPV4_ADDR_SUBNET this tag should exist and be a network address.

*Netmask* If the ID-type is IPV4_ADDR_SUBNET this tag should exist and be a network subnet mask.

*Protocol* If the ID-type is IPV4_ADDR or IPV4_ADDR_SUBNET, this tag indicates what transport protocol should be transmitted over the SA. If left unspecified, all transport protocols between the two address (ranges) will be sent (or permitted) over that SA.

*Port* If the ID-type is IPV4_ADDR or IPV4_ADDR_SUBNET, this tag indicates what source or destination port is allowed to be transported over the SA (depending on whether this is a local or remote ID). If left un-specified, all ports of the given transport protocol will be transmitted (or permitted) over the SA. The Protocol tag must be speci-fied in conjunction with this tag.