Auto-configuration des VPNs - IPSec

Adil El Kaysouni

Dédicace

A mon cher père, pour ses sacrifices et ses conseils.

A ma chère mère pour son amour, son affection, et ses prières.

A mon cher frère, symbole de fraternité.

A ma chère sœur pour son soutien et encouragement.

A tous mes amis et mes professeurs.

El Kaysouni Adil

Avant-propos

Présentation générale

Ce stage de troisième année, d'une durée de quatre mois au sein du laboratoire LORIA, a pour objectif de mettre en œuvre les compétences et les bases techniques, scientifiques et humaines acquises durant les trois années de formation d'ingénieur à l'ENSIAS. Etant également réalisé dans un laboratoire de recherche, il constitue aussi une initiation au domaine de la recherche scientifique.

Remerciement

Au terme de ce stage, je tiens à adresser mes remerciements chaleureux à mes encadrants au Loria, M. Olivier Festor et M. Laurent Ciarletta pour leur grande disponibilité. Ils n'ont cessé de me prodiguer leurs conseils et leurs remarques, qui me seront fort utiles pour l'avenir.

Mes vifs remerciements vont à M. Laurent Ciarletta, pour l'aide et le temps qu'il a consacré à la mise en ordre des différents éléments et idées contenus dans ce rapport et pour ses directives qui m'ont été d'un appui considérable tout au long de ma démarche.

Enfin, je remercie tous les membres de l'équipe MADYNES qui m'ont appuyé, de près ou de loin, tout au long de ce projet, je leur fais part de toute mon amitié et de ma gratitude.

SOMMAIRE

LISTE DES FIGURES

Introduction

Les services Web et la technologie XML (eXtended Markup Language) s'imposent aujourd'hui dans de nombreux domaines de services de l'Internet comme le commerce électronique, le multimédia ou de façon générale, dans l'échange d'informations entre applications. La gestion de configuration d'éléments réseaux n'échappe pas à cette évolution.

C'est dans ce cadre que s'inscrit notre projet. Ce dernier a pour but de réaliser une plate-forme de gestion dynamique de configuration VPN (Virtual Private Networks) IPsec(Internet Protocol Security protocol). Elle se base sur les Services Web et plus particulièrement sur Active XML, un modèle pair à pair de Service Web développé dans le cadre du projet GEMO(Intégration de données et de connaissances distribuées sur le Web)à l'INRIA Rocquencourt.

Dans un souci de clarté, nous avons décomposé ce rapport en cinq chapitres qui seront détaillés comme suit :

Un premier chapitre présentera le projet avec les objectifs attendus, ainsi qu'une brève présentation du lieu de travail.

Un second chapitre présentera le protocole IPSec, nous allons décrire le fonctionnement de ce dernier sur différents systèmes d'exploitation.

Un troisième chapitre dans lequel nous présenterons le framework Active XML et les Services Web.

Un quatrième chapitre qui décrira l'architecture logicielle proposée de notre application, nous allons donc présenter les différents modules qui composent cette application ainsi que les fonctions logicielles de chaque module.

Un cinquième chapitre dans lequel nous présenterons la conception pour la solution proposée, donc nous exposerons les modèles conceptuels.

Un dernier chapitre qui donnera une vue globale de la réalisation de l'application. Il comportera une étude bibliographique sur les outils de travail, une description de l'enchaînement des différentes fenêtres, ainsi que quelques interfaces de l'application.

Chapitre 1 : Présentation du projet

Ce chapitre présente d'une manière détaillée le contexte général de notre projet, il établit ainsi une brève introduction sur l'établissement d'accueil.

I Présentation de LORIA

1.1 Historique :

Le Laboratoire Lorrain de Recherche en Informatique et ses Applications (LORIA) est une unité mixte de recherche (UMR 7503) commune au CNRS (Centre National de la Recherche Scientifique), à l'INRIA (Institut National de Recherche en Informatique et en Automatique), à l'INPL (Institut National Polytechnique de Lorraine), et aux universités Henri Poincaré Nancy 1 et Nancy 2.
Cette unité, dont la création a été officialisée le 19 décembre 1997 par la signature du contrat quadriennal avec le Ministère de l'Éducation Nationale, de la Recherche et de la Technologie et par une convention entre les cinq partenaires, succède ainsi au Centre de Recherche en Informatique de Nancy (CRIN), et aux équipes communes entre celui-ci et l'Unité de Recherche INRIA de Lorraine.

1.2 Organisation :

Le LORIA est composé de plus de 300 personnes réparties en 25 équipes de recherche et 7 services d'aide à la recherche. Chaque équipe rassemble des chercheurs, des doctorants et des assistants techniques ou administratifs, pour la réalisation d'un projet de recherche.

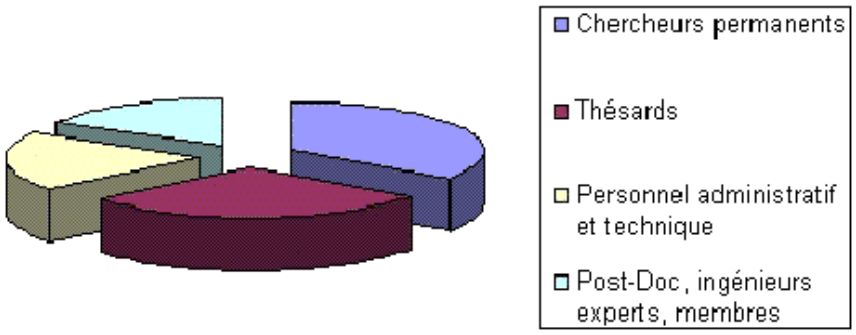


Figure 1.2 : Répartition du personnel du LORIA

Depuis le 1er janvier 2001, Hélène Kirchner dirige le LORIA. Elle est assistée par 5 instances garantissant la cohérence de la politique scientifique et le bon fonctionnement au quotidien :

- L'Équipe de Direction : composée de plusieurs membres du laboratoire, elle assiste la directrice dans ses fonctions.
- Le Comité de Gestion : composé des chefs de service, il gère le fonctionnement journalier du LORIA.
- Le Comité des Projets : il conseille la Directrice sur la politique scientifique du LORIA, participe à l'évaluation des projets/équipes, et instruit les restructurations nécessaires.
- Le Conseil du LORIA : il émet des avis sur la politique scientifique mise en oeuvre par le Comité des Projets. Sa composition est fixée par les statuts d'UMR.
- Le Conseil des Orientations Scientifiques : composé de représentants des équipes de recherche, il conseille la Direction dans la gestion scientifique du laboratoire.

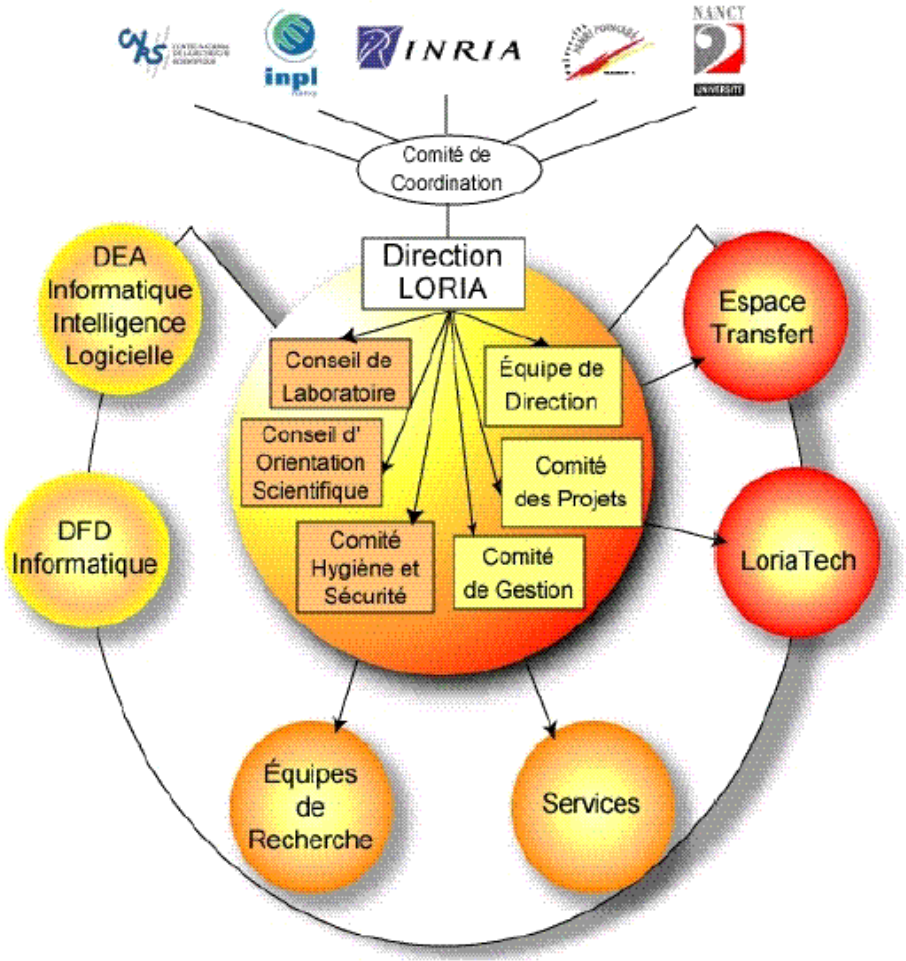


Figure 1.3 Organigramme du laboratoire

1.3 Thématiques :

Les activités de chacune des 25 équipes de recherche du LORIA sont concentrées sur une ou

plusieurs des cinq thématiques du laboratoire :

- Calculs, réseaux et graphismes à hautes performances,
- Télé-opérations et assistants intelligents,
- Ingénierie des langues, du document et de l'information scientifique et technique,
- Qualité et sûreté des logiciels et systèmes informatiques,
- Bioinformatique et applications à la génomique.
-

II Présentation de l'équipe MADYNES

L'équipe de recherche MADYNES (MAnagement of DYnamic NEtworks and Services), dans laquelle j'ai effectué mon stage, se place dans la thématique Calculs, réseaux et graphismes à hautes performances du laboratoire et travaille sur la supervision des réseaux et services dynamiques.
Elle vise la conception, la validation et la mise en oeuvre de nouvelles architectures de supervision et de contrôle capables de maîtriser la dynamicité croissante des services et résistantes au facteur d'échelle induit par l'Internet.

2.1 Composition de l'équipe :

L'équipe MADYNES est composée de 14 personnes (dont 8 permanents) :

Responsable scientifique :
Olivier FESTOR
Responsable permanent :
Isabelle CHRISMENT
Assistante de projet :
Josiane REFFORT
Personnel INRIA :
Radu STATE
Personnel université :
Laurent ANDREY, Jacques GUYARD, Emmanuel NATAF, André SCHAFF, Laurent Ciarletta
Personnel contractuel :
Isabelle ASTIC, Abdelkader LAHMADI
Chercheurs doctorants :
Mouna Benaissa, Nizar BEN YOUSSEF, Guillaume DOYEN, Hassen SALLAY

2.2 Principaux axes de recherche :

Les travaux de l'équipe sont organisés suivant deux axes orthogonaux. Le premier axe développe des travaux sur l'évolution des modèles génériques de supervision afin de définir l'infrastructure de base

pour la gestion autonome. Le second axe contribue à étendre ces fondements au travers des aires fonctionnelles de la supervision. Les aires privilégiées sont celles de la sécurité, la configuration de services (de la souscription à l'activation en passant par le déploiement) et le respect des contraintes de performances (paramétrage, monitorage et mesures de qualité). Le schéma de la figure 2.1 montre les axes de recherche :

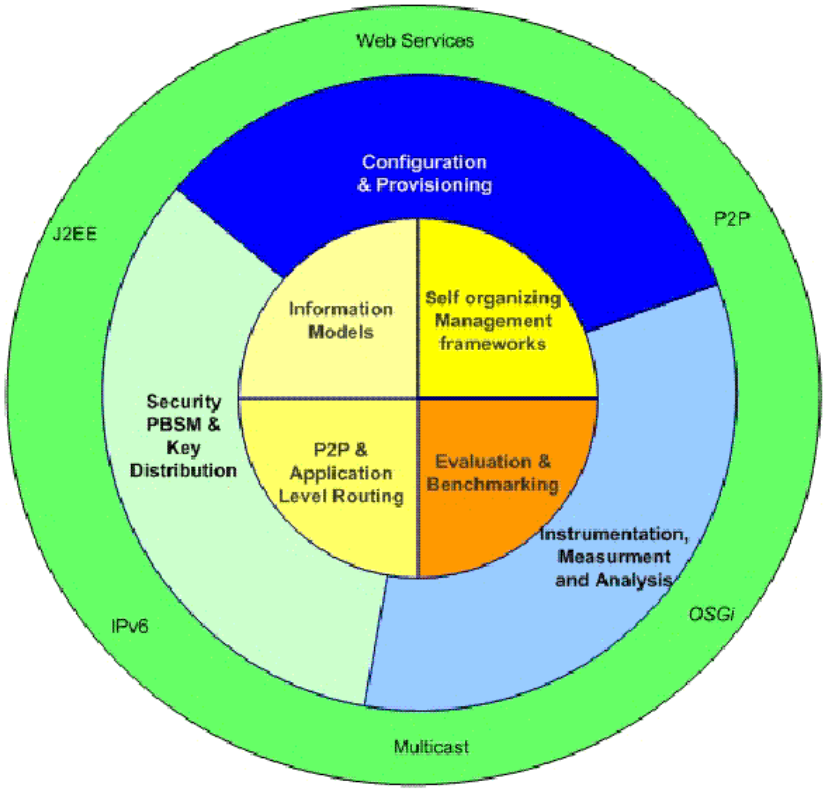


Figure 2.1 : Axes de recherche de l'équipe MADYNES

III Définition de projet

Des travaux préliminaires ont été menés par Mr Pierre Hambert [4] durant l'année 2002/2003. Il s'agissait d'étudier le modèle des services Web dans le cadre de la configuration d'éléments d'un réseau, Cela a donné lieu à la mise en œuvre d'un prototype fonctionnel qui harmonise les règles de filtrages entre trois firewall.

Ce projet constitue la suite de cette première étude, par l'extension des fonctionnalités à la configuration dynamique de VPN (Virtual Private Networks) IPsec(Internet Protocol Security protocol).

Chapitre 2 : IPsec (Internet Protocol Security protocol).

1. Introduction :

La sécurisation des communications dans les réseaux informatiques n'est plus une option, c'est devenue une nécessité. Plusieurs approches peuvent être utilisées pour sécuriser les échanges dans un réseau TCP/IP :

- niveau applicatif (PGP),
- niveau transport (protocoles TLS/SSL, SSH),
- niveau physique (boîtiers chiffrant).

Ainsi dans cette perspective, IPsec est une extension de sécurité pour le protocole Internet IP. Il peut être mis en œuvre sur tous les équipements du réseau, et de nombreux fournisseurs l'intègrent désormais dans leurs produits. Les exemples les plus pertinents d'utilisation d'IPsec sont les réseaux privés virtuels (VPN), mais aussi la sécurisation des accès distants à un Intra-net.

Le réseau IPv4 étant largement déployé, et la migration complète vers IPv6 nécessitant encore beaucoup de temps, il est vite apparu intéressant de définir des mécanismes de sécurité qui soient communs à la fois à IPv4 et IPv6. Ces mécanismes sont couramment désignés par le terme IPsec pour IP Security Protocols. IPsec vise ainsi la sécurisation des échanges au niveau de la couche réseau.

IPsec se présente sous la forme d'une norme, développée par un groupe de travail au sein de l'IETF(*Internet Engineering Task Force*) depuis 1992. Une première version basique de cette extension d'IP est apparue, sous forme de RFC (*Request For Comment*), en 1995. Une seconde version, comportant en plus un système de gestion dynamique des paramètres de sécurité, a été publiée en novembre 1998. La maturité grandissante de la norme conduit désormais de nombreux fournisseurs à intégrer IP sec dans leurs produits, et l'on peut considérer que le marché commence à prendre de l'importance et sera bientôt un secteur incontournable de la sécurité réseau.

2. Principe de fonctionnement :

IPsec fonctionne autour des bases de données des politiques de sécurité. Le fonctionnement de ces bases est décrit dans ce qui suit.

SA (Security Association) :

L'association de sécurité IPsec est une connexion qui fournit des services de sécurité au trafic qu'elle transporte. Il s'agit d'une structure de données servant à stocker l'ensemble des paramètres associés à une communication donnée. Une SA est unidirectionnelle. Ainsi la protection bidirectionnelle d'une communication classique requiert deux associations, une dans chaque sens. Les services de sécurité sont fournis par l'utilisation soit de AH soit de ESP (paragraphe 4). Le rôle d'une SA est donc de consigner, pour chaque adresse IP avec laquelle l'implémentation IPsec peut communiquer, les informations suivantes :

- Index de la SA : appelé SPI (pour Security Parameter Index) choisi par le récepteur,
- Un numéro de séquence : indicateur utilisé pour le service d'anti-rejeu,
- Une fenêtre d'anti-rejeu : compteur 32 bits,
- Le dépassement de séquence,
- Paramètres d'authentification et paramètres de chiffrement : algorithmes et clés,
- La durée de vie (TTL) de la SA,
- Le mode du protocole IPsec : tunnel ou transport.

Chaque association est identifiée de manière unique à l'aide d'un triplet composé de:

- l'adresse de destination des paquets,
- l'identifiant du protocole de sécurité (AH ou ESP),
- l'index de la SA (SPI).

SAD (Security Association Database) :

Pour gérer les associations de sécurité actives, on utilise une base de données des associations de sécurité (Security Associations). Cette base de données contient tous les paramètres relatifs à chaque SA et sera consultée pour savoir comment traiter les paquets reçus ou les paquets à émettre.

SPD (Security Policy Database):

Les protections offertes par IPsec sont basées sur des choix définis dans une base de données de politique de sécurité (Security Policy). Cette base de données est établie et maintenue par un utilisateur, un administrateur ou une application mise en place par ceux-ci. Elle permet de décider, pour chaque paquet, s'il se verra apporter des services de sécurité, s'il sera autorisé à passer ou s'il sera rejeté. Ces services sont basés sur des mécanismes cryptographiques. Pour cela, IPsec fait appel à deux protocoles de sécurité qui viennent s'ajouter au protocole IP classique : il s'agit des protocoles AH et ESP. IPsec offre ainsi deux possibilités d'encapsulation distinctes. Toutefois, l'évolution de ce protocole fait que ESP assure désormais l'ensemble des fonctionnalités des deux mécanismes.

Au-delà de AH et ESP, l'IETF a jugé judicieux d'offrir un service supplémentaire appelé chiffrement en mode Fast Forward qui conserve la même taille de datagrammes et garde ainsi des performances optimales. Cependant, il protège en confidentialité uniquement. L'en-tête IP et la longueur du datagramme restent inchangés sauf éventuellement le champ d'options IP qui lui peut être chiffré.

Les SA contiennent tous les paramètres nécessaires à IPsec, notamment les clés utilisées. La gestion des clés pour IPsec n'est liée aux autres mécanismes de sécurité de IPsec que par le biais des SA. Une SA peut être configurée manuellement dans le cas d'une situation simple, mais la règle générale consiste à utiliser un protocole spécifique qui permet la négociation dynamique des SA et notamment l'échange des clés de session. Le protocole de négociation des SA, développé pour IPsec, est le protocole de gestion des clés et des associations de sécurité pour Internet (Internet Security Association and Key Management Protocol, ISAKMP). ISAKMP est en fait inutilisable seul (il s'agit d'un cadre générique qui permet l'utilisation de plusieurs protocoles d'échange de clé). Dans le cadre de la standardisation de IPsec, ISAKMP est associé à une parti des protocoles SKEME et Oakley pour donner un protocole final du nom de Internet Key Exchange, IKE.

Le schéma ci-dessous montre le principe du fonctionnement SA / SAD /SPD, notamment la création d'une SA basée sur l'échange dynamique des clés à l'aide du protocole IKE :

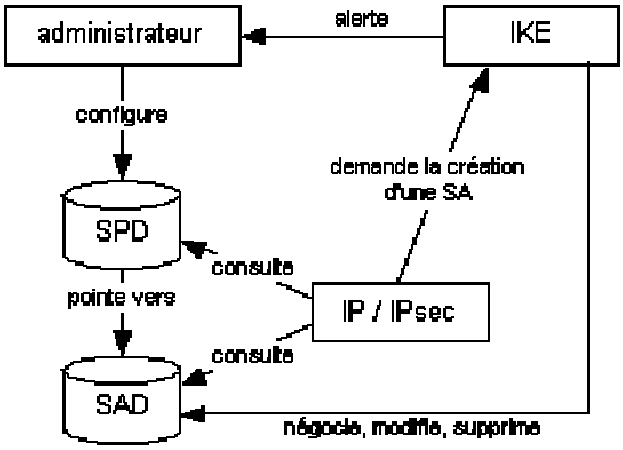


Figure 2.2.1 : Principe du fonctionnement SA / SAD /SPD

Mais pour bien illustrer le fonctionnement d'IPsec, reprenons le schéma précédent sur deux exemples.

Exemple 1 : trafic sortant

Lorsque la couche IPsec reçoit des données à envoyer, elle commence par consulter la base de données des politiques de sécurité (SPD) pour savoir comment traiter ces données. Si cette base lui indique que le trafic doit appliquer des mécanismes de sécurité, elle récupère les caractéristiques requises pour la SA correspondante et va consulter la base des SA (SAD). Si la SA nécessaire existe déjà, elle est utilisée pour traiter le trafic en question. Dans le cas contraire, IPsec fait appel à IKE pour établir une nouvelle SA avec les caractéristiques requises.

Exemple 2 : trafic entrant

Lorsque la couche IPsec reçoit un paquet en provenance du réseau, elle examine l'en-tête pour savoir si ce paquet s'est vu appliquer un ou plusieurs services IPsec et si oui, quelles sont les références de la SA. Elle consulte alors la SAD pour connaître les paramètres à utiliser pour la vérification et/ou le déchiffrement du paquet. Une fois le paquet vérifié et/ou déchiffré, la SPD est consultée pour savoir si l'association de sécurité appliquée au paquet correspondait bien à celle requise par les politiques de sécurité. Dans le cas où le paquet reçu est un paquet IP classique, la SPD permet de savoir s'il a néanmoins le droit de passer. Par exemple, les paquets IKE sont une exception. Ils sont traités par IKE, qui peut envoyer des alertes administratives en cas de tentative de connexion infructueuses.

3. Services de sécurité offerts par IPsec :

IPsec vise à prévenir les diverses attaques possibles, notamment empêcher un adversaire d'espionner les données circulant sur le réseau ou de se faire passer pour autrui afin d'accéder à des ressources ou données protégées.
Dans ce but, IPsec peut fournir, suivant les options sélectionnées, totalement ou partiellement les services de sécurité suivants :

- **Confidentialité** des données et protection partielle contre l'analyse du trafic :

  Les données transportées ne peuvent être lues par un adversaire espionnant les communications. En particulier, aucun mot de passe, aucune information confidentielle ne circule en clair sur le réseau. Il est même possible, dans certains cas, de chiffrer les en-têtes des paquets IP et ainsi masquer, par exemple, les adresses source et destination réelles. Cela permet la protection contre l'analyse du trafic.

- **Authenticité** des données et **contrôle d'accès continu** :

  L'authenticité est composée de deux services, généralement fournis conjointement par un même mécanisme : l'authentification de l'origine des données et l'intégrité. L'authentification de l'origine des données garantit que les données reçues proviennent de l'expéditeur déclaré. L'intégrité garantit que les données n'ont pas été modifiées durant leur transfert. La garantie de l'authenticité de chaque paquet reçu permet de mettre en œuvre un contrôle d'accès fort tout au long d'une communication, contrairement à un contrôle d'accès simple à l'ouverture de la connexion, qui n'empêche pas un adversaire de récupérer une communication à son compte. Ce service permet en particulier de protéger l'accès à des ressources ou données privées.

- **Protection contre le rejeu :**

  La protection contre le rejeu permet de détecter une tentative d'attaque consistant à envoyer de nouveau un paquet valide intercepté précédemment sur le réseau.

Ces services sont basés sur des mécanismes cryptographiques modernes qui leur confèrent un niveau de sécurité élevé lorsqu'ils sont utilisés avec des algorithmes forts. Ainsi les services de sécurité mentionnés ci-dessus sont fournis au moyen de deux extensions du protocole IP appelées AH (*Authentication Header*) et ESP (*Encapsulating Security Payload*) . Ce que nous allons voir dans la prochaine partie.

4. Mécanismes de sécurité IPsec :

4.1 Protocol AH (Authentication Header) :

4.1.1 Vue d'ensemble

AH est conçu pour assurer l'intégrité en mode non connecté et l'authentification de l'origine des datagrammes IP sans chiffrement des données donc pas de confidentialité. L'absence de confidentialité permet de s'assurer que ce standard pourra être largement répandu sur Internet, y compris dans les endroits où l'exportation, l'importation ou l'utilisation du chiffrement dans des buts de confidentialité est restreint par la loi.

Son principe est d'adjoindre au datagramme IP classique un champ supplémentaire permettant à la réception de vérifier l'authenticité des données incluses dans le datagramme. Ce bloc de données est appelé "valeur de vérification d'intégrité" (Integrity Check Value, ICV). La protection contre le rejeu se fait grâce à un numéro de séquence comme illustré dans le schéma ci-dessous :

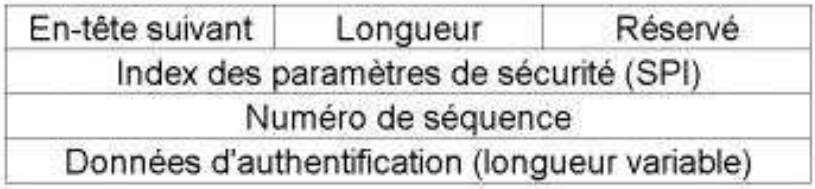


Figure 2.4.1 : Format de l' Authentication Header

4.1.2 Deux modes d'utilisation de AH : (mode transport et mode tunnel)

Le mode transport prend un flux de la couche transport et réalise les mécanismes de signature et de chiffrement puis transmet les données à la couche IP. Dans ce mode, l'insertion de la couche IPsec est transparente entre TCP et IP. TCP envoie ses données vers IPsec . L'inconvénient de ce mode réside dans le fait que l'en-tête extérieur est produit par la couche IP c'est-à-dire sans masquage d'adresse. De plus, le fait de terminer les traitements par la couche IP ne permet pas d'éviter des options IP potentiellement dangereuses. L'intérêt de ce mode réside dans une facilité de mise en œuvre. Dans le mode tunnel, les données envoyées par l'application traversent la pile de protocole jusqu'à la couche IP incluse, puis sont envoyées vers le module IPsec. L'encapsulation IPsec en mode tunnel permet le masquage d'adresses. Le mode tunnel est utilisé entre deux passerelles de sécurité (routeur, firewall, …) alors que le mode transport se situe entre deux hôtes. La figure x montre les différences essentielles entre les deux modes au niveau de l'interconnexion et au niveau des datagrammes.

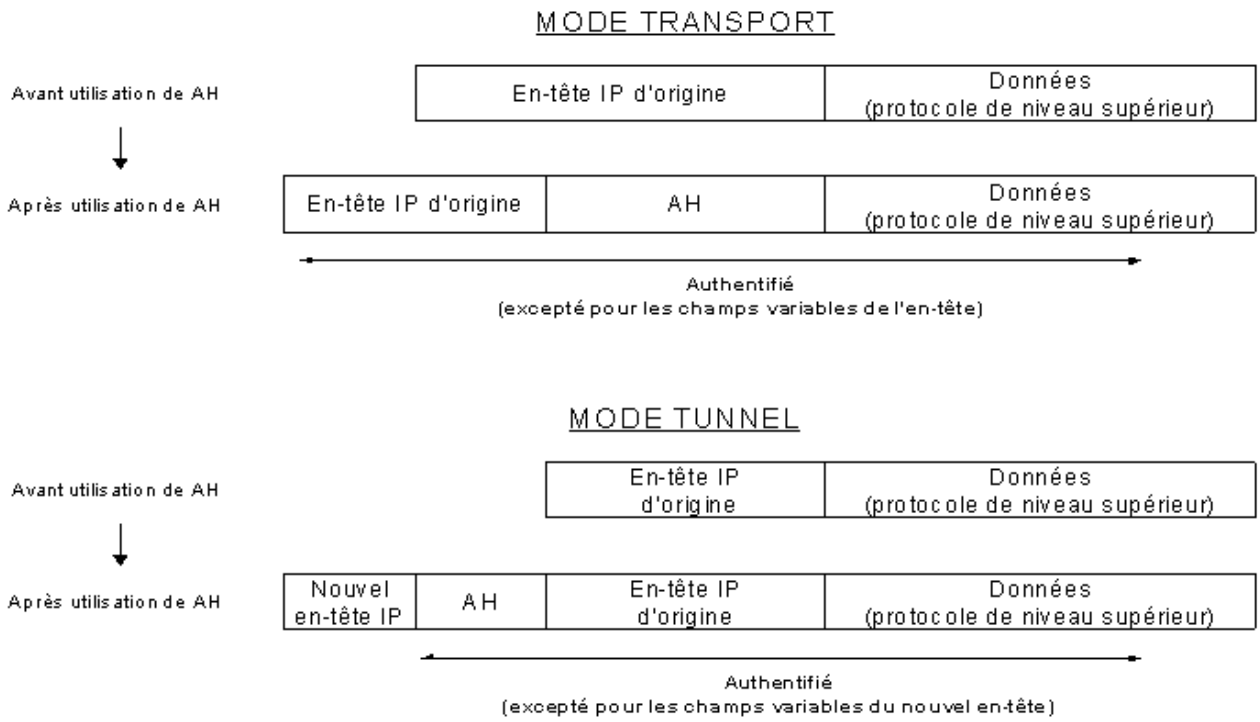


Figure 2.4.2 : Comparaison entre mode tunnel et mode transport

4.2 Protocol ESP (Encapsulating Security Payload) :

ESP peut assurer, au choix, un ou plusieurs des services suivants :

- confidentialité des données et protection partielle contre l'analyse du trafic si l'on utilise le mode tunnel
- intégrité des données en mode non connecté et authentification de l'origine des données, protection partielle contre le rejeu.

La figure ci-dessous montre le format de l'entête ESP :

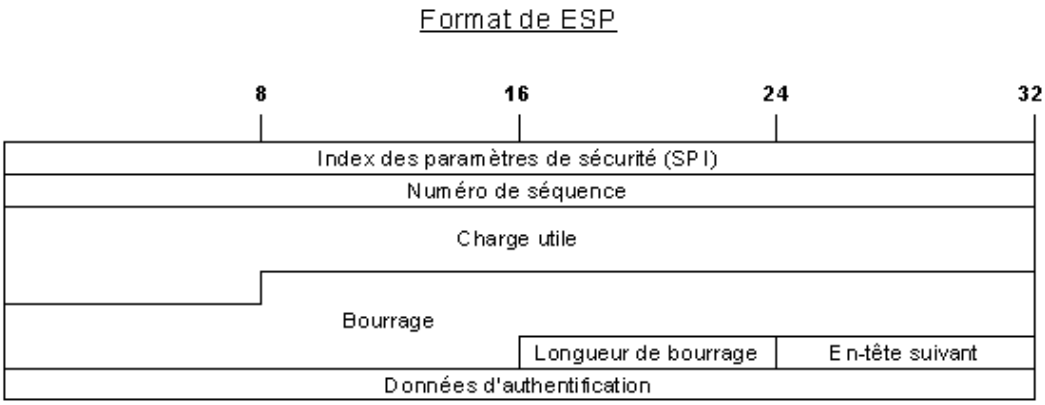


Figure 2.4.3 : Format de l'entête ESP

Contrairement à AH, où l'on se contentait d'ajouter un en-tête supplémentaire au paquet IP, ESP fonctionne suivant le principe de l'encapsulation : les données originales sont chiffrées puis encapsulées. Les figure x et y montre respectivement l'encapsulation ESP dans les deux modes tunnel et transport :

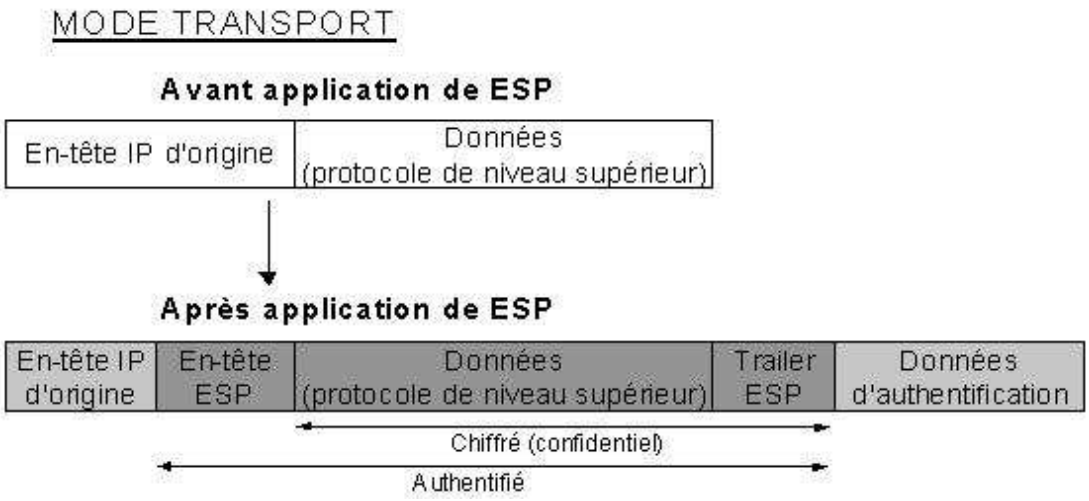


Figure 2.4.4 : Encapsulation ESP dans le mode transport.

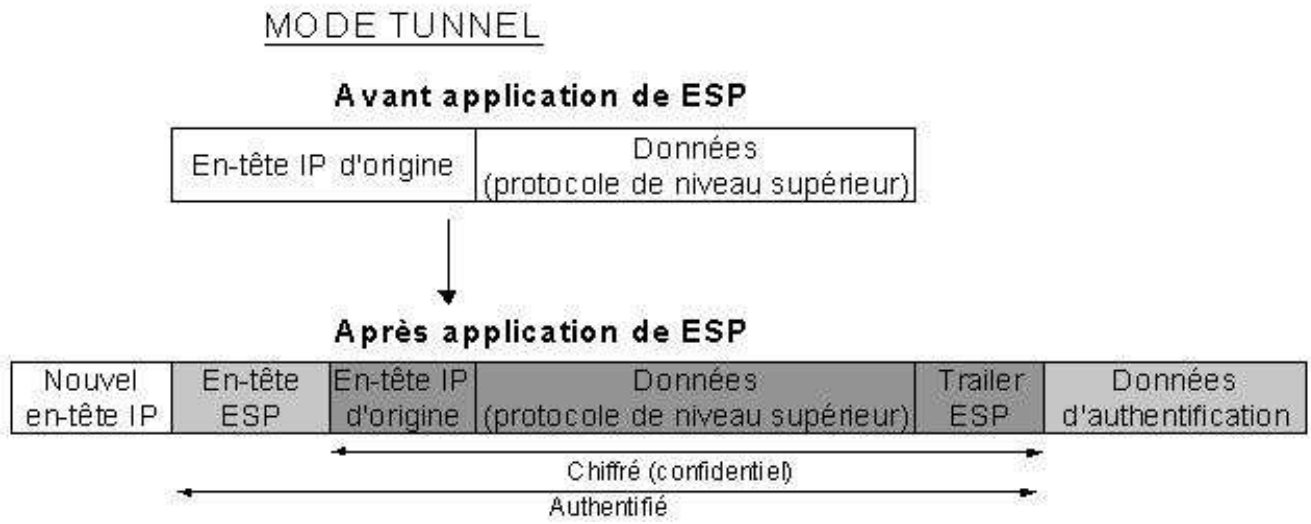


Figure 2.4.5 : Encapsulation ESP dans le mode tunnel.

5. Implémentation d'IP Sec

Dans cette partie nous allons donner une vue globale sur les capacités de quelques systèmes d'exploitation les plus populaires en matière de supports pour IPsec.

Les informations fournies constituent une synthèse des sites Web des systèmes d'exploitation cités, des éditeurs et des FAQs des logiciels concernés. Ainsi pour de plus amples détails voir ces sites illustrés dans la bibliographie.

Globalement nous pouvons citer les systèmes d'exploitation suivants :

5.1 Linux

L'implémentation IPsec la plus utilisée sous Linux est sous licence GPL, il s'agit de FreeS/WAN. La partie kernel de cette implémentation s'appelle KLIPS. Ce support ne gère pas les datagrammes dans lesquels des options IP sont présentes.

Enfin, FreeS/WAN décode les paquets IPsec avant de les transmettre au code de firewalling. Les règles de firewalling peuvent néanmoins tester si les datagrammes ont été envoyés en clair ou non.

5.2 Windows

Windows XP et 2000 incluent tous les deux un support du mode transport d'IPsec et de l'échange de clefs à l'aide de certificats en natif. Il existe de très nombreux logiciels permettant de créer des VPN sous Windows, y compris un serveur VPN en option sous Windows 2000 et XP, capable de mettre en place des tunnels IPsec/L2TP.

D'autre part, Windows 95, 98 et NT 4 disposent d'une grande quantité d'implémentations IPsec commerciales et à peu près toutes les versions de Windows depuis 95 sont capables de créer des

tunnels PPTP. En outre, Microsoft met à la disposition de ces clients un client IPsec gratuit pour Windows 95, 98 et ME.

5.3 NetBSD

NetBSD utilise KAME, c'est une implémentation IPsec et plus généralement IPv6 sous license BSD. La foire aux questions concernant IPsec pour NetBSD est disponible à l'adresse <http://www.netbsd.org/Documentation/network/ipsec/.

5.4 FreeBSD

FreeBSD s'appuie également sur KAME. (voir bibliographie pour la documentation sur L'utilisation d'IPsec sous FreeBSD).

5.5 OpenBSD

A l'instar de FreeBSD et NetBSD, OpenBSD utilise KAME(voir bibliographie pour la documentation relative à l'utilisation d'IPsec sous OpenBSD)

6. Implémentation du tunnel IPsec sous Linux

Actuellement, il existe plusieurs versions d'IPSEC qui sont disponibles pour Linux. la première implémentation majeure fût FreeS/WAN [5], qui existe pour les noyaux Linux 2.2 et 2.4. FreeS/WAN [5] n'a jamais été intégré dans le noyau pour un certain nombre de raisons. Celle qui est la plus souvent mentionnée concerne un problème politique avec les américains travaillant sur la cryptographie et qui freinent ainsi son exploitabilité. De plus, la mise en place de FreeS/WAN dans le noyau Linux est délicate, ce qui n'en fait pas un bon candidat pour une réelle intégration dans le noyau Linux,, le projet FreeS/WAN n'est plus à un stade actif de développement dés le 1er mars 2004. Apparemment un nouveau projet openSWAN qui prend la relève.

Néanmoins, On va s'intéresser dans le cadre de ce projet implémentation native d'IPSec est présente dans le noyau à partir de la version Linux 2.5.47. Elle a été écrite par Alexey Kuznetsov et Dave Miller, qui se sont inspirés des travaux du groupe USAGI IPv6. Avec cette fusion, les CryptoA-PI de James Morris deviennent également une partie du noyau, qui fait ainsi vraiment du cryptage.L'activation de cette dernière version nécessite :

1 : Source noyau > = 2.5.47 **(Lors de la compilation du noyau, soyez sûr d'activer 'PF_KEY', 'ESP,'AH' et tous les éléments de CryptoAPI !)**

2 : Les outils de configuration d'IPSec « **setkey** » .

Tout d'abord, nous montrerons en premier lieu comment configurer manuellement une communication sécurisée entre deux hôtes. En fait, une grande partie de ce processus peut être automatisée, mais nous le ferons ici à la main afin de bien comprendre les mécanismes de fonctionnement du protocole.

Configuration manuelle :

Les protocoles d'IP Sec ESP et AH s'appuient tous les deux sur des Associations de Sécurité (Security Associations (SA)). Une Association de Sécurité (SA) consiste en une source, une destination et une instruction.

Un exemple simple d'Association de Sécurité (SA) pour l'authentification AH peut ressembler a ceci :

```
add 152.81.48.21 152.81.48.22 ah 15700 -A hmac-md5 "1234567890123456";
```

Ceci indique que le trafic allant de 152.81.48.21 vers 152.81.48.22 a besoin d'un En-tête d'Authentification (AH) qui peut être signé en utilisant HMAC-MD et le secret 1234567890123456. Cette instruction est repérée par l'identificateur SPI (Security Parameter Index) 15700, dont nous parlerons plus en détail par la suite. Le point intéressant à propos des Associations de Sécurité (SA) est qu'elles sont symétriques. Les deux cotés de la conversation partagent exactement la même Association de Sécurité (SA), qui n'est pas recopiée sur l'hôte distant. Notez cependant qu'il n'y a pas de règles "d'inversion automatique". Cette Association de Sécurité (SA) décrit une authentification possible de 152.81.48.21 vers 152.81.48.22. Ainsi pour un trafic bidirectionnel, deux associations de sécurité (SA) sont nécessaires.

Un exemple d'association de sécurité (SA) pour ESP peut ressembler a ceci :

```
add 152.81.48.21 152.81.48.22 esp 15701 -E 3des-cbc   "123456789012123456789012";
```

Ceci signifie que le trafic allant de 152.81.48.21 vers 152.81.48.22 est chiffré en utilisant 3des-cbc avec la clé « 123456789012123456789012 » l'identificateur SPI est 15701.

Jusqu'ici, nous n'avons vu que les Associations de Sécurité (SA) qui décrivent les instructions possibles, mais pas la politique qui indique quand ces SA doivent être utilisées. En fait, il pourrait y avoir un nombre arbitraire de SA presque tous identiques et ne se différenciant que par les identificateurs SPI. Entre parenthèses, SPI signifie Security Parameter Index, ou Index du Paramètre de Sécurité en français. Pour faire vraiment du cryptage, nous devons décrire une politique. Cette politique peut inclure des choses comme "utiliser ipsec s'il est disponible" ou « rejeter le trafic à moins que vous ayez ipsec ».

Une « Politique de sécurité » (Security Policy (SP)) typique ressemble à ceci :

spdadd 152.81.48.21 251.81.48.22 any –P out ipsec

esp/transport//require

ah/transport//require;

Si cette configuration est appliquée sur l'hôte 152.81.48.21, cela signifie que tout le trafic allant vers 152.81.48.22 doit être encrypté et encapsulé dans un en-tête d'authentification AH. Notez que ceci ne décrit pas quelle SA sera utilisée. Cette détermination est un exercice laissé à la charge du noyau.

En d'autres termes, une Politique de Sécurité spécifie CE QUE nous voulons, une association de Sécurité par contre décrit COMMENT nous le voulons. Ainsi les paquets sortants sont étiquetés avec le SPI SA ('le comment') que le noyau utilise pour l'encryptage et l'authentification et l'hôte distant peut consulter les instructions de vérification et de décryptage correspondantes.

7. Conclusion :

Forces et faiblesses du cadre protocolaire IPsec

Avantages :

- Modèle de sécurité flexible et non exhaustif basé sur une boîte à outils modulaire
- Possibilité d'instaurer plusieurs crans de sécurité : chiffrement faible ou fort et/ou authentification
- Services de sécurité totalement transparent pour les applications

Inconvénients :

- Mécanismes de sécurité trop nombreux, engendrant un système complexe
- IPsec interdit la translation d'adresses (Network address translation, NAT)
- L'interaction du protocole IKE avec les infrastructures à clé publique (PKI) est possible mais il reste à normaliser
- Les outils d'administration centralisée des règles de sécurité (Security Policies) font défaut
- Entorses propriétaires nuisibles à l'interopérabilité
- IPsec est limité à l'instauration de VPN entre réseaux (passerelles-serveurs). Le protocole orienté client-serveur IPSRA (IPsec Remote Access) devra s'imposer face à PPTP ou L2TP.

Chapitre 3 : Active XML et les Services Web.

Chronologiquement, voici les différents outils sur lesquels nous avons travaillé concernant le développement de services Web pour la configuration.

1. Les services Web

Les Services Web sont des systèmes logiciels publiables localisés et invoqués via le Web, ils permettent de proposer diverses fonctionnalités à d'autres programmes grâce à des protocoles dits de l'Internet et basés sur l'échange de messages décris en XML.

Un Service Web est décrit dans un document WSDL (Web Services Description Language)[12], précisant les méthodes pouvant être invoquées, leurs signatures, et les points d'accès au service (URT ,Port ..). Ces méthodes sont accessibles via SOAP [10], la requête et la réponse sont des messages XML transportés par HTTP. Un service Web est accessible depuis n'importe quelle plate-forme ou langage de programmation. On peut utiliser un Service Web pour exporter les fonctionnalités d'une application et les rendre accessibles via les protocoles standard. Les Services Web fournissent une architecture générale pour les applications réparties sur Internet, qui peuvent également communiquer au moyen de middlewares (RMI, EJB, CORBA, etc.).

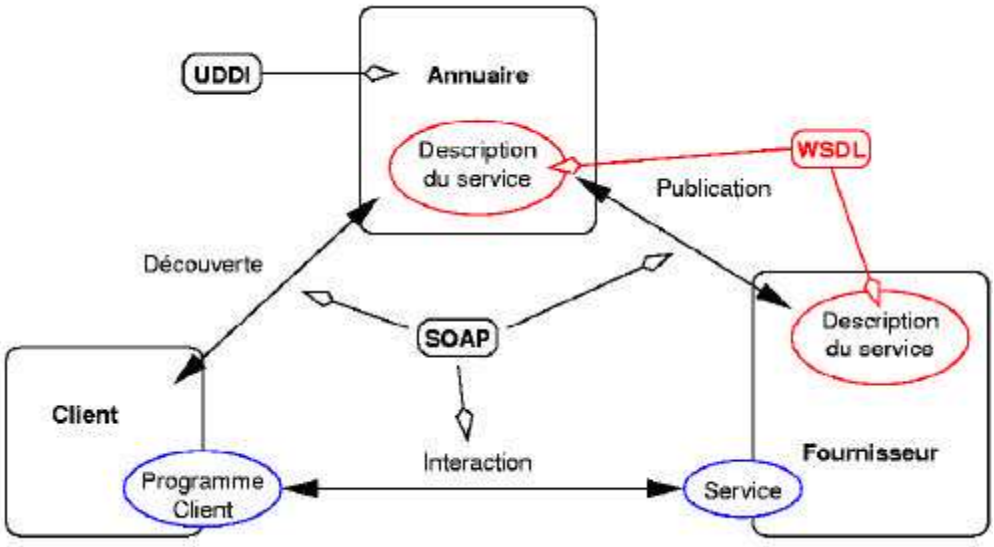


Figure 3.1.1 : schéma général de fonctionnement.

1.1 SOAP

SOAP (Simple Object Acces Protocol), développé sous les auspices du W3C, est défini comme un protocole léger permettant d'échanger les informations structurées et typées dans un environnement distribué.

1.2 WSDL

WSDL (Web Service Description Laguage) est un standard XML visé par le W3C, qui est utilisé pour décrire les Web Services et pour permettre aux clients de savoir comment y accéder. Le fichier de description WSDL d'un Web Service contient un ensemble de définitions qui décrit :

- **L'interface du service** : c'est à dire les opérations que le service fournit, les formats de données et les protocoles utilisés.
- **L'implémentation du service** : c'est à dire l'adresse applicative (URL) pour accéder au service.

1.3 UDDI

UDDI (Universal Description, Discovery and Integration) est un standard spécifié par OASIS, pour la publication et la découverte des Services Web. UDDI définit à la fois un modèle structurel des données (UDDI Registry) et les API que l'opérateur d'annuaire doit fournir pour publier et découvrir les services. Concrètement, un annuaire UDDI permet à un client de récupérer la description WSDL d'un Service Web .

1.4 Synthèse

Les Services Web améliorent l'intégration et l'interopérabilité entre services à travers l'infrastructure Web en utilisant différents standards XML, SOAP pour l'échange de messages, WSDL pour la description de services et UDDI pour la publication et la découverte de services. Ils correspondent à des composants logiciels qui peuvent être combinés grâce à un langage de composition, pour former de nouveaux services plus élaborés.

2. Active XML

Active XML est une approche qui exploite les services Web pour réaliser l'intégration des données au sein d'une architecture pair à pair (figure 4.5). Typiquement l'intégration de données consiste à rechercher et à combiner les données provenant de différentes sources en respectant un schéma prédéfini. Ces données peuvent ensuite servir à un autre schéma d'intégration et ainsi de suite de façon hiérarchique.

De façon à offrir une plus grande flexibilité et un meilleur passage à l'échelle, Active XML élimine cette contrainte hiérarchique : chaque pair joue un rôle symétrique à la fois fournisseur et intégrateur de données.

De plus, Active XML apporte une solution au problème de l'hétérogénéité des sources de données externes grâce à l'utilisation des Services Web, et cela aussi bien dans la communication inter-pairs que dans la communication entre un pair et une source de données externe.

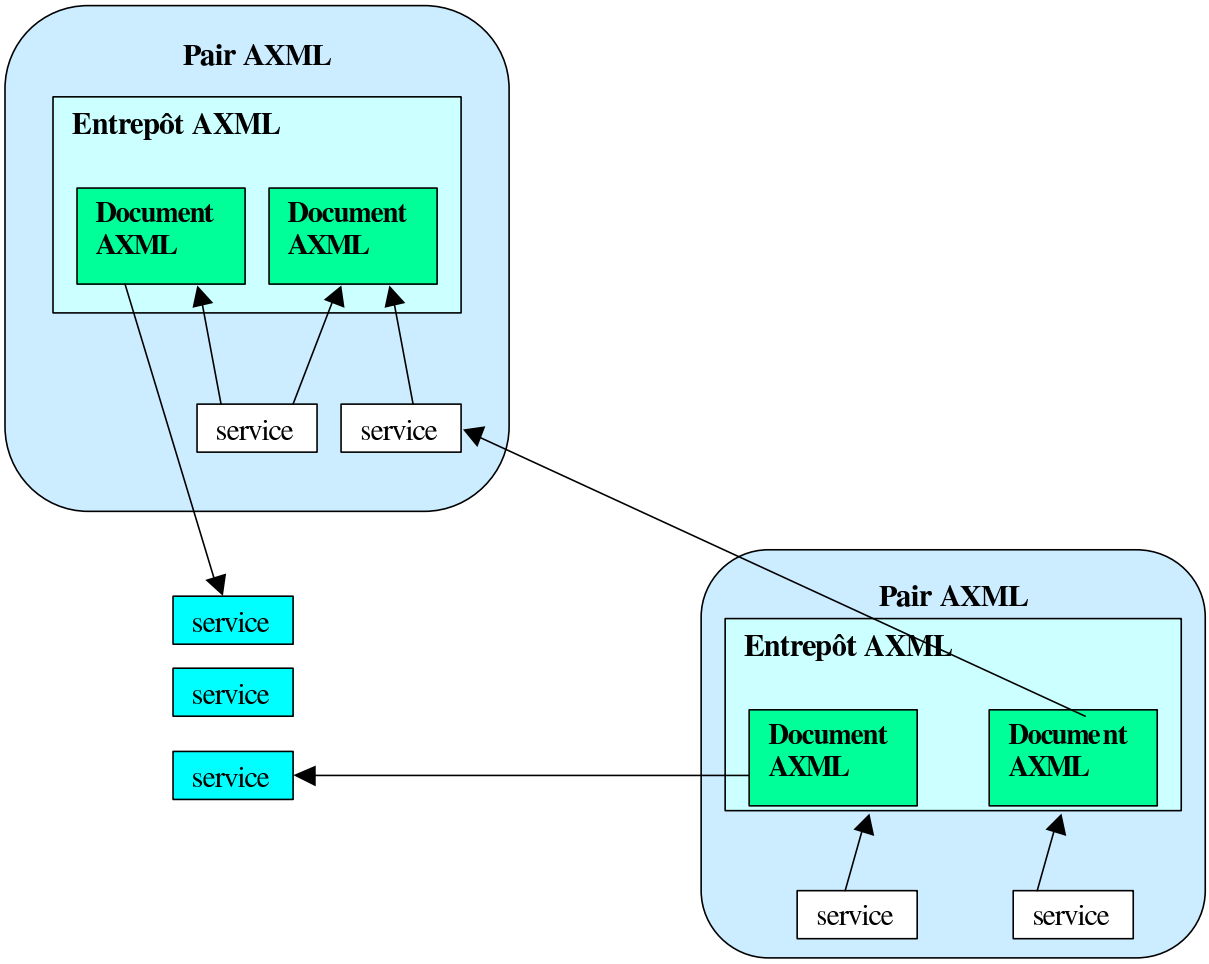


Figure 3.2.1 : Environnement Active XML

Active XML est centré autour des documents AXML, qui correspondent à des documents XML qui contiennent des appels à des Services Web. Le langage permet à la fois de spécifier de tels document et de définir grâce à XQuery[9] de nouveaux Services Web.

Les caractéristiques de ce framework sont :

- La résistance au passage à l'échelle,
- Le contrôle du moment d'activation des appels de services,
- Le contrôle de la durée de vie des données recueillies,
- L'échange de données explicites ou implicites (par référence).

2.1 Pair AXML

Un pair AXML (Cf .figure) est composé à la fois de repository (répertoires ou entrepôts), de documents AXML et d'un repository de documents définissant les Services Web qu'il expose. Les Services Web exposés correspondent à l'exécution d'une requête Xquery[5] sur un document AXML du pair. Nous avons détaillé les notions de documents AXML et de documents de définitions de Services Web dans la suite de cette partie.

2.2 Le document AXML

Un document AXML peut être vu comme une matrice partiellement matérialisée intégrant du code XML statique ainsi que des données dynamiques obtenues par un appel à un Service Web . De ce fait, si la source de données change, la partie dynamique du document est changée.

Cet appel de service implicite est représenté par l'introduction d'une balise <sc> au sein du document AXML comme présenté dans l'exemple ci-dessous :

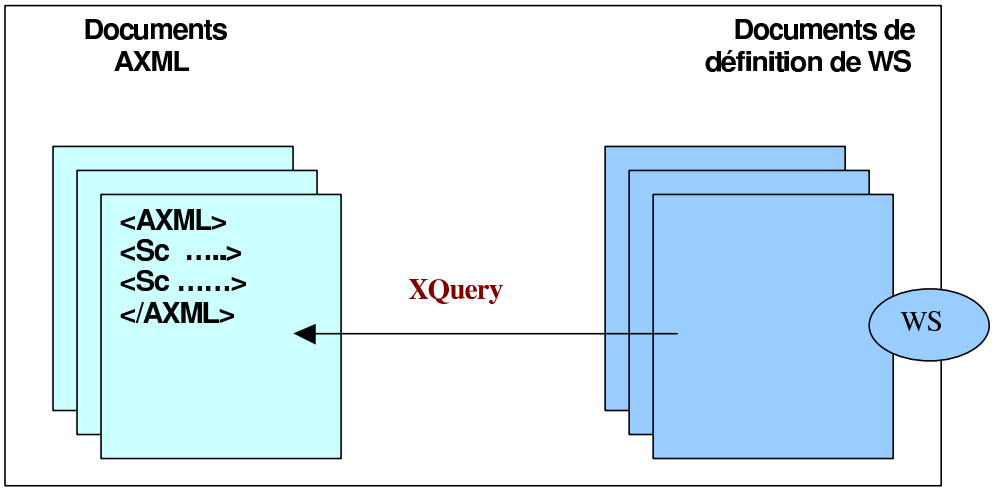


Figure 3.2.2 –Schéma d'un pair Active XML

```
<monproduit> je suis intéressé par le  produit
  <catégorie  nom= « boisson» >
     <sc>enchère.com/getOffert(« boisson »)</sc>
  </catégorie>
</monproduit>
```

Le résultat de cet appel produit l'intégration des données réceptionnées au sein du document AXML comme ci dessous :

```
<monproduit> je suis intéressé par le  produit
  <catégorie  nom= « boisson» >
     <sc>enchère.com/getOffert(« boisson »)</sc>
     <enchère aID= « 1 »> <description> jus d'orange </description></enchère>
     <enchère aID= « 1 »> <description> ? </description></enchère>
     <sc>enchère.com/getOffert(« importboisson »)</sc>
  </catégorie>
</monproduit>
```

Comme on peut le constater dans l'exemple ci-dessus, la réponse à un appel de service peut elle même en comporter un autre.

2.3 Activation des appels

L'activation des appels de services est contrôlée par deux attributs additionnels de la balise « sc » : mode et frequency. On distingue trois types d'appel :

- Les appels explicites (mode « pull », immédiat),

  L'appel est activé ou réactivé dans deux cas :

- lorsque la durée spécifiée par l'attribut « frequency » (1jours, 1semaine…) est écoulée.
- lorsque d'autres appels de services sont effectués

- Les appels implicites (en mode « pull »,lazy) ,

  L'appel est activé lorsqu'il a expiré et que les données sont demandées,

- Les appels en mode « push ».

  Basée sur le mécanisme de souscription (le serveur Web envoie les données au client), l'appel est activé lors de la synchronisation avec une source de données externes.

Il est à noter que selon la façon d'effectuer l'appel, on peut contrôler quel pair effectue le travail.

2.4 Fusion des résultats des appels de services

Les résultats des appels étant stockés dans le document appelant, il est nécessaire de pouvoir remplacer les anciens résultats par les nouveaux et de ne pas récupérer les données obsolètes pour éviter une accumulation. Un attribut « **ExpireOn(date)** »peut être attaché à tous les nœuds de données dans un document AXML pour spécifier une date à partir de laquelle les données ne sont plus pertinentes : dans ce cas les données sont considérées comme effacée du document . De plus, dans le cas où il n'est pas précisé.

5 Conclusion

Active XML est une approche qui exploite les services Web pour réaliser de l'intégration de données au sein d'une architecture pair-à-pair. Elle permet de spécifier des documents contenant des appels à des Services Web et de définir de nouveaux Services Web basés sur ses derniers. De plus cette approche contrôle le moment d'activation des appels de services et de la durée de vie des données recueillies. Enfin, elle présente une bonne résistance au passage à l'échelle.
Cet approche sera à la base de notre architecture d'auto configuration de VPN IPsec.

Chapitre 4 : Mission et traduction des objectifs

Nous avons présenté dans le chapitre précédent Active XML et les Services Web. Ces deux technologies sont à la base de notre approche pour la configuration dynamique d'IP Sec que nous présentons dans ce chapitre.

1. Expressions de besoins

- Concevoir un mécanisme d'échange dynamique des paramètres de sécurités requis pour configurer IPsec,
- Posséder des interfaces qui permettent différentes implémentations d'échanges des clés (exemple IKE Internet key Exchange),
- Exportation du model de gestion sur un réseau dynamique,
- Test de performance des différentes solutions proposées.

1.1 Etude et Conception

1.1.1 Gestion automatique des clés

Rappelons que la Politique de Sécurité spécifie CE QUE nous voulons tandis que l'Association de Sécurité décrit COMMENT nous le voulons. L'utilisation de la gestion automatique des clés nous permet de ne spécifier que ce que nous voulons.
Pour ce faire, nous avons étudié deux mécanismes d'échanges automatiques des clés, le premier basé sur une implémentation d'IKE appelée **Racoon** et l'autre sur l'approche Active XML.

1.1.1.1 Démon IKE

Comme expliqué dans le deuxième chapitre, l'idéal est de bien configurer les politiques, en ne fournissant aucune Association de Sécurité. Si le noyau découvre qu'il y a une politique IPsec, mais pas d'Association de Sécurité, il va le notifier au démon IKE qui va chercher à en négocier une. Pour

étudier ce mécanisme on va utiliser une implémentation IPSEC de Linux 2.5 fonctionnant avec le démon IKE "KAME Racoon" sur deux machines :

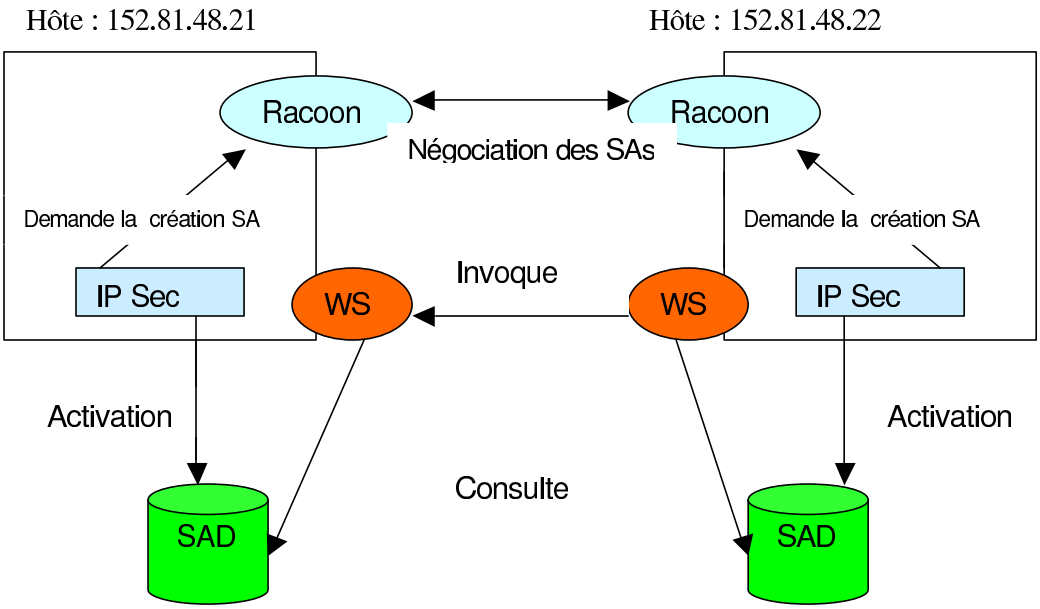


Figure 4.1.1 Gestion automatique des à l'aide de l'outil Racoon

Dans cet exemple, 152.81.48.21 et 152.81.48.22 sont sur le point d'établir des communications à l'aide du démon Racoon. Cette configuration utilisera des clés pré-partagées, les redoutés 'secrets partagés' (Config. Racoon. Annexe 5). Avec un temps d'activation de deux minutes, alors que les services web se mettent à extraire périodiquement les dates d'activation, Ceci nous permet d'extraire les informations suivantes :

Etape	Date d'activation sur la machine 152.81.48.21	Date d'activation sur la machine 152.81.48.22	Différence
1	12:17:49	12 :17 :29	20
2	12:19:27	12:19:11	16
3	12:21:03	12:20:47	16
4	12:24:16	12 :20 :00	16
5	12:25:54	12:25:38	16
6	12:27:30	12 :27 :13	17

1.1.1.2 Gestion automatique des clés avec AXML

Dans cette approche (Fig 3.2.2), chaque pair AXML possède à son lancement un document de management générique ainsi qu'un document de définition des Services Web associés.

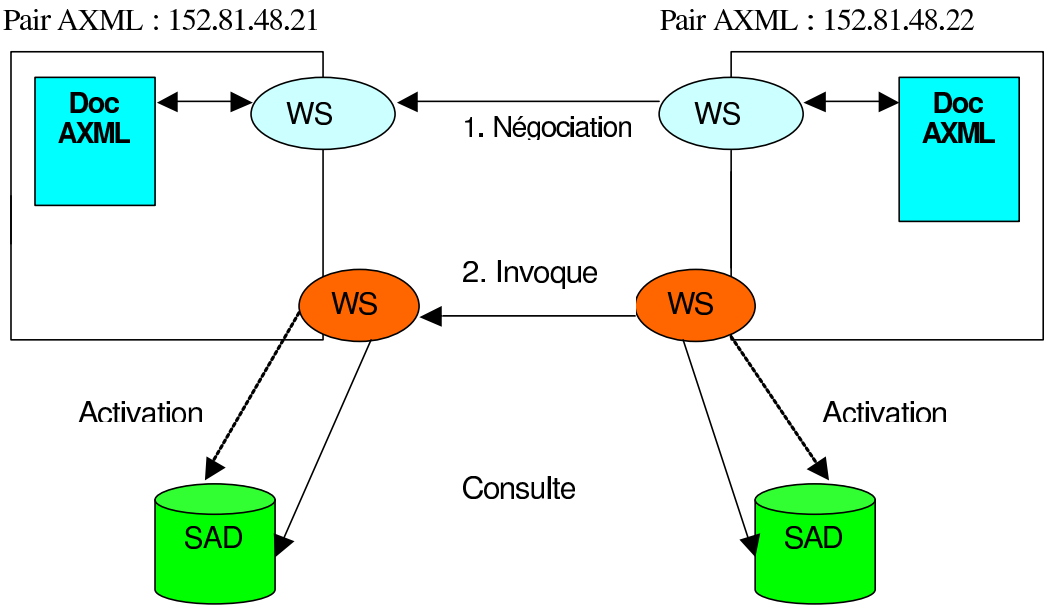


Figure 4.1.2 Gestion automatique des à l'aide de Document AXML

Dans cet exemple, les pairs 152.81.48.21 et 152.81.48.22 sont sur le point d'établir des communications sécurisées mais, cette fois, avec l'utilisation d'AXML. Chaque pair doit offrir deux fonctionnalités primaires :

1. Permettre aux autres pairs d'accéder à ses données concernant la politique de sécurité via un service Active XML.
2. Activation des nouvelles clés dynamiquement à l'aide de fichiers Active XML.

L'implémentation de cette architecture entre les deux machines, avec un temps d'activation de deux minutes, nous a permis de comparer les différentes méthodes présentées et nous avons obtenu le résultat suivant concernant la différence entre différentes dates d'activation des clés :

Etape	Date d'activation sur la machine 152.81.48.21	Date d'activation sur la machine 152.81.48.22	Différence
1	14:10:16	14:10 :06	10
2	14:12:10	14:12:02	8
3	14:14:11	14:12:04	7
4	14:16:15	14:12:07	8
5	14:18:14	14:12:06	8
6	14:20:10	14:12:02	8

1.1.2 Synthèse

Pour pouvoir comparer les résultats issus de chaque implémentation, nous avons rassemblé les courbes dans la **Figure 3.1.3 :**

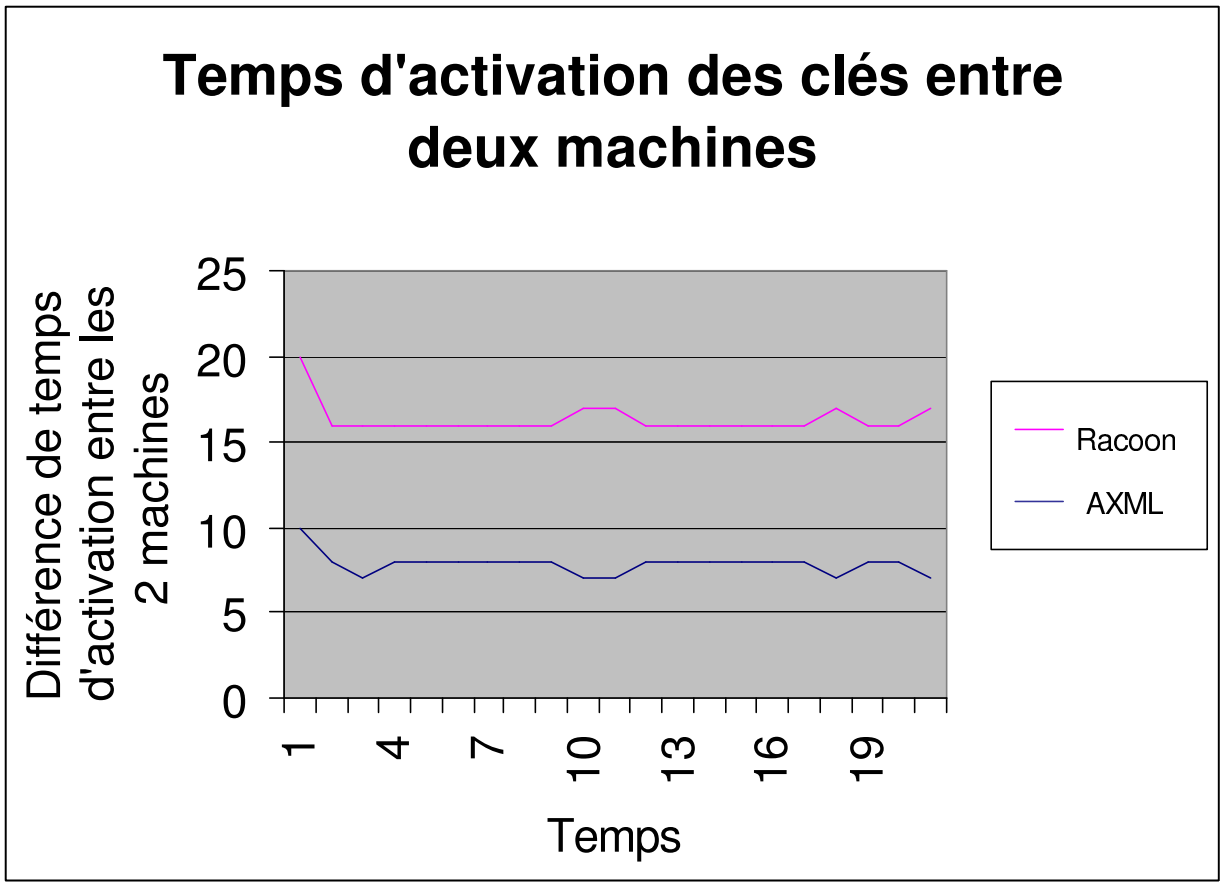


Figure 4.1.3 : diagramme d'implémentation des clés.

On voit bien que l'implémentation d' Active XML est moins coûteuse en terme de temps CPU, mais elle est moins sécurisée dans notre version actuelle par rapport au démon Racoon, et surtout ce n'est pas un standard. Au vu de ces avantages et de ces inconvénients, il nous semble judicieux de garder les deux implémentations et d'utiliser au choix celle correspondant à nos besoins.

2. Solution technique proposée

Rappelons que l'objectif du projet est de pouvoir alimenter dynamiquement des bases de données SAD et SPD à base de politiques de sécurités décrites par des documents XML sur un réseau dynamique. Pour y parvenir, trois grandes étapes sont nécessaires :

1. Auto configuration des interfaces connectées.
2. Distribution périodique des clés pour les associations de sécurité.
3. Distribution périodique des documents XML qui décrivent la politique de sécurité.

2.1 Auto configuration des interfaces connectées.

Cette fonctionnalité permet l'auto configuration des tunnels de n'importe quelle interface connectée au réseau. L'objectif est de fournir un modèle autonome de gestion des éléments réseau. Par conséquent, lorsqu'une station se connecte à un réseau, elle doit pouvoir connaître, sans que l'utilisateur ou l'administrateur n'ait à renseigner aucune information, ni son adresse IP ni l'adresse d'un annuaire par défaut.

Lorsqu'une station se connecte au réseau, elle procède à la découverte de l'annuaire, ce qui permet d'ajouter à ce dernier une nouvelle entrée correspondant à sa connexion (fig 4.2.1).

Après l'identification, l'annuaire répond par un message qui contient les informations nécessaires à la configuration des tunnels, comme les adresses IP des autres stations ou encore la politique de sécurité par défaut. Connaissant les adresses IP, la nouvelle station peut construire des documents Active XML qui permettent l'activation périodique des deux dernières étapes.

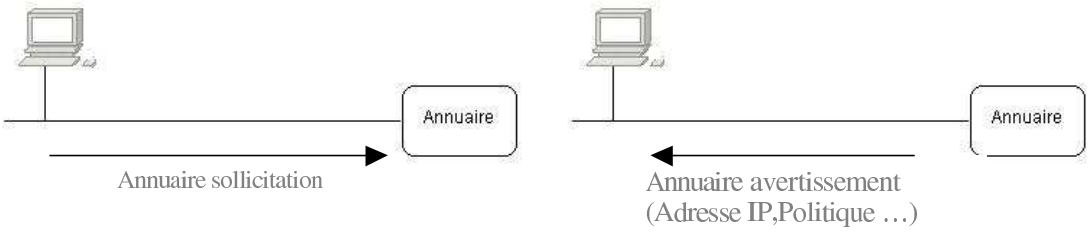


Figure 4.2.1 : Consultation d'annuaire

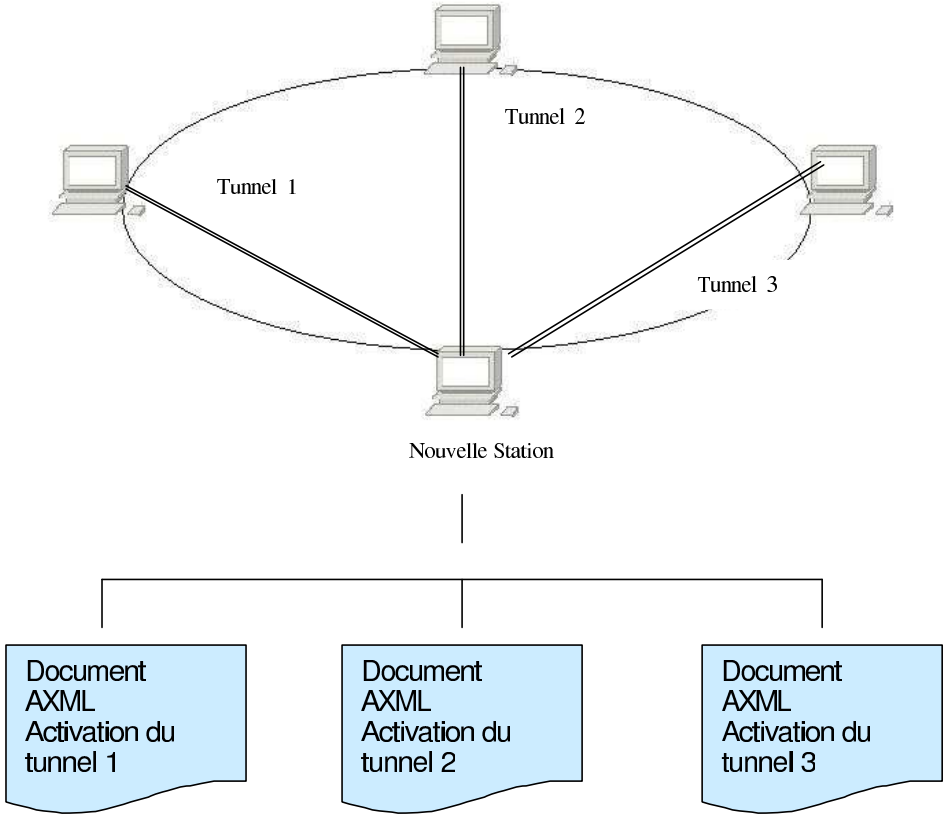


Figure 4.2.2 : Création des documents AXML

2.2 Distribution périodique des paramètres de sécurité

La deuxième étape a pour objectif la distribution sécurisée des clés et politiques de sécurité. Pour y parvenir, trois phases sont nécessaires. La première phase se déroule surtout avec des messages en clair. Son but est l'authentification et la négociation de l'échange des clés secrètes. La seconde phase a pour rôle de négocier les politiques de sécurité déployées pour chaque pair, et la troisième phase procède à la Configuration de Tunnel en fonction de la politique choisie dans la deuxième phase. Chacune des phases possède ses paramètres cryptographiques, générés lors de la première phase et qui servent à protéger les échanges. Ainsi, les paramètres des SAs IPsec ne sont connus que des deux parties. On peut voir l'ensemble des opérations à **la Figure 4.2.3**. Revenons maintenant plus en détail sur chacune des phases.

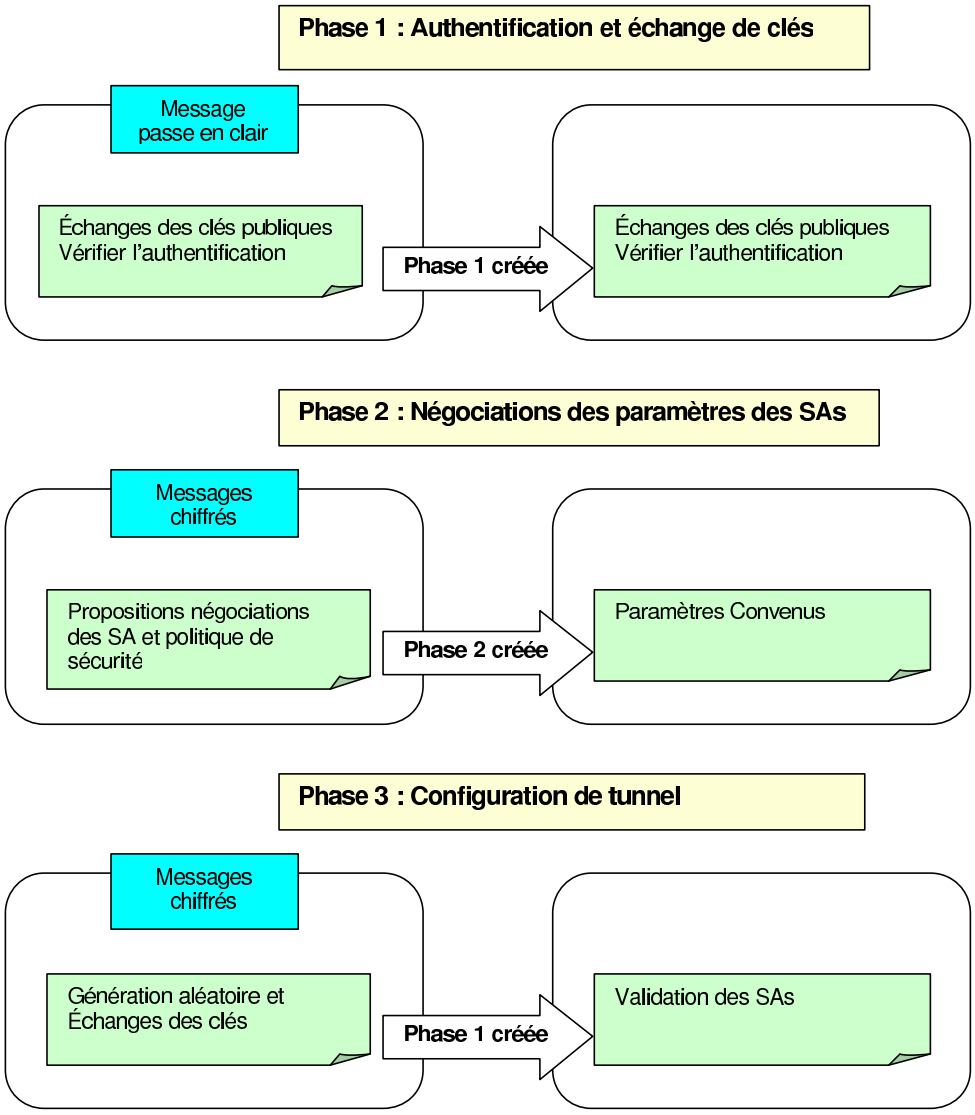


Figure 4.2.3 : Vue d'ensemble des phases 1 , 2 et 3

2.2.1 Phase 1

La première phase a pour but de paramétrer un canal d'authentification sécurisé entre les deux parties. Les premiers messages sont en clair. Ils servent à déterminer les paramètres qui sécuriseront les échanges futurs. On a employé l'algorithme d'échange de clés Diffie-Hellman pour générer les clés secrètes partagées. La première phase est réalisée en utilisant trois messages entre trois services Web déployés dans les deux pairs. L'invocation de la première phase se fait par le document AXML. Cette phase permet de générer trois secrets:

- une clé de chiffrement,
- une clé d'authentification,
- un autre secret partagé.

Les clés servent à chiffrer et à authentifier les échanges de la deuxième phase. Voyons cela d'un peu plus près grâce à un exemple entre deux machines « Adil » et « Khalid ».

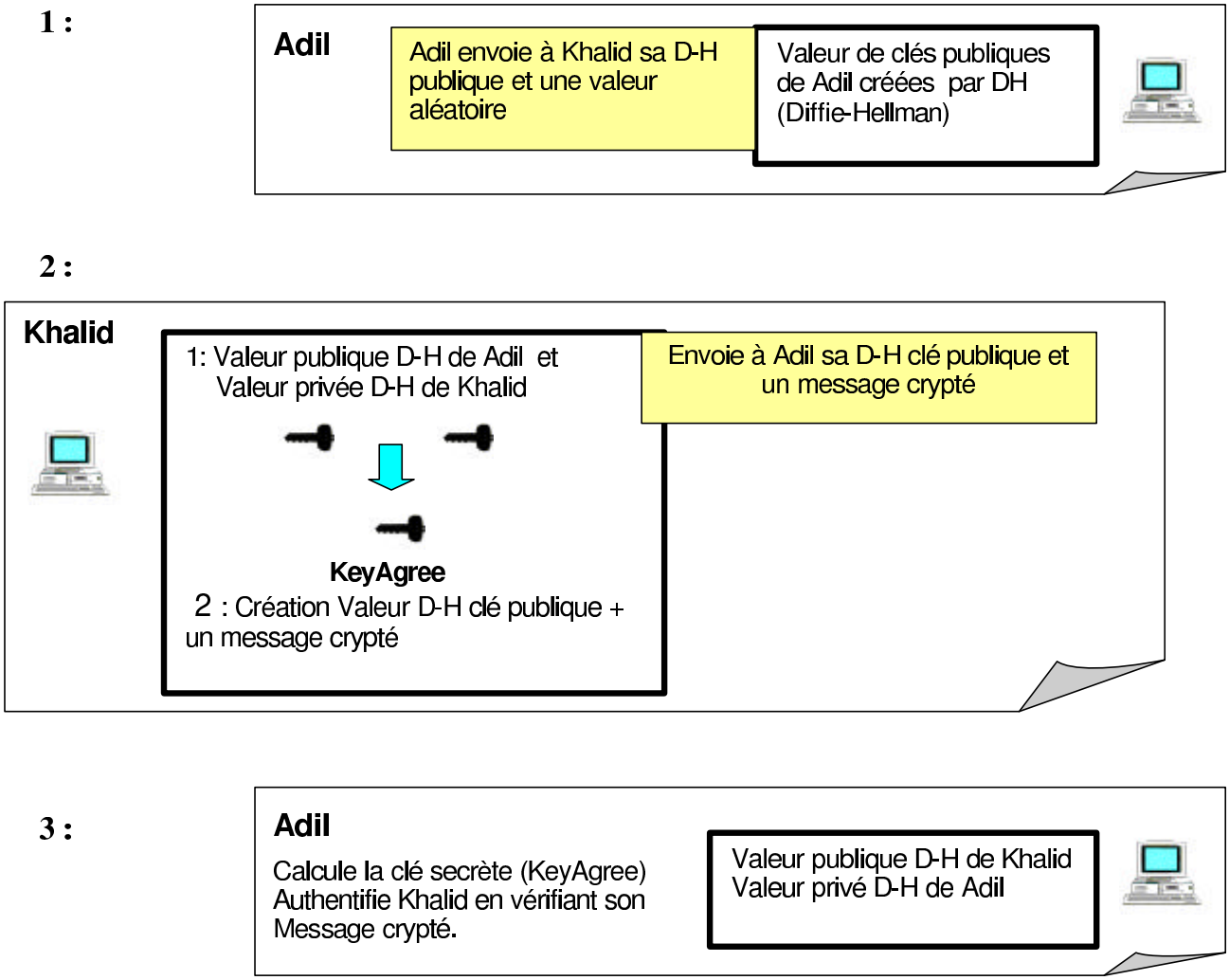


Figure 4.2.4 : Phase d'authentification

1. Adil initie l'échange et envoie sa valeur DH publique et une valeur aléatoire .
2. Khalid utilise la valeur DH et aléatoire de Adil ainsi que sa propre valeur DH privée pour calculer la clé secrète partagée KeyAgree. Khalid envoie à Adil sa valeur DH publique et son message crypté .
3. Adil emploie la valeur DH publique de Khalid et la sienne pour calculer à son tour le KeyAgree. Adil vérifie la preuve de Khalid.

2.2.2 Phase 2

Tous les messages de la deuxième phase sont chiffrés et authentifiés par les clés secrètes partagées de la phase précédente. La deuxième phase peut s'achever en quatre étapes. Durant cette phase, les deux parties négocient les paramètres des SAs IPsec et Politique de sécurité décrits dans un Document AXML (Document de def .WS,cf Annexe 3) et traités par un service Web .

Voyons, toujours grâce à un exemple entre deux Machine Adil et Khalid, comment se déroule la troisième phase. Les échanges sont repris à la figure 4.2.3.

Adil, qui initie la connexion, récupère la proposition faite à Khalid pour les paramètres des SAs IPsec par un appel à un service écrit en XQuery[10]. Adil compare sa proposition envers Khalid avec la proposition de Khalid et il génère un document XML des paramètres de sécurités pour les deux pair. Ce Document sera à la base de la configuration troisième phase

Cette phase est invoquée continuellement pour prendre en considération les mises à jour proposées à l'aide d'une interface graphique.

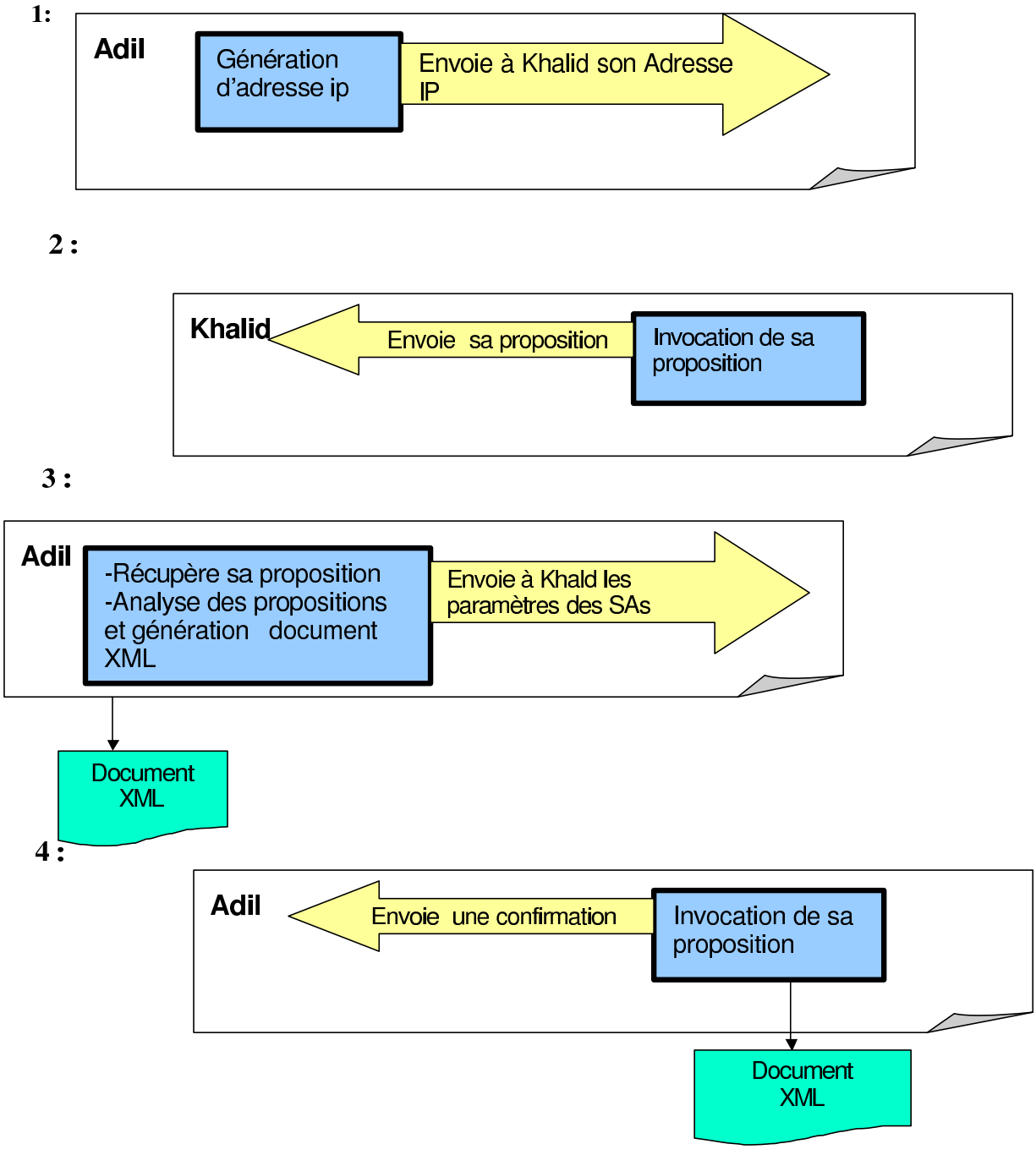


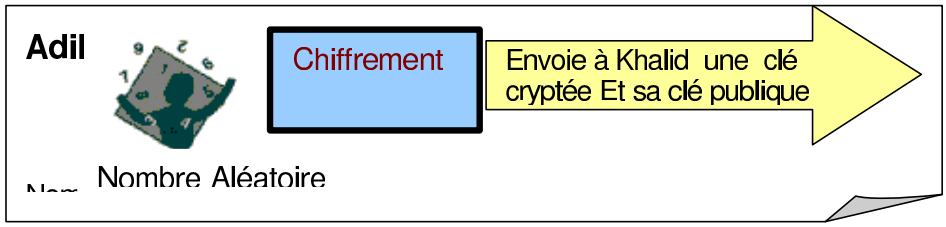
Figure 4.2.4 : Phase de négociation de politique de sécurité.

3.2.3 Phase 3

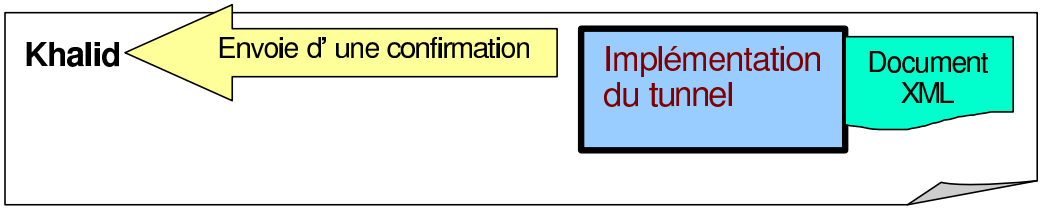
La troisième phase a pour but de paramétrer un tunnel sécurisé entre les deux parties, cela se fait grâce à un document XML fourni lors de la phase précédente et des valeurs aléatoires générées (ces valeurs seront les clés de l'algorithme de cryptage d' IPSec), tous les messages de cette phase sont chiffrés et authentifiés par les clés secrètes partagées de la première phase .

Voyons, toujours grâce à un exemple entre deux Machines Adil et Khalid, comment se déroule la troisième phase. Les échanges sont repris à la Figure 3.2.5.

1 :



2 :



3 :

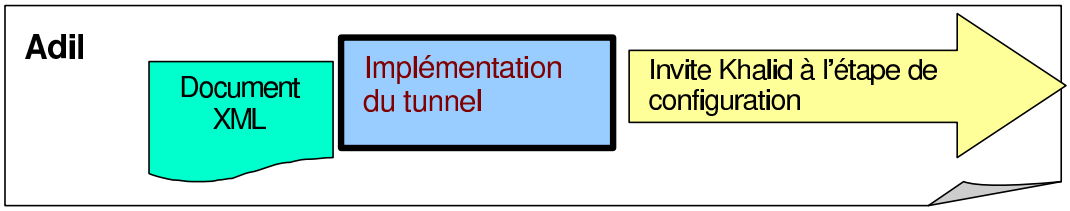


Figure 4.2.5 : Phase de configuration

1. Adil qui initie l'échange envoie une valeur aléatoire cryptée.
2. Khalid récupère la valeur cryptée de Adil et procède à l'implémentation du tunnel suivant la politique de sécurité décrite dans un document XML. Khalid envoie une confirmation à Adil
3. Suite à la réception d'une confirmation, Adil procède aussi à l'implémentation de tunnel.

Notons que le document XML qui décrit la politique de sécurité est identique au niveau des deux partenaires.

4. Conclusion

En résumé, cette architecture permet l'implémentation de différentes fonctions de la gestion automatique de configuration d'IPsec, tout en respectant au mieux les critères d'autonomie, de passage à l'échelle et d'intégration de solutions existantes. Dans ce qui suit, nous présentons le travail effectué pour son implémentation.

Chapitre 5 : Modélisation

Nous avons utilisé UML [14], le langage unifié de représentation des modèles, pour la conception des modèles de notre projet. Dans ce chapitre, nous donnons les différents diagrammes correspondant aux modèles élaborés.

1. Diagramme de Classes

Nous allons décrire dans cette partie l'application module par module, en partant d'une vision globale par l'approche UML pour décrire ensuite les classes de chaque module.

1.1 Le module de Sécurité

Ce module comporte un ensemble de classes qui offre des services de sécurité basés sur l'algorithme de Diffie-Hellman. Voyons sa structure sous forme de diagramme en modélisation UML.

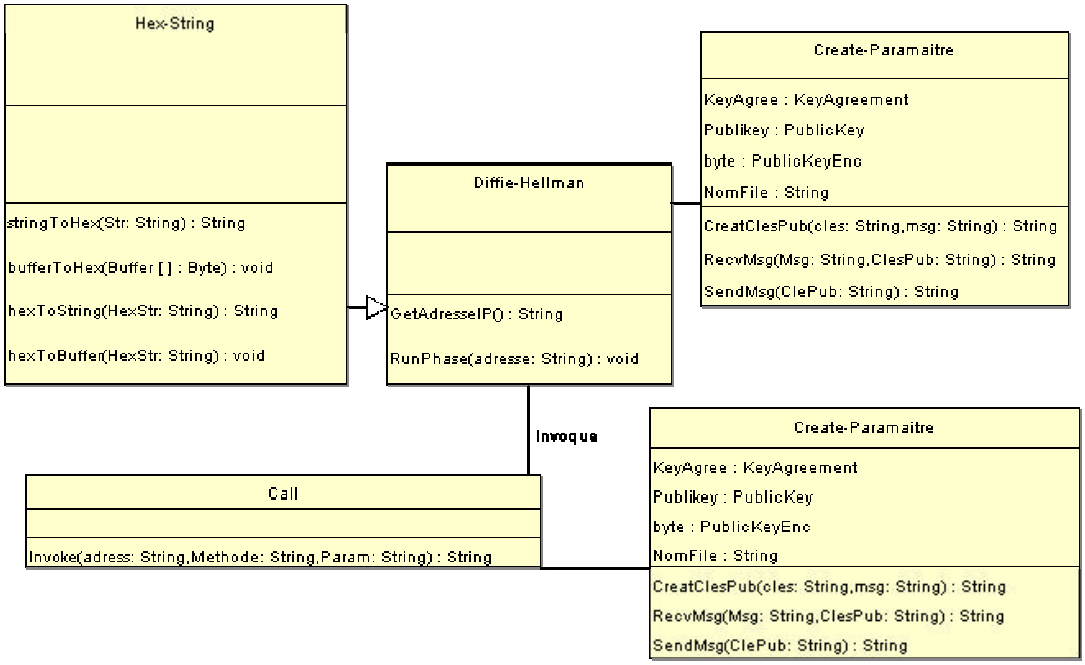


Figure 5.1.1 Diagramme de classes du module Sécurité

1.2 Le module de configuration

Ce module comporte un ensemble de classes qui offre des Services Web pour la manipulation des bases de données des associations de sécurité SAD et bases de politiques de sécurité SPD. Voyons sa structure sous forme de diagramme en modélisation UML [14] (nous ne faisons apparaître que les méthodes les plus importantes, les méthodes d'accès aux données privées sont ignorées).

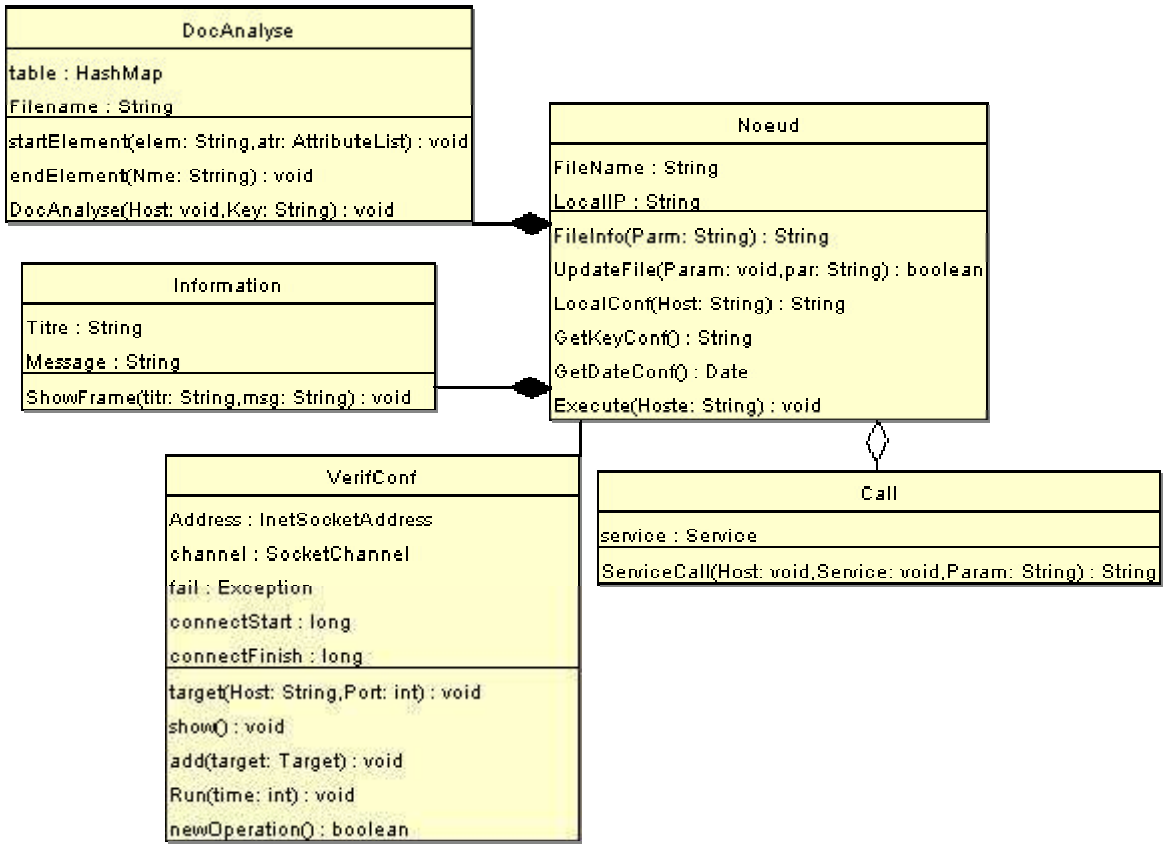


Figure 5.1.2 Diagramme de classes du module Configuration

1.3 Module d'interface IHM

Ce module comporte un ensemble de classes qui offre des interfaces homme machine**.** Voyons sa structure sous forme de diagramme en modélisation UML.

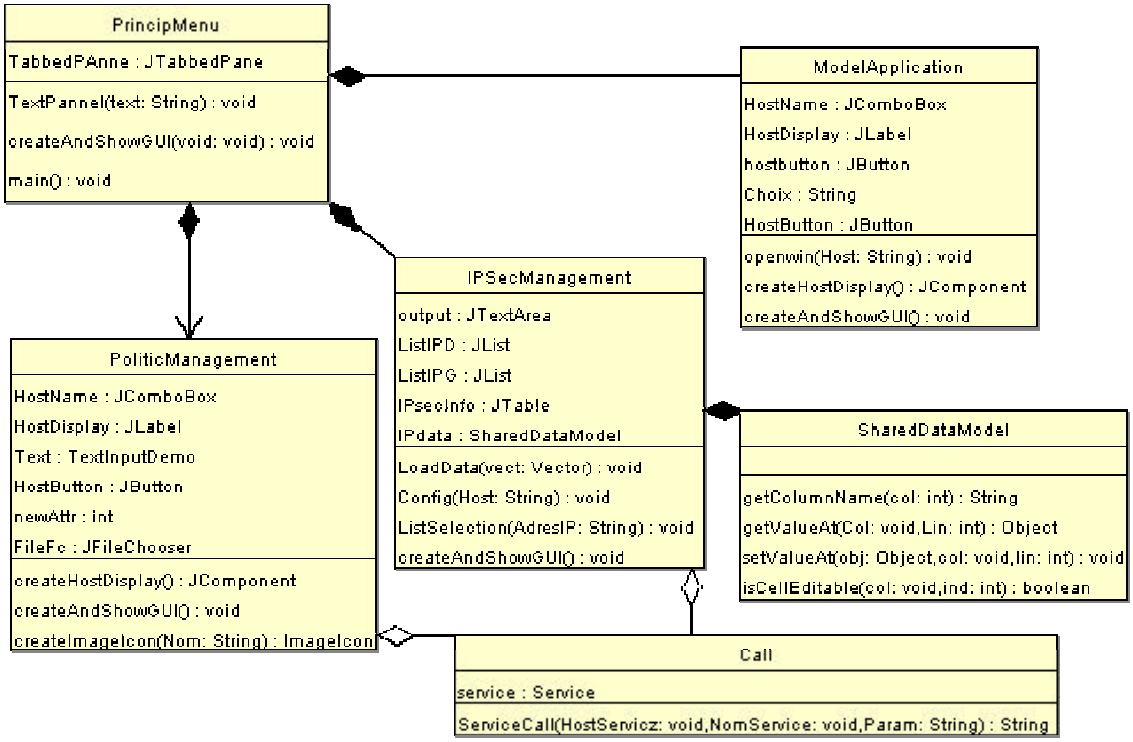


Figure 5.1.3 Diagramme de classes du module interface IHM

2. Diagrammes de cas d'utilisation

2.1 Enregistrement

La figure 5.2.1 représente le diagramme du cas d'utilisation «Enregistrement » concernant l'identification des éléments réseau.

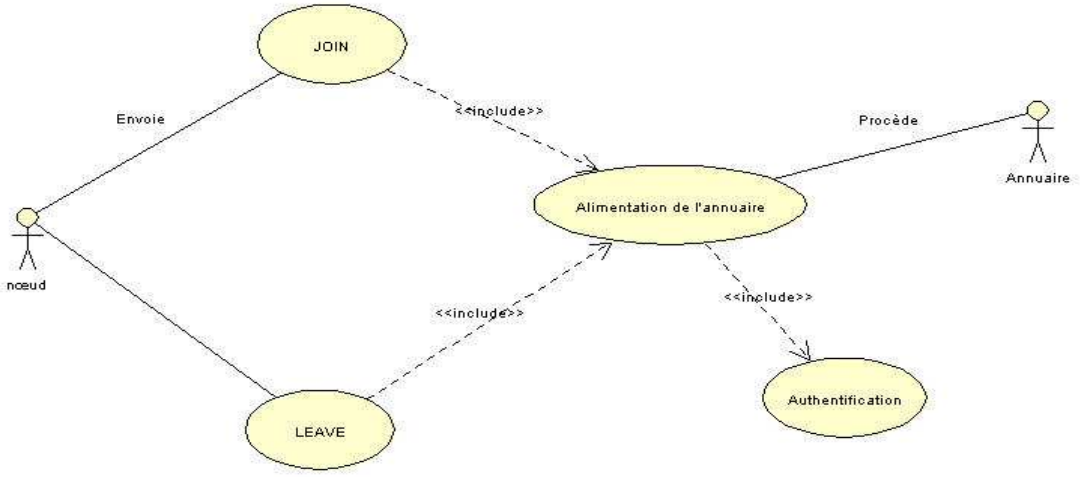


Figure 5.2.1 Cas d'utilisation d'enregistrement.

Sur le plan de management, c'est à dire pour ce qui concerne les éléments réseau, suite à la réception d'un Join ou d'un Leave, l'annuaire stocke les informations (adresse IP, WAN) concernant l'élément en cause.

2-2 Création des fichiers Active XML

La figure 5.2.2 représente le diagramme du cas d'utilisation « Création des fichiers Active XML » pour chaque élément réseau enregistré dans l'annuaire.

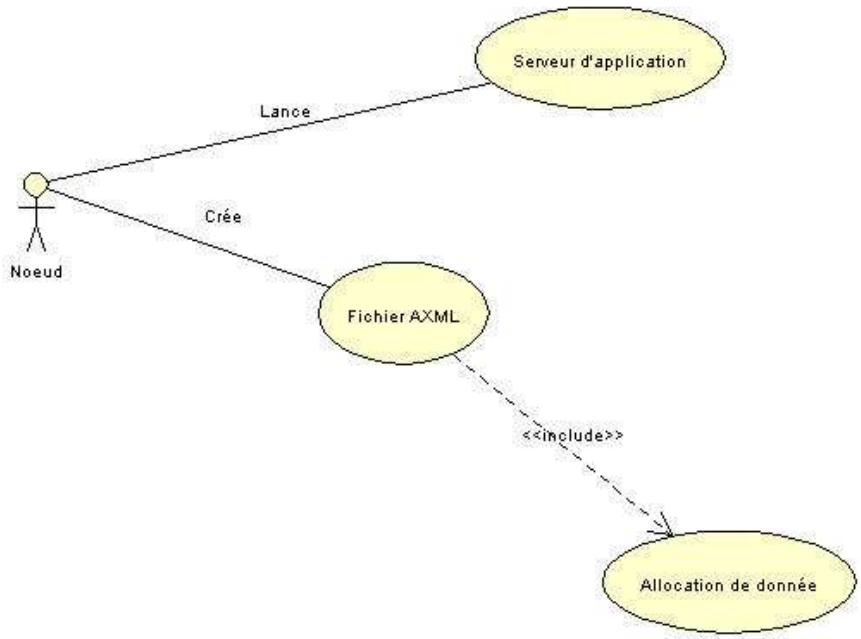


Figure 5.2.2 Cas d'utilisation Création des fichiers Active XML

Une fois qu'un élément est enregistré, il collecte les informations localisées au niveau de l'annuaire et procède à l'allocation des paramètres concernant le fichier AXML (temps d'appel de service Web, IP etc..), puis il lance le serveur d'application pour la prise en charge des mises à jour.

2.3 Mise à jour des SAD

La figure 5.2.3 représente le diagramme du cas d'utilisation « Mise à jour des SAD » concernant la mise à jour de la base de données des associations de sécurité (Security Association Database, SAD).

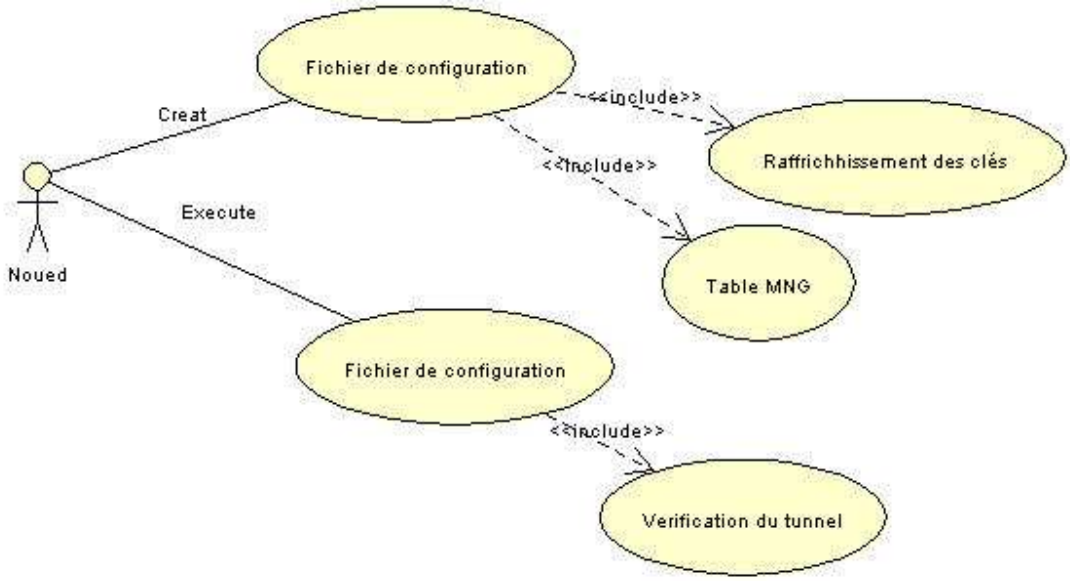


Figure 5. 2.3 Cas d'utilisation Mise à jour des SAD

Suite à l'appel du Web Service par le document AXML, l'élément réseau procède au rafraîchissement des clés et à l'alimentation de la table MNG (information de management).

Sur le plan de contrôle, le Membre exécute la nouvelle configuration pour la création de tunnel avec de nouveaux paramètres.

2.4 Interface de supervision

La figure 5.2.4 représente le diagramme du cas d'utilisation « Interface de supervision ».

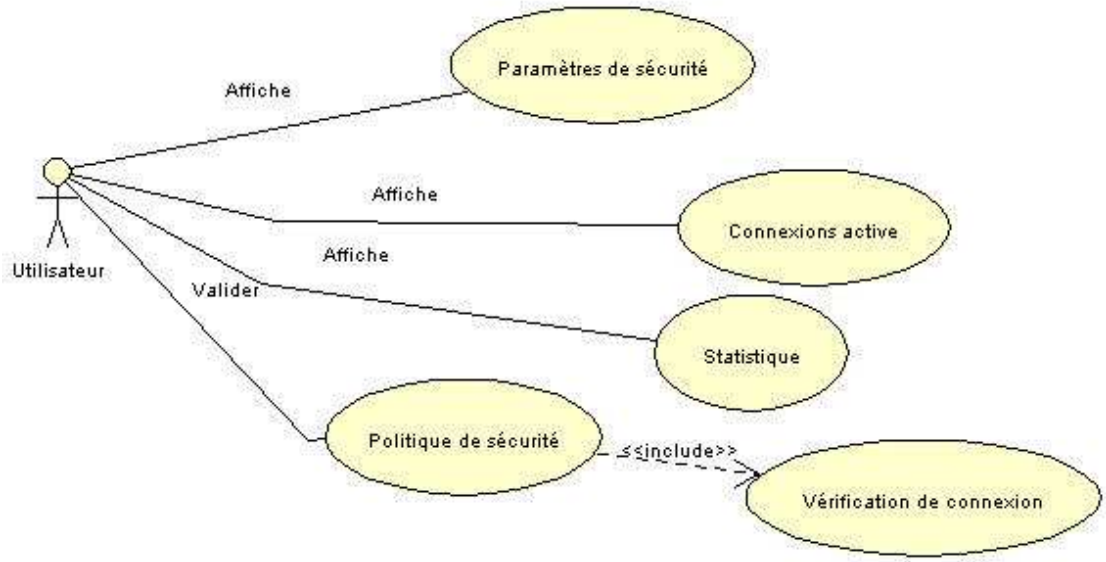


Figure 5.2.4 Cas d'utilisation Interface de supervision

L'interface de supervision permet à un utilisateur d'afficher : les clés globales (clés AH et ESP), les connections actives, les numéros SPI (Security Parameter Index), et de réaliser les configurations effectives.

3. Diagrammes de séquence

3.1 Configuration du tunnel

La figure 5.3.1 représente un scénario de configuration de tunnel entre deux pairs.

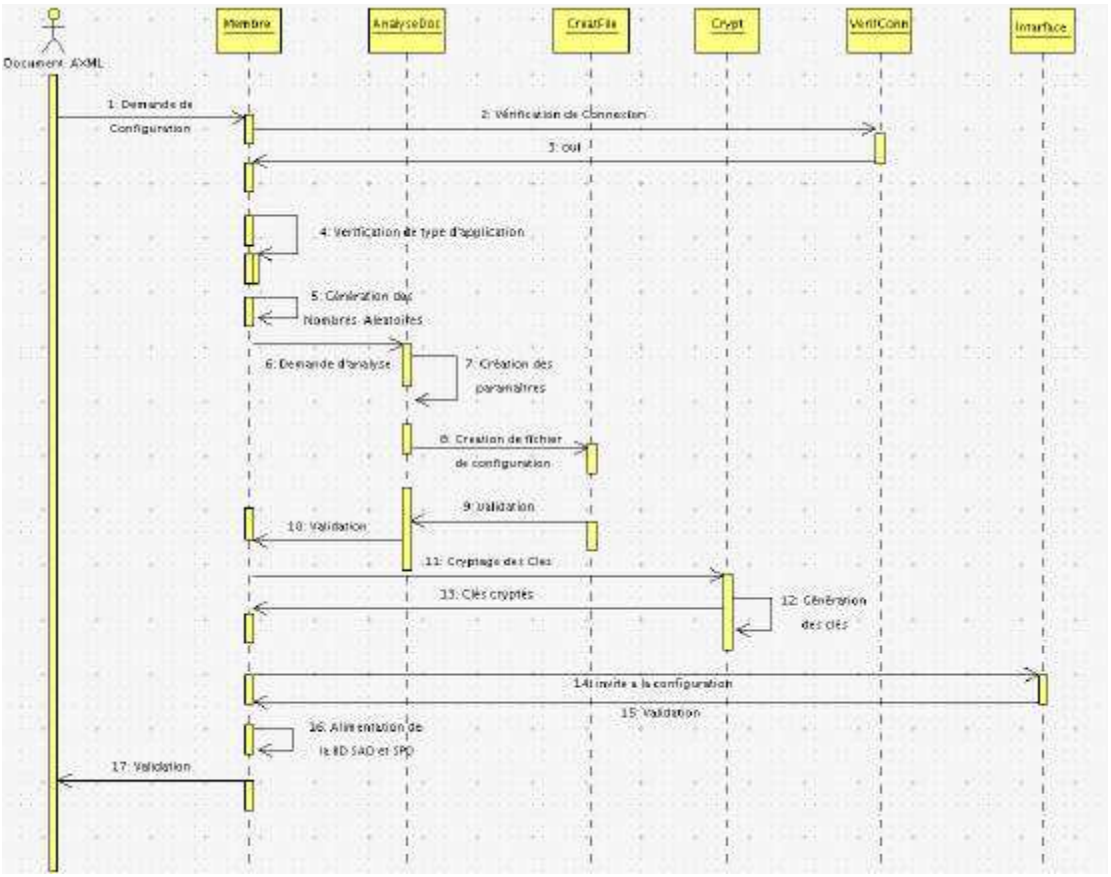


Figure 5.3.1 scénario de configuration

Cette figure illustre les processus de configurations suivant la stratégie de sécurité décrite dans les fichiers de configuration.

Une activation périodique de configuration est assurée par le Document AXML. Après une vérification de connexion, un message est envoyé à la classe AnalyseDoc pour la création de fichier de configuration (Fich de configuration, annexe 9) selon la politique décrite dans un document XML(Doc XML, annexe 8). La classe Membre invite l'autre bout de tunnel à effectuer la

configuration. A la fin de la session et après réception d'une validation de la classe Interface, les deux pairs se mettent à alimenter les bases de données SAD et SPD.

3.2 Consultation des SAD

La figure 5.3.2 représente un scénario de la consultation de la base de données SAD de l'ensemble des éléments réseau.

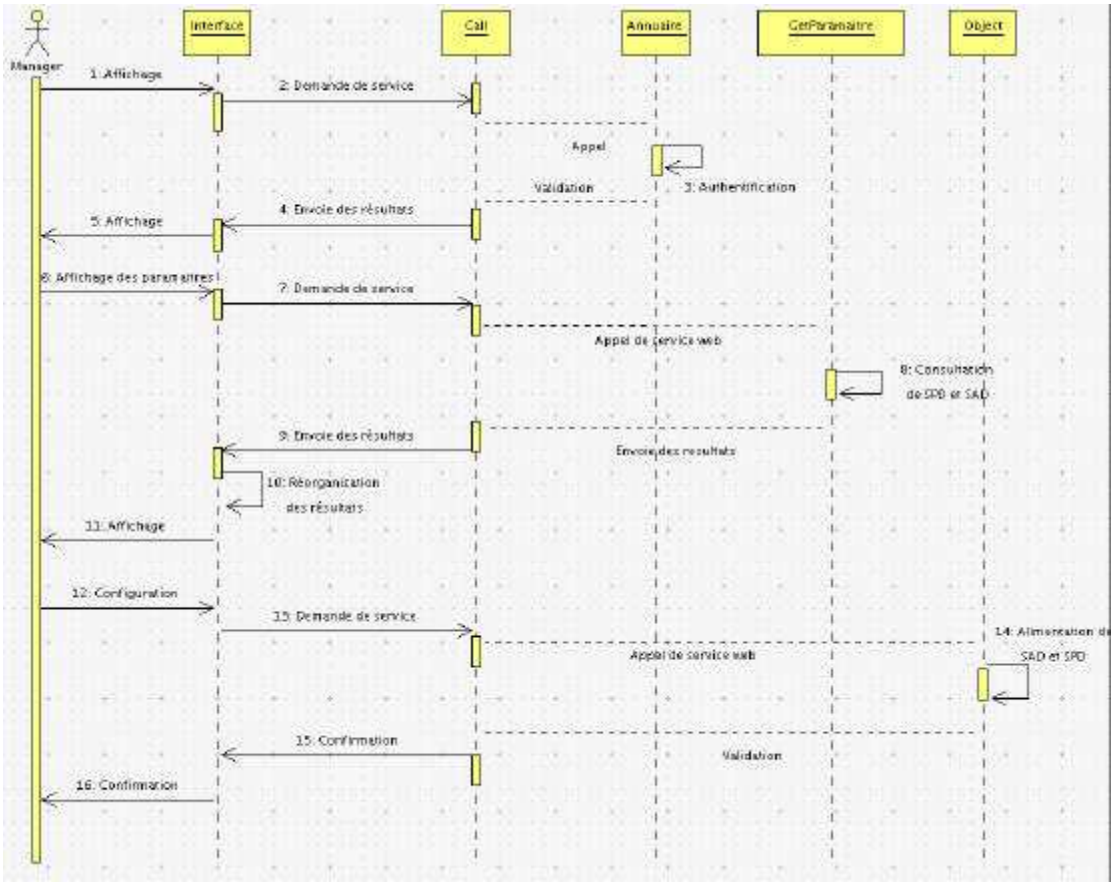


Figure 5.3.2 scénario de consultation des SAD

Après l'authentification de manager au niveau de l'annuaire, ce dernier transmettra toutes les adresses IP des éléments réseaux.

A la réception de la validation, Le manager peut lancé la collecte d'information concernant un tunnel donnés ou l'activation des configurations sur n'importe quel élément du réseau.

4. Conclusion

Dans ce chapitre, nous avons présenté les modèles conceptuels servant à l'implantation de notre application. Dans le chapitre suivant, nous décrirons cette implémentation tout en donnant des exemples des interfaces IHM.

Chapitre 6 : Réalisation

Après avoir mené l'étude du projet et de ses différentes parties fonctionnelles et conceptuelles, nous entamons la phase de réalisation. Le premier paragraphe de ce chapitre sera consacré à la présentation de la plate-forme d'implémentation. Dans le second, nous dévoilons les outils de gestion offerts aux utilisateurs du système.

1 Implémentation

Le prototype que nous avons réalisé fonctionne sous linux 2.6. Il est développé en Java et utilise le framework Active XML avec la plate forme de Web Services Tomcat Axis.

Afin de mettre en oeuvre les Services Web voulus, nous avons opté pour une solution en open source basée sur le couple Tomcat comme moteur de servlet et sur Axis comme interface de Web Service.

Tomcat est l'implémentation de référence choisie par Sun en tant que moteur de Servlet pour la spécificationJ2EE.

Tomcat permet de mettre en oeuvre des Servlets et des JSP (applications côté serveur) c'est à dire fournit une interface de programmation ainsi qu'un serveur permettant la réalisation d'applications Web à contenu dynamique ainsi que leur déploiement et leur exploitation.

Dans le cadre de création de Services Web , l'utilisation de Tomcat semble assez évidente puisque l'on souhaite créer des services dont le contenu correspond à un format Web, lequel est géré de manière puissante et performante par le moteur de servlet. Ainsi Tomcat permet la gestion des Services Web ainsi que leur relais vers Axis.

Axis va quant à lui se comporter comme une servlet standard dans Tomcat à ceci près qu'il va permettre de répondre à des requêtes soumises sous forme de service Web c'est à dire écrite en XML avec SOAP.

Axis est une implémentation sur la plate forme JAVA des spécifications de la norme SOAP 1.1, et d'une partie des fonctionnalités de la nouvelle version de la norme SOAP 1.2. AXIS est développé par la fondation Apache (Apache Software Fondation).

Apache AXIS est à la fois un environnement d'hébergement de Services Web et un toolkit de développement pour la création ainsi que l'accès client à des services tiers. ce toolkit comporte des fonctionnalité avancées de génération de code automatique basées sur l'analyse de descriptions

WSDL de Services Web et le mapping automatique à double sens entre classe java et description WSDL

Nous avons l'architecture suivante entre Tomcat et Axis :

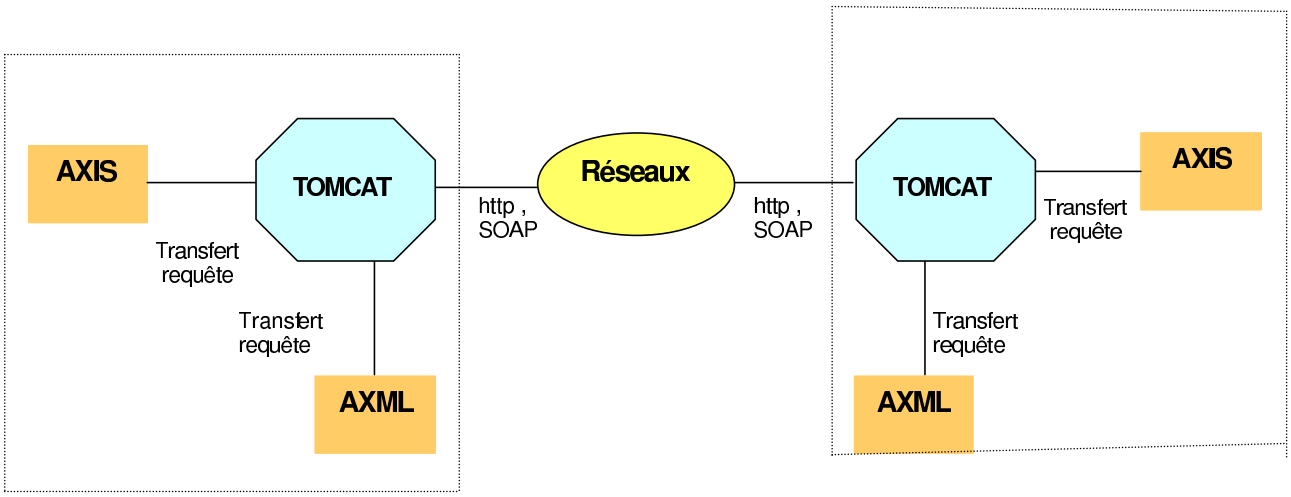


Figure 6.1.1 : Schéma général de fonctionnement de TOMCAT-AXIS

Après la avons description de l'implémentation TOMCAT – AXIS, nous allons présenter dans ce qui suit le travail que nous effectué pour l'implémentation de notre approche.

2. Mise en oeuvre de la plate forme

Durant la réalisation de notre projet, nous étions amenés, d'une part, à assurer les fonctionnalités de notre application, et d'autres parts, à gérer la phase d'alimentation des SAP et SPD en tenant compte des mises à jour des paramètres de sécurité.

2.1 Schéma général de fonctionnement

Notre approche fonctionne de la manière suivante : les pairs AXML représentant chacun un élément réseaux possèdent au départ un document de définition de Services Web qu'ils exposent (Document de def .WS,cf Annexe 5), ainsi que deux types de documents de managements. Le premier assure le stockage d'information sur la politique de sécurité pour chaque connexion (Document AXML,cf Annexe 6), alors que le deuxième (Document AXML, cf Annexe 7) joue deux rôles principaux :

- Il impose la politique de sécurité auprès de son partenaire grâce à un appel à des Services Web AXML (Document de def .WS,cf Annexe 5).

- Il met à jour la configuration d'IPSec.

Le principe général est représenté dans la figure 5.1.1, en effet, d'après la politique d'échange que nous avons définie, les pairs AXML s'appellent entre eux périodiquement pour s'échanger leurs politiques de sécurités respectives grâce à des appels à des Services Web.

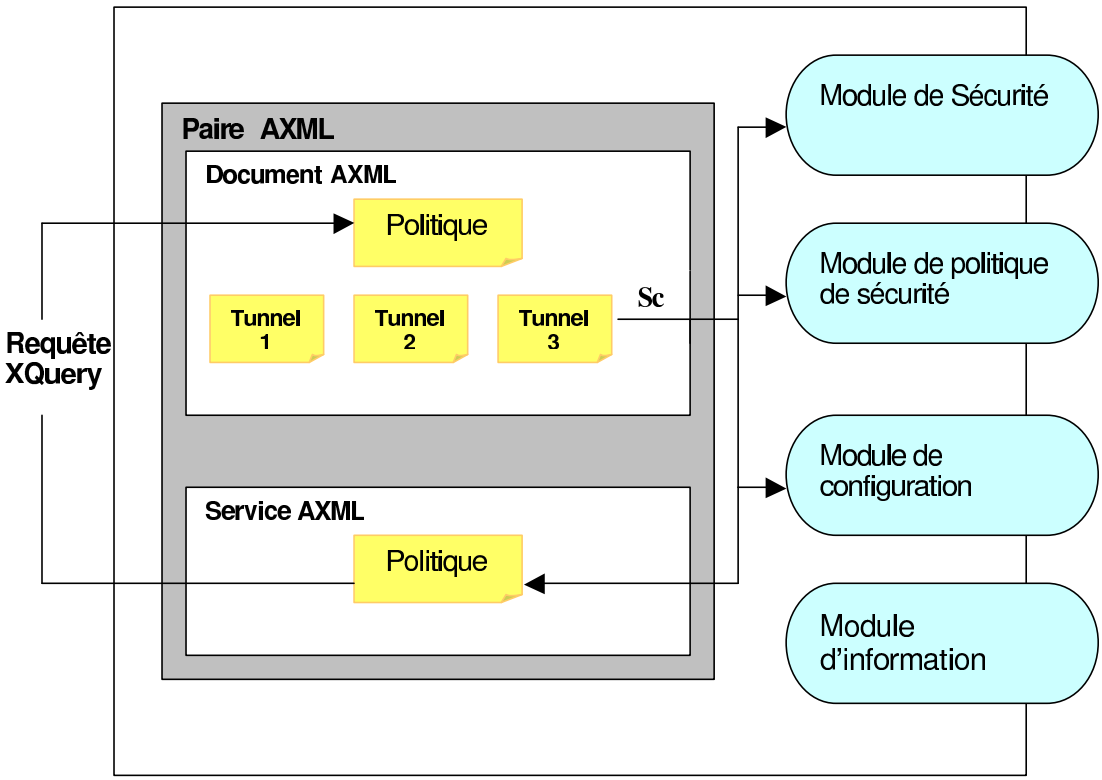


Figure 6.1.1 : Organisation des documents AXML

Après avoir présenté une vue générale de notre approche, nous entamons la mise en œuvre de chaque module :

2.1 Implémentation du module de sécurité.

Ce module est responsable de l'exécution des fonctions de sécurité. Il assure les services suivants :

- **Publication des clés publiques** : qui a en charge de distribuer les clés générées à l'aide de la classe **KeyPairGenerator du package java.security[6],**
- **Cryptage et décryptage :** responsable du cryptage décryptage du trafic et de la génération des clés à partir de l'objet KeyAgreement enregistré dans le fichier security.conf,
- **Communication :** son rôle est d'assurer la communication entre les différents services de sécurités déployés sur les éléments réseau à l'aide de la classe **Call** (Classe CALL, annexe 6).

Notons que les messages sont échangés sous format de chaîne de caractère **(String).** Pour assurer la conversion de certains paramètres on utilise la classe **HexString** (Class **HexString** . Annexe 7) qui réalise la conversion Byte – String, hexa – String, etc…

2.2 Implémentation du module de politique de sécurité

Comme précédemment cité, la collecte d'information concernant la politique de sécurité est assurée par un service Active XML. Il reste à faire négocier cette politique entre les entités concernées. Pour cela ce module adopte les services suivants :

- **Communication** : qui permet de négocier les paramètres de sécurité à déployer.
- **Analyse de ces politiques de sécurité** : permet de générer une politique selon le niveau de sécurité le plus élevé.

Pour stocker les informations sur la politique de sécurité, nous avons créé un document spécifique AXML pour chaque pair. Ce dernier contient les informations concernant chaque connexion.
Notons que les niveaux de sécurité sont classés selon la puissance des algorithmes de cryptages utilisés.

2.3 Implémentation du module de configuration

Comme précédemment cité, ce module est responsable de l'alimentation de la base de données SAD et SPD.
La mise en œuvre de ce module est assurée par les fonctions suivantes :

- la vérification de la connexion sur le port 8080 à l'aide de la classe **VerifConf**, cette fonction est assurée par l'utilisation de la classe **SocketChannel (Package java.net)[6],**
- Récupération de l'adresse IP de la machine locale,
- Collecter les informations concernant le type d'application de gestion des clés, la politique de sécurité grâce au document XML (Doc AXML. Annexe 8) générées par le module de politique de sécurité,
- La génération et la distribution des clés selon l'algorithme de cryptage utilisé (la génération des nombres aléatoires est réalisée par la classe **Math.random[6]).**
- Création et exécution du fichier de configuration (Doc de conf Annexe 9) pour l'alimentation de la base de données des SAD et SPD.

Pour collecter les informations de politiques de sécurités, nous avons créé un fichier XML (Doc AXML. Annexe 8) par connexion. Ce dernier sera analysé par la classe **DocAnalyse[6]** qui utilise le **package org.xml.sax[15] .**

Toutes les informations concernant l'application de gestion de clés sont stockées dans le fichier **Swan.conf** qui se trouve dans le répertoire d'installation.

Notons que l'alimentation de base de données SAD, est réalisée par l'exécution automatique de fichiers de configuration (Fich de configuration annexe 9) créés par la classe **CreateFile.**

2.4 Implémentation du module d'information

Ce module fournit les fonctionnalités de supervision. Il assure les services suivants :

- La consultation des bases de données SAD et SPD,
- La mise à jour des fichiers de configuration.

Notons que les services de ce module peuvent être invoqués à l'aide des interfaces graghiques.

2.5 Lancement de l'application

A partir du n'importe quel poste qui possède l'implémentation de IP Sec, le lancement de l'application pourra se faire à partir d'une icône placée sur le «Bureau ». L'utilisateur sera alors invité à saisir le mot de passe lui permettant l'installation de tous les modules de l'application.

Nous présentons ci-dessous l'interface de lancement. L'Interface Homme Machine (IHM) est développée avec le langage Java.

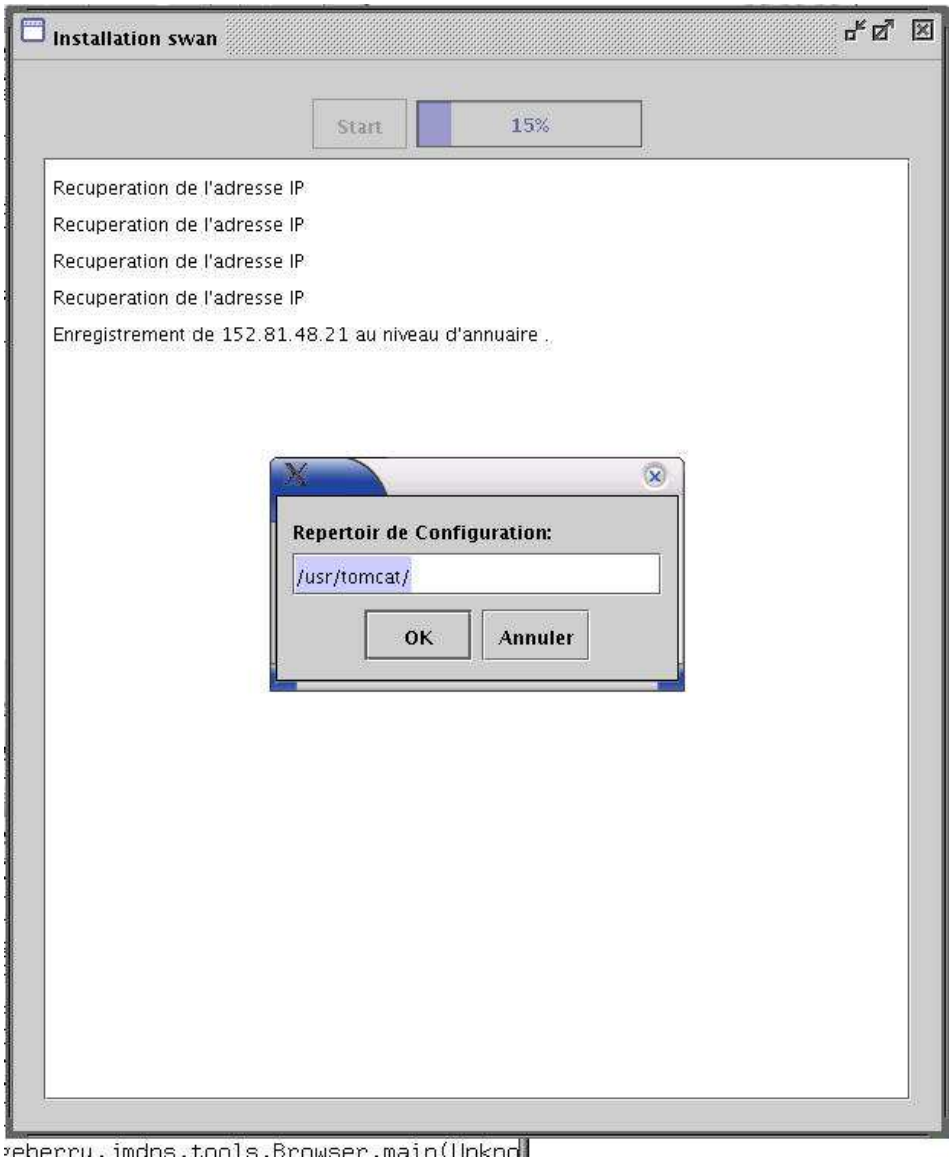


Figure 6.1 : Installation de l'application

Cette interface « **figure 6.1** » permet de réaliser les principales fonctionnalités pour l'activation de l'application. Ces dernières sont regroupées de la façon suivante :

- **Consultation de l'annuaire :** permet la consultation des informations concernant l'architecture du réseau ainsi que de la politique de sécurité requise pour chaque élément du réseau. Ces informations servent à la création des fichiers AXML (Doc AXML. Annexe 2) pour l'activation ultérieure des tunnels.

- **Création des fichiers de configuration :** permet la création des fichiers de configuration suivants :

- **Politique.xml** : ce fichier rassemble la description de la politique de sécurité sous format XML (Doc AXML. Annexe 4) pour chaque connexion réseau.

- **Swan.conf** : Ce fichier contient les informations concernant le type l'application de gestion des clés et le répertoire d'installation pour chaque connexion réseau.

- **Déploiement des services Web** : permet de copier dans le répertoire approprié le package alpha.jar (rassemble les classes qui représentent les services Web) afin de réaliser leurs fonctionnalités.

- **Mise en marche du serveur d'application :** Un redémarrage du serveur d'application est nécessaire, pour qu'il puisse prendre en considération les mises à jour au niveau de framework Active XML.

A la fin de cette étape une activation périodique des politiques et associations de sécurité est assurée grâce aux documents Active XML (Doc AXML. Annexe 2). Pour prendre en considération les mises à jour concernant la politique de sécurité ou l'application de gestion des clés utilisées, on peut utiliser des interfaces graphiques traitées dans le paragraphe suivant.

2.5 Menu principal

Nous présentons ci-dessous l'interface de menu général. L'Interface Homme Machine (IHM) est développée avec le package Swing de Java [6] qui propose les éléments de base de l'ergonomie traditionnelle des boîtes de dialogue (boutons, zones de texte, images, listes « déroulantes », etc...).

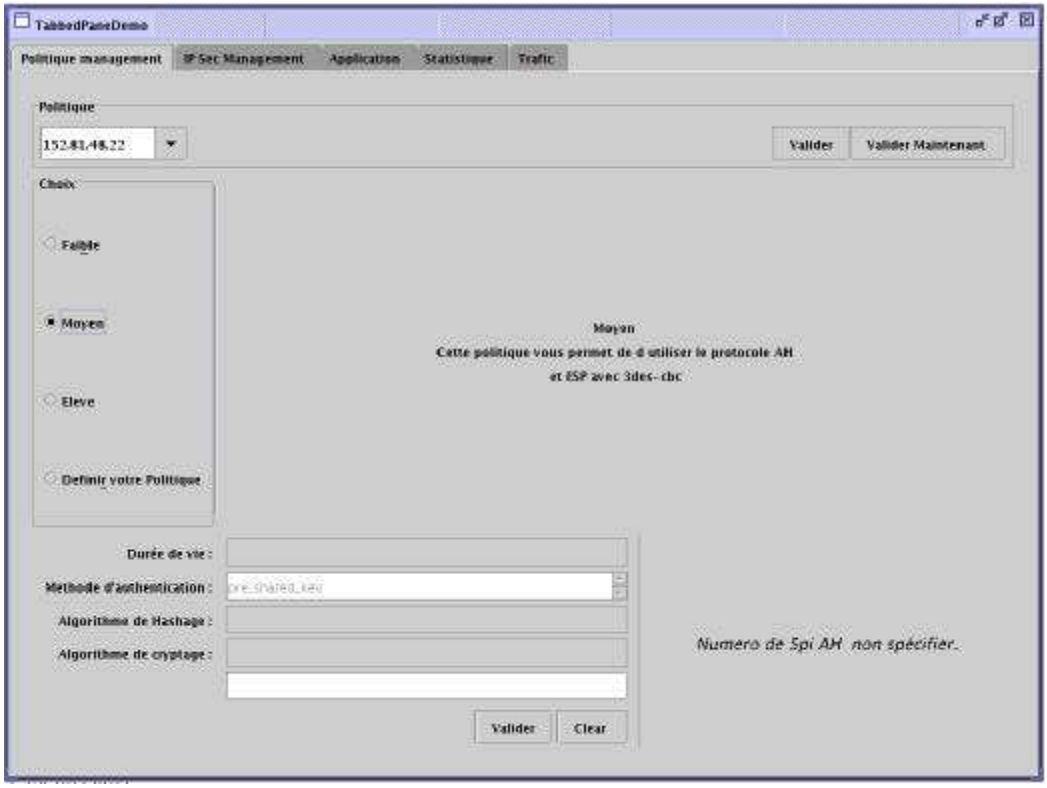


Figure 6.2 : Menu principal de l'application

Ce menu principal permet la supervision de l'ensemble des éléments réseau. Il donne l'accès à cinq fonctionnalités principales de l'application. Ces dernières sont ainsi accessibles à l'utilisateur :

1. **Module Politique Management :** permet la mise à jour de la politique de sécurité.
2. **Module IP Sec Management** : dédié à l'édition des informations concernant les tunnels (clés, algorithme de cryptage etc…) .
3. **Module Application** : permet de choisir entre les différentes implémentations des échanges de clés.
4. **Module Statistique** : effectue des tests de performances concernant l'échange des clés.
5. **Trafic** : permet de filtrer les paquets pour une connexion choisie.

2.5.1 Module Politique Management :

Un exemple de capture d'écran pour ce module est donné dans la figure 6.3 :

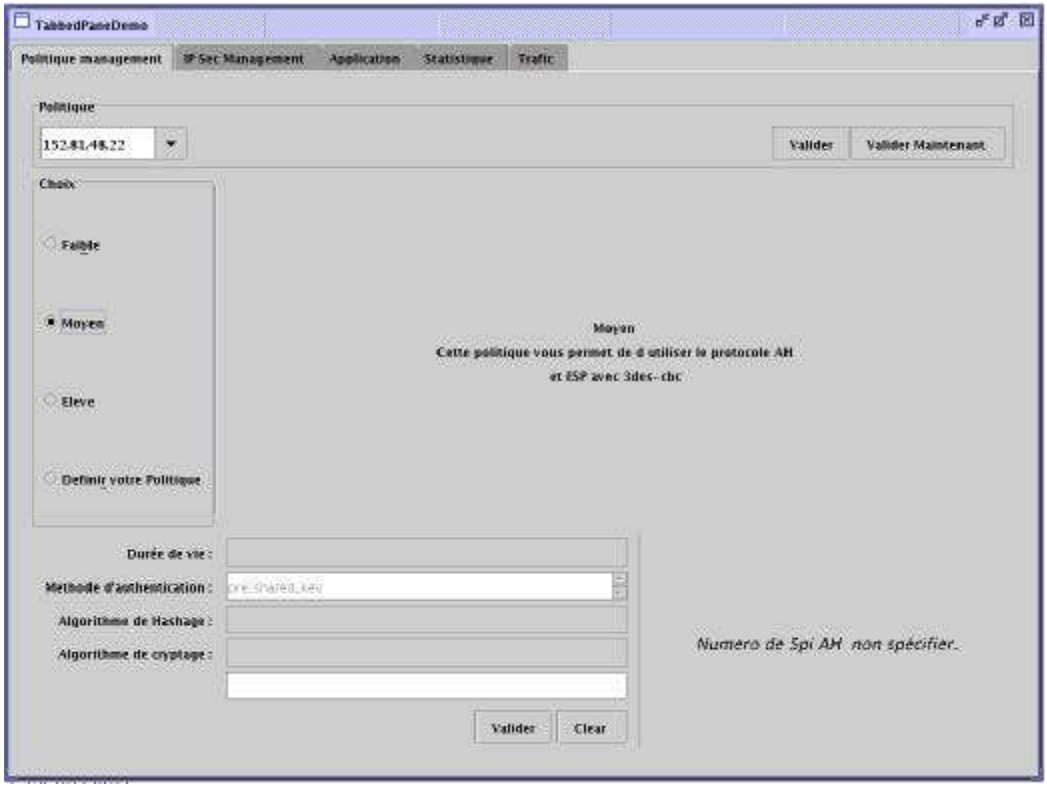


Figure 6.3 : Ecran du module de politique management.

Le fonctionnement comprendra pour chaque élément du réseau représenté par son adresse IP :

- Le choix de déploiement entre trois niveaux de sécurité Faible, moyen et Fort.
- Editer la description de chaque niveau.
- La possibilité de définir son propre mode de sécurité (grâce aux différents mécanismes offerts par l'implémentation d'IP Sec) et l'enregistrer pour pouvoir l'utiliser une prochaine fois .

2.5.2 Interface du Module IP Sec Management :

Nous présentons ci-dessous une capture d'écran de l'interface utilisateur permettant l'édition et l'activation des paramètres concernant un tunnel.

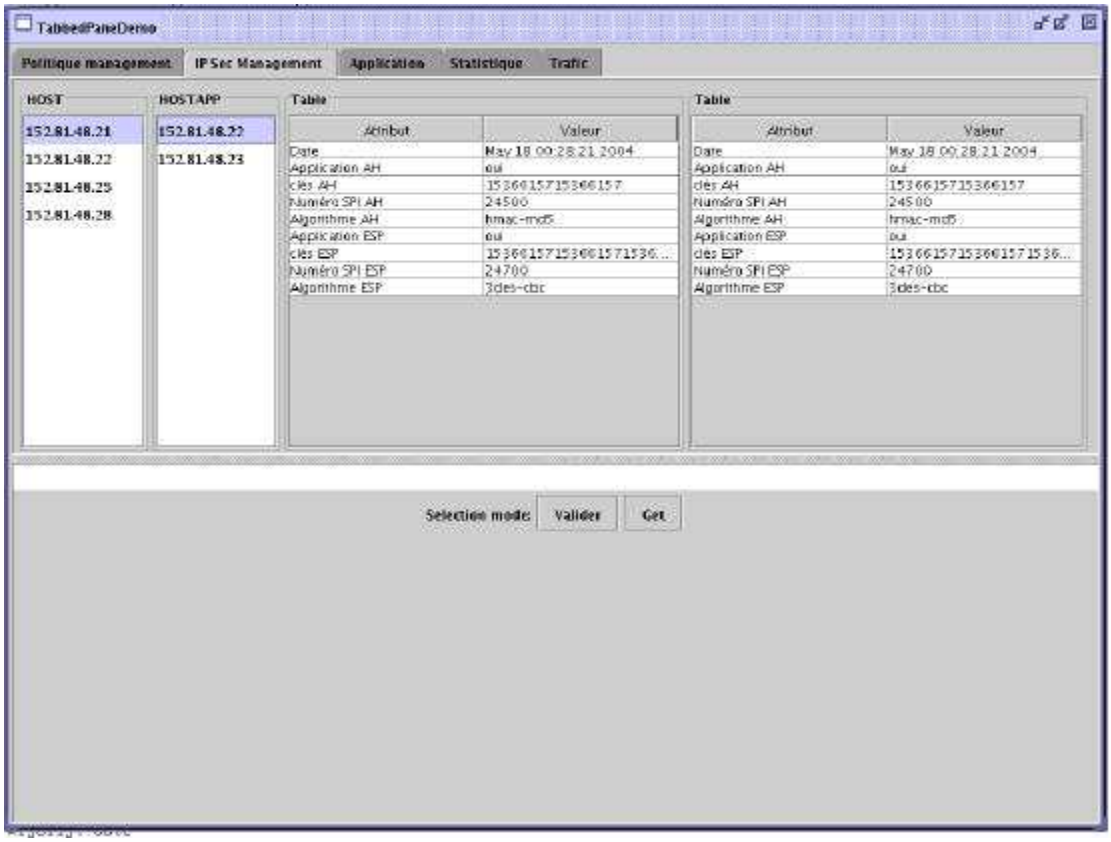


Figure 6.3 : Ecran du module d'IP Sec management.

L'édition comprendra pour chaque connexion :

- Les algorithmes de cryptages utilisés,
- Le numéro de SPI,
- La date d'activation des clés,
- Avec la possibilité de changer ses paramètres et de les activer.

L'extraction des données est réalisée par la classe CALL (Classe CALL, annexe 6) correspondantes.

Une fois que l'utilisateur a spécifié les informations concernant une connexion choisie, les services Web correspondants lancent une requête d'extraction des données, les mettent en forme, puis les affichent sous forme de tableaux.

5.2.3 Module Application

Nous présentons ci-dessous une capture d'écran de l'interface utilisateur permettant la mise à jour de l'application de gestion de clés.

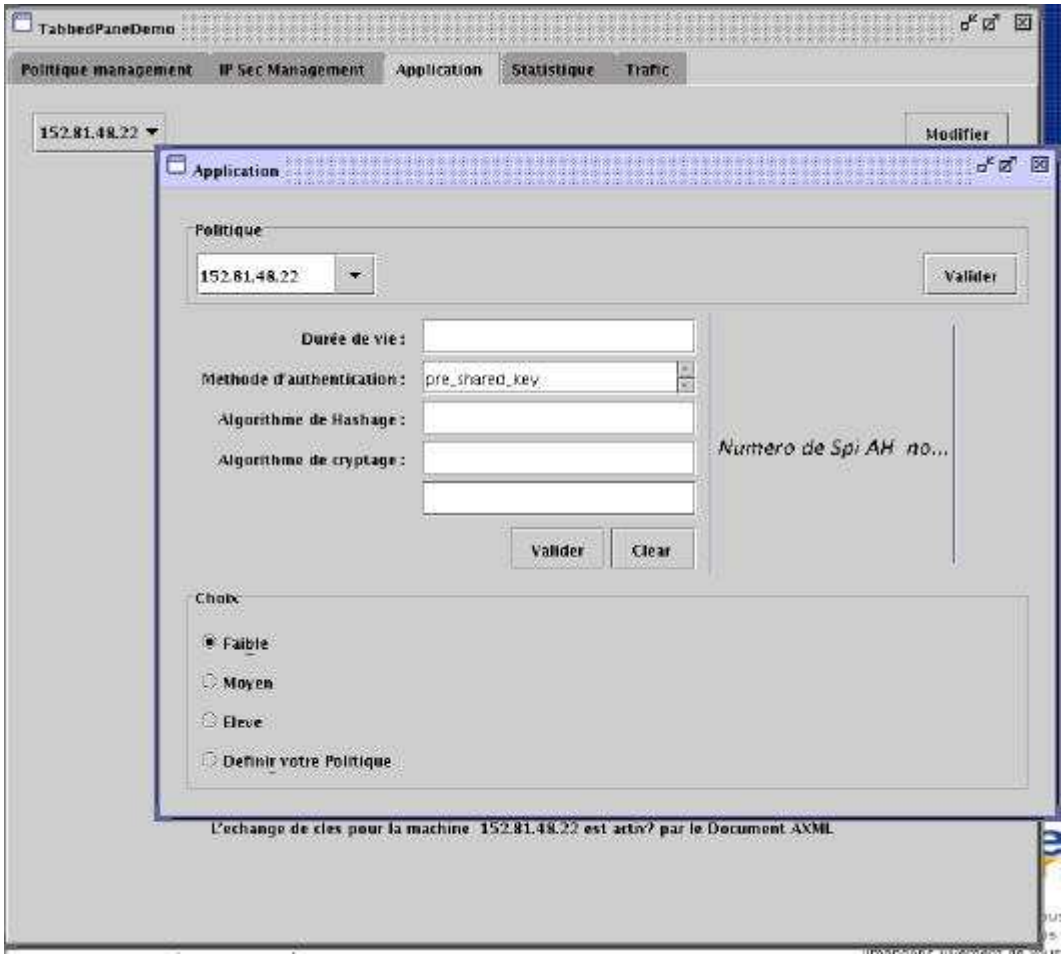


Figure 6.4 : Ecran du module application.

Les paramètres cités ci-dessus à la figure 6.4, donnent la possibilité à l'administrateur de choisir l'application de gestion de clés, les paramètres qui seront activés par le Document Active XML (Document AXML, cf Annexe 7). Il peut aussi choisir la durée d'activation des clés et la méthode d'authentification (clés partagées ou certificat).

2.5.4 Module statistique :

Un exemple de maquette pour ce module est donné dans la figure 6.5 :

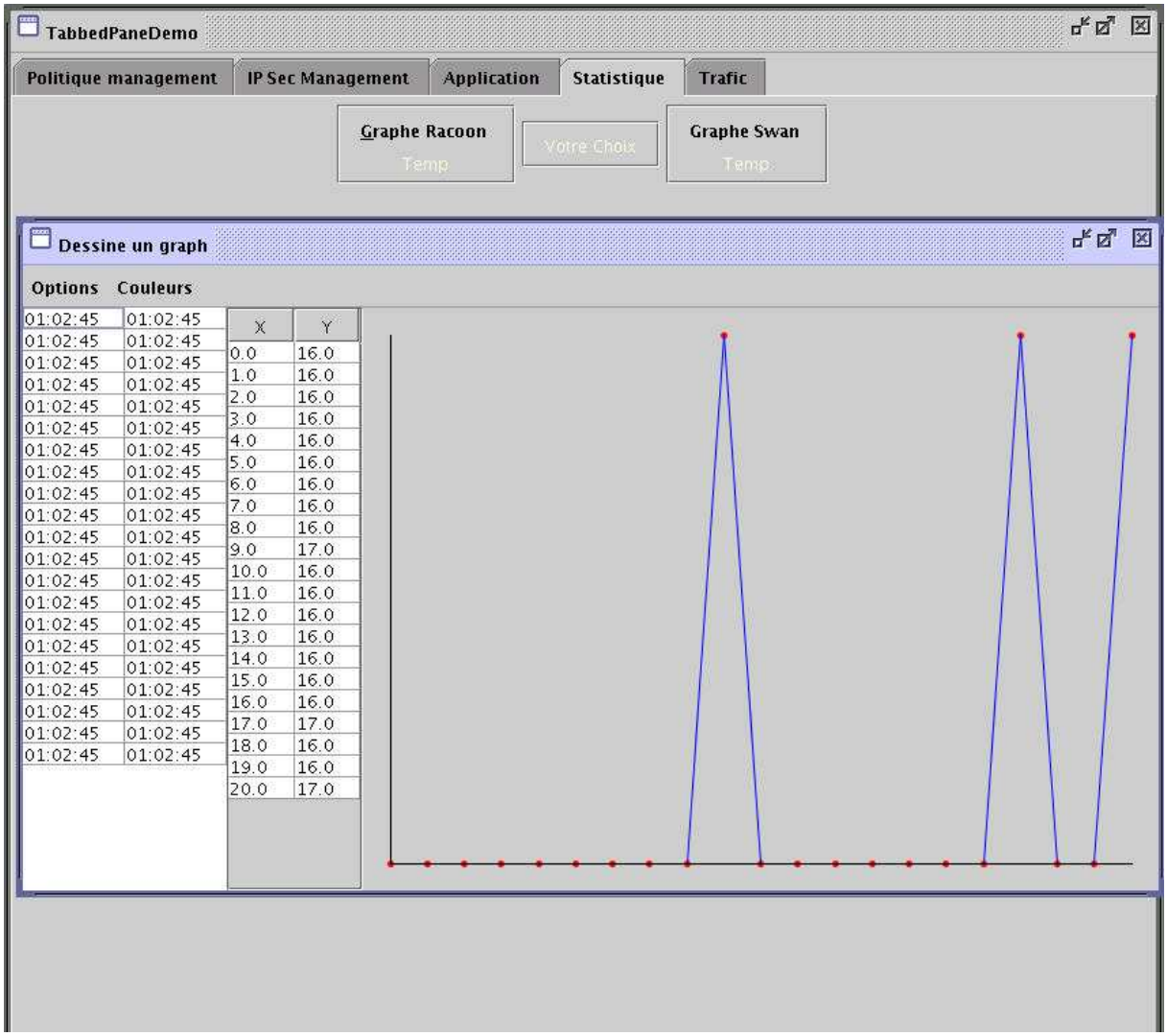


Figure 6.5 : Ecran du module statistique.

L'interface de la figure 6.5 est celle qui affiche les informations concernant les dates d'activation des clés (voir chapitre 3 paragraphe 2). Le diagramme de la figure concerne la différence entre les dates d'activation des clés entre deux machines. Ces informations sont tirées directement d'un fichier conf.log. Ce fichier représente l'évolution de la base de données SAD dans le temps.

3 Conclusion

Cette étape d'intégration était la phase finale du projet. Au cours de ce chapitre nous avons essayé de décrire les interfaces de supervisions les plus importantes de l'application.

Conclusion et perspective

Au cours de ce projet, nous avons implémenté une plate forme de gestion automatique de configuration d'IPsec. Les principales fonctions de gestions telles que la sécurité et l'autonomie ont été réalisées en utilisant une architecture pair à pair basée sur Active XML.

Nous avons aussi réalisé des interfaces graphiques pour la supervision qui permettent d'utiliser et de suivre l'évolution des configurations et donc de montrer l'efficacité de cette approche pour les fonctions de configuration.

Les connaissances acquises au cours de ce stage sont multiples, je cite la mise en œuvre des Services Web, la configuration et les problèmes liés à sa supervision, la familiarisation avec les outils de configurations d' IP Sec sans oublier la réalisation des interfaces graphiques.

En outre, j'ai eu la chance d'apprendre à manipuler le framework Active XML et de concevoir et réaliser des modules de gestion dans cette plate-forme.

En perspective, nous espérons que ce travail sera complété par la réalisation d'un module qui permet la manipulation directe des bases de données SAD et SPD, et par l'intégration de nouvelles fonctionnalités telles que la configuration dynamique de la qualité de service (QoS) et des firewalls dans un ensemble cohérent.

Bibliographie

[1] http://lartc.org/howto/index.html.

[2] S Abiteboul,O.Benjelloun ,I .Manolescu T.Milo and R.Weber Active xml :A data centric perspective on Services Web .Evry france http://www-rocq.inria.fr/cerso/Gemo/Projects/axml.

[5] Linux Free/Swan : http://www.freeswan.org

[4] Pierre Hambert, "Impact des services web sur les architectures de supervision de réseaux ", mémoire pour l'obtention du DEA de l'Université Henri Poincarre Nancy I .

[3] Mémoire Impact des modèles Services Web sur les architecture de supervision de réseaux. Pierre HUMBERT.

[4] (VPN)Virtual Private Network http://194.51.152.252/VPN/

[6] JavaTM 2 Platform, Standard Edition, v 1.4.2 API Specification : http://java.sun.com/j2se/1.4.2/docs/api/.

[7] IP Sec : http://www.securiteinfo.com/crypto/IPSec.shtml

[9] XQuery 1.0 : http://www.w3.org/TR/2004/WD-xquery-semantics-20040220/#sec_union_interpretation.

[10] Swing books http://java.sun.com/docs/books/uiswing/.

[12] G. Kakivaya D. Box, D. Ethenbuske and A. Layman

[14] UML en action par Pascal Roques, Franck Vallée ISBN 2-212-09127-3 Editions Eyrolles.

[15] http://www.saxproject.org/apidoc/org/xml/sax/package-summary.html.

Glossaire

Apache Axis : Implémentation Java de la spécification SOAP . Intégrée à un serveur d'application cette plate forme permet le déploiement de services Web .

API (Application Programming Interface) : Une API contient un ensemble de fonctions qui facilitent la programmation , un jeu de fonctions et de méthodes , utilisé pour accéder à certaines fonctionnalités .

Package : C'est un ensemble de classes, regroupées suivant leurs fonctionnalités. Par exemple "java.io" regroupe toutes les classes relatives aux entrées-sorties, "java.net" aux connexions réseaux ...

DES (Data Encryption Standard) : Algorithme de chiffrement basé sur la technique de cryptographie symétrique publié par IBM et adopté par le département de la défense américaine en 1977. DES repose sur une clé symétrique de 56 bits et effectue un chiffrement par blocs de 64 bits.
CORBA (Common Object Request Broker Architecture) : Architecture Standardisée de Négociateurs de Requêtes d'Objets. Document de l'OMG (Object Management Group) spécifiant le fonctionnement d'un ORB, la syntaxe du langage IDL, l'interopérabilité des ORB et l'adaptation à différents langages.

L2TP : Layer 2 Tunneling Protocol. VPN de niveau 2 (couche liaison du système OSI) basé sur les propositions PPTP et L2F (RFC 2341 obsolète), L2TP permet l'établissement de tunnels de bout en bout et assure les services de sécurité suivants :
- Authentification (CHAP, PAP).
- Confidentialité (chiffrement par secret partagé, ou par clé publique en utilisant l'algorithme RC4 à 40 ou 128 bits).

PKI : La Public Key Infrastructure permet de sécuriser de façon globale l'accès à un réseau, à des informations et données. La PKI est utilisée dans les domaines suivants : E-mail, e-commerce, VPNs (réseau privé virtuels), Extranets.

Tunneling : le principe de tunneling consiste à utiliser un réseau public non sécurisé (ex. : Internet) comme élément/extension d'un réseau privé sécurisé. en ajoutant une couche de sécurité.

VPN (Virtual Private Network) [Réseau Privé Virtuel] : Un VPN est un réseau de données qui utilise les moyens de télécommunications d'un réseau public en ajoutant des services de sécurité et des protocoles de tunneling .

SAX (Simple API for XML) : API pour traiter séquentiellement un document XML en utilisant des événements .

Swing : Framework pour le développement d'interfaces graphiques composé de composants légers.

ISA/ KMP (Internet Security Association, Key Mgt. Protocol) [association pour la sécurité d'Internet, protocole de gestion de clés]
Définit les procédures pour l'authentification des deux parties d'une communication, la création et la gestion d'associations de sécurité [Security Associations], les techniques de génération de clés, et l'atténuation des menaces, par exemple le blocage de service et les attaques par rejeu.

IETF (Internet Engineering Task Force) [groupe spécial d'ingénierie d'Internet]
Grande communauté internationale ouverte de concepteurs de réseaux, d'opérateurs, de vendeurs et de chercheurs intéressés par l'évolution de l'architecture et le fonctionnement sans heurt d'Internet. Elle est ouverte à tout individu intéressé.

PGP (Pretty Good Privacy) [Assez bonne confidentialité]
Application et protocole (RFC 1991) pour le courrier électronique sécurisé et le chiffrement de fichiers développé par Phil R. Zimmermann. Diffusé gratuitement à l'origine, le code source a toujours été disponible et inspecté.

SSH (Site Security Handbook) [livret de sécurité de site]
Le groupe de travail (WG) de l'IETF a planché depuis 1994 pour produire deux documents destinés à éduquer la communauté Internet sur la sécurité.

SSL (Secure Socket Layer) [Couche de sockets sûres]
Développé par Netscape pour fournir sécurité et confidentialité sur Internet. Supporte l'authentification du serveur et du client et assure la sécurité et l'intégrité du canal de transmission.

SST (Secure Transaction Technology) [technologie de transactions sûres]
Protocole de paiement sûr développé par Microsoft et Visa comme "compagnon" du protocole PCT.

S/ WAN (Secure Wide Area Network) [réseau géographiquement étendu sûr]
Spécifications de mise en oeuvre d'IPsec lancées par RSA Data Security, Inc., pour assurer l'interopérabilité des gardes- barrières et produits TCP/ IP.

TLS ([Transport Layer Security) [sécurité de la couche transport]
Brouillon IETF, la version 1 est basée sur la version 3.0 du protocole SSL, et assure la confidentialité des communications sur Internet.

XQuery : Xquery est un langage de requêtes sur un document XML (à la manière de SQL pour interroger des bases de données). Il permet, par exemple, de rechercher simplement tous les éléments ayant un nom donné.

Annexe

Annexe 1

Exemple de configuration d' IP Sec

(Prototype)

Ce qui suit est une configuration très simple permettant le dialogue de l'hôte 152.81.48.21 vers l'hôte 152.81.48.22 en utilisant l'encryptage et l'authentification. Notez que le trafic de retour de cette première version est en clair et que cette configuration ne doit pas être déployée pour des raisons de sécurité.

Sur l'hôte 152.81.48.21 :

```
#/usr/local/sbin setkey –f     #ici en précise l'emplacement de la commande setkey

add 152.81.48.21 152.81.48.22 ah 15700 -A hmac-md5 "1234567890123456";

add 152.81.48.21 152.81.48.22 esp 15701 -E 3des-cbc    "123456789012123456789012";

spdadd 152.81.48.21 251.81.48.22 any –P out ipsec

    esp/transport//require

    ah/transport//require;
```

Avec la mise en place de la configuration ci-dessus (ces fichiers peuvent être exécutés si 'setkey' est installé dans /usr/local/sbin), la commande ping 152.81.48.22 exécutée sur 152.81.48.22 va donner la sortie suivante avec tcpdump :

```
22:37:52 152.81.48.21 > 152.81.48.22: AH(spi=0x00005fb4,seq=0xa):
ESP(spi=0x00005fb5,seq=0xa) (DF)

22:37:52 152.81.48.22 > 152.81.48.21: icmp: echo reply
```

Notez que le paquet de retour provenant de la machine 152.81.48.22 est en effet complètement visible. Le paquet ping émis par la machine 152.81.48.21 ne peut évidemment pas être lu par tcpdump, mais celui-ci montre l'Index du Paramètre de Sécurité (SPI) de l'AH, ainsi que l'ESP, qui indique à la machine 152.81.48.22 comment vérifier l'authenticité de notre paquet et comment le décrypter.

Quelques éléments doivent être mentionnés. La configuration ci-dessus est proposée dans de nombreux exemples d'IPsec, mais elle est très dangereuse. Le problème est qu'elle contient la politique indiquant à la machine 152.81.48.21 comment traiter les paquets allant vers la machine 152.81.48.22 et comment la machine 152.81.48.22doit traiter œs paquets, mais ceci n'INDIQUE pas à la machine 152.81.48.22 de rejeter le trafic non authentifié et non encrypté .

N'importe qui peut maintenant insérer des données "spoofées" (NdT : usurpées) et entièrement non cryptées que 152.81.48.22 acceptera. Pour remédier à ceci, nous devons avoir sur 152.81.48.22 une Politique de Sécurité pour le trafic entrant :

```
#!/usr/local/sbin/setkey -f

spdadd 152.81.48.2 152.81.48.22 any -P IN ipsec

   esp/transport//require

   ah/transport//require;
```

Ceci indique à 152.81.48.22 que tout le trafic venant de 152.81.48.21 nécessite d'avoir un ESP et AH valide.

Maintenant, pour compléter cette configuration, nous devons également renvoyer un trafic encrypté et authentifié. La configuration complète sur 152.81.48.21 est la suivante :

```
#!/usr/local/sbin/setkey -f

flush;

spdflush;



# AH

add 152.81.48.22 152.81.48.21 ah 15700 -A hmac-md5 "1234567890123456";

add 152.81.48.21 152.81.48.22 ah 24500 -A hmac-md5 "1234567890123456";
```

```
# ESP

add 152.81.48.22 152.81.48.21 esp 15701 -E 3des-cbc "123456789012123456789012";

add 152.81.48.21 152.81.48.22 esp 24501 -E 3des-cbc "123456789012123456789012";

spdadd 152.81.48.21 152.81.48.22 any -P out ipsec

        esp/transport//require

        ah/transport//require;

spdadd 152.81.48.22 152.81.48.21 any -P in ipsec

        esp/transport//require

        ah/transport//require;
```

Et sur 152.81.48.22 :

```
#!/sbin/setkey -f

flush;

spdflush;

# AH

add 152.81.48.22 152.81.48.21 ah 15700 -A hmac-md5 "1234567890123456";

add 152.81.48.21 152.81.48.22 ah 24500 -A hmac-md5 "1234567890123456";

# ESP

add 152.81.48.22 152.81.48.21 esp 15701 -E 3des-cbc "123456789012123456789012";

add 152.81.48.21 152.81.48.22 esp 24501 -E 3des-cbc "123456789012123456789012";

spdadd 152.81.48.22 152.81.48.21 any -P out ipsec

        esp/transport//require
```

ah/transport//require;

spdadd 152.81.48.21 152.81.48.22 any -P in ipsec

esp/transport//require

ah/transport//require;

Notez que, dans cet exemple, nous avons utilisé des clés identiques pour les deux directions du trafic. Ceci n'est cependant en aucun cas exigé.

Pour examiner la configuration que nous venons de créer, **exécuter setkey -D**, qui montre les SA **ou setkey -DP** qui montre les politiques configuré .

Annexe 2

Document AXML pour l'activation de tunnel

(Prototype)

152.81.48.22.xml

Cette annexe représente l'organisation de document Active XML cités au chapitre étude conceptuelle, sous forme de document XML.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <ipsec xmlns:axml="http://www-rocq.inria.fr/verso/AXML" xmlns:ipsec="urn:ipsec">
  - <phase>
    - <axml:sc doNesting="true" frequency="every 700000" id="9C5CD28C-7F00-0001-
        0032-3274E2D34C21" lastCalled="1084839979309" methodName="RunPhase1"
        mode="replace" serviceNameSpace="Machine2"
        serviceURL="http://localhost:8080/axml2/servlet/AxisServlet">
      - <axml:params>
        - <axml:param name="RunPhase1">
            <axml:value>152.81.48.21</axml:value>
          </axml:param>
        </axml:params>
      </axml:sc>
```

<axml:text axml:origin="9C5CD28C-7F00-0001-0032-3274E2D34C21"><clespublic>308201243081990609 2A864886F70D01030130818B02818100F488FD584E49DBCD20B49DE49107366B336C380D451D0F7C88B31C7C5B2D8EF6F3C923C043F0A55B188D8EBB558CB85D38D334FD7C175743A31D186CDE33212CB52AFF3CE1B1294018118D7C84A70A72D686C40319C807297ACA950CD9969FABD00A509B0246D3083D66A45D419F9C7CBD894B221926BAABA25EC355E92F78C7020102020203FF038185000281810 0D79A7FE0CAE610FB07C99F7805072780F5E70677CD73A1496B88DEDD4704C39D35DBB833581C1EDAD3D7C018410A074CB75330634065D3A3C196161D602E578DFE9715B14BB17D21F262B3D83791FA8200B07920E7282D27EC57F0E1A65CA532CF4F1792F33747A446D5B86C2EAFB8D67F7AD8BAA61F1963056584691EE6367A<clespublic></axml:text>

</phase>

- <politique>
  - <axml:sc doNesting="true" frequency="every 50000" id="96BB80CBB-9851-07C1-01C7-865BFD047B73" lastCalled="1084840279434" serviceURL="http://localhost:8080/axml2/services/politique">
    - <axml:params>
      - <axml:param name="get">
        <axml:value>pol152.81.48.22</axml:value>
      </axml:param>
    </axml:params>
  </axml:sc>
</politique>

- <control>
  - <axml:sc doNesting="true" frequency="every 60000" id="9C5CD28C-7F00-0001-0032-3274E2D34B11" lastCalled="1084840279357" methodName="confintern" mode="replace" serviceNameSpace="Membre" serviceURL="http://localhost:8080/axml2/servlet/AxisServlet">
    - <axml:params>
      - <axml:param name="confintern">
        <axml:value>152.81.48.22</axml:value>
      </axml:param>
    </axml:params>
  </axml:sc>
  <axml:text axml:origin="9C5CD28C-7F00-0001-0032-3274E2D34B11">11390999</axml:text>
</control>

- <controlex>

```
- <axml:sc doNesting="true" frequency="every 60044" id="9C5CD28C-7F00-0001-0032-
      3274E2D34A00" lastCalled="1084840279649" methodName="execute"
      mode="replace" serviceNameSpace="Membre"
      serviceURL="http://localhost:8080/axml2/servlet/AxisServlet">
    - <axml:params>
      - <axml:param name="execute">
          <axml:value>152.81.48.22</axml:value>
        </axml:param>
      </axml:params>
    </axml:sc>
  </controlex>
</ipsec>
```

Annexe 3

Document AXML de définition de service Web

(Prototype)

Politique.xml

Cette annexe représente un service AXML :
Flux d'entré : adresse IP .
Flux de sortie : politique de sécurité sous format XML.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <serviceDefinition type="query" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <parameters>
      <param name="s" type="xsd:string" />
    </parameters>
  - <definition>
    - <query>
        - <![CDATA[
          select pol
          from pol in politique//s;
        ]]>
      </query>
    </definition>
</serviceDefinition>
```

Annexe 4

Document AXML de politique de sécurité

(Prototype)

Politique.xml

Cette annexe représente le fichier AXML qui décrit la politique de sécurité au sein d'un pair Active XML.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <Ipsecpolitique xmlns:axml="http://www-rocq.inria.fr/verso/AXML">
    <Default>moyen</Default>
  - <pol152.81.48.26>
    - <Association>
      - <protocole id="AH" spi="24700">
        - <algorithme id="A">
            <name id="rfc2403" />
            <value name="hmac-md5" />
          </algorithme>
        </protocole>
      - <protocole id="ESP" spi="24701">
        - <algorithme id="E">
            <name id="rfc2451" />
            <value name="3des-cbc" />
          </algorithme>
        </protocole>
      </Association>
    - <Policy>
        <direction id="2" />
        <AH id="1" value="require" />
        <ESP id="1" value="require" />
      </Policy>
    </pol152.81.48.26>
    <pol152.81.48.22>moyen</pol152.81.48.22>
    <pol152.81.48.21>faible</pol152.81.48.21>
```

</Ipsecpolitique>

Annexe 5

Fichier de configuration racoon

(Prototype)

Cette annexe représente le fichier AXML de configuration de Racoon. Dans cette exemple nous allons à peu près rester fidèle à la configuration par défaut, qui est identique sur les deux hôtes 152.81.48.22 et 152.81.48.21 :

```
path pre_shared_key "/usr/local/etc/racoon/psk.txt";

remote anonymous
{
        exchange_mode aggressive,main;
        doi ipsec_doi;
        situation identity_only;

        my_identifier address;

        lifetime time 2 min;    # sec,min,hour
        initial_contact on;
        proposal_check obey;    # obey, strict or claim

        proposal {
                encryption_algorithm 3des;
                hash_algorithm sha1;
                authentication_method pre_shared_key;
                dh_group 2 ;
        }
}

sainfo anonymous
{
        pfs_group 1;
        lifetime time 2 min;
        encryption_algorithm 3des ;
```

Beaucoup de paramètres. Je pense que l'on peut encore en supprimer pour se rapprocher de la configuration par défaut. Remarquons ici quelques éléments notables. Nous avons configuré deux sections "anonymous", ce qui convient pour tous les hôtes distants. Ceci va ainsi faciliter les configurations supplémentaires. Il n'est pas nécessaire d'avoir de sections spécifiques à une machine particulière, à moins que vous ne le vouliez vraiment.

De plus, la configuration précise que nous nous identifions grâce à notre adresse IP ('my_identifier address') et que nous pouvons faire du 3des, sha1 et que nous utiliserons une clé "pré-partagée" se trouvant dans psk.txt.

Dans le fichier psk.txt, nous avons configuré deux entrées qui sont différentes suivant les hôtes. Sur 152.81.48.22 :

```
152.81.48.21 password1
```

Sur 152.81.48.21

```
152.81.48.22 password1
```

Soyez sûr que ces fichiers sont la propriété de root, et qu'ils ont le mode 0600. Dans le cas contraire, racoon ne pourra faire confiance à leur contenu. Notez que ces fichiers sont symétriques l'un de l'autre

Annexe 6

Exemple d'appel des service web AXIS par java

(Prototype)

```
import java.io.*;
import java.math.BigInteger;
import java.security.*;
import java.security.spec.*;
import java.security.interfaces.*;
import javax.crypto.*;
import javax.crypto.spec.*;
import javax.crypto.interfaces.*;
import com.sun.crypto.provider.SunJCE;
import deff.Machine2;
import deff.Machine3;
import deff.HexString;
import java.io.*;
```

```
import java.util.*;

//Add the following import statement:
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import org.apache.axis.encoding.XMLType;
import org.apache.axis.utils.Options;
import javax.xml.namespace.QName;

import javax.xml.rpc.ParameterMode;

public class CALL {}



public String ServiceCall(String host, String Paramètre, String methode)
{ //String res;
String  ret;
   try {
         boolean test=true;

         // Assign a variable for the endpoint URL, the destination for SOAP messages:

         String endpointURL = "http://"+host+":8080/axis/services/Machine2";

         // Instantiate Axis Service and Call objects; these objects store metadata about the
         // service to invoke:

         Service  service = new Service();
         Call     call    = (Call) service.createCall();

          // Set the web service endpoint address;
         call.setTargetEndpointAddress( new java.net.URL(endpointURL) );

         Set the name of the web service operation (method) to be invoked
         call.setOperationName(methode);

         // Set the parameter name :
         call.addParameter( "arg1", XMLType.XSD_STRING, ParameterMode.IN);

         // Set the return type. For example :
```

```
        call.setReturnType( org.apache.axis.encoding.XMLType.XSD_STRING );

        // Invoke the web service.
        ret = (String) call.invoke( new Object[] { Paramètre } );

       //Return the result to the console
        return ret;
    } catch (Exception e) {
        //System.err.println(e.toString());

            return "erreure";
    }

}

}
```

Annexe 7

Class HexString

(Prototype)

```
package deff;
import java.io.*;

/**
Contains utility functions for converting between hexadecimal strings
and their equivalent byte buffers or String objects.
*/
public class HexString
        {
        /**
        Returns a string containing the hexadecimal representation of the
        input string.
        @param s            a string to convert to hex
        @return the hex string version of the input string
        */
```

```
        public HexString(){}
  public static String stringToHex(String s)
        {
        byte[]    stringBytes = s.getBytes();

        return HexString.bufferToHex(stringBytes);
        }

        /**
        Returns a string containing the hexadecimal representation of the
        input byte array.
        @param buffer    a buffer to convert to hex
        @return the hex string version of the input buffer
        */
  public static String bufferToHex(byte buffer[])
        {
        return HexString.bufferToHex(buffer, 0, buffer.length);
        }

        /**
        Returns a string containing the hexadecimal representation of the
        input byte array.
        @param buffer              a buffer to convert to hex
        @param startOffset         the offset of the first byte in the buffer to process
        @param length              the number of bytes in the buffer to process
        @return the hex string version of the input buffer
        */
  public static String bufferToHex(byte buffer[], int startOffset, int length)
        {
        StringBuffer       hexString = new StringBuffer(2 * length);
        int                                endOffset = startOffset + length;

        for (int i = startOffset; i < endOffset; i++)
                {
                HexString.appendHexPair(buffer[i], hexString);
                }

        return hexString.toString();
        }
```

Annexe 8

Document XML de politique de sécurité

adresseip.xml

Cette annexe représente le fichier XML qui décrit la politique de sécurité au sein d'un pair Active XML.

```
<Association>
  - <protocole id="AH" spi="24700">
    - <algorithme id="A">
        <name id="rfc2403" />
        <value name="hmac-md5" />
      </algorithme>
    </protocole>
  - <protocole id="ESP" spi="24701">
    - <algorithme id="E">
        <name id="rfc2451" />
        <value name="3des-cbc" />
      </algorithme>
    </protocole>
 </Association>
- <Policy>
    <direction id="2" />
    <AH id="1" value="require" />
    <ESP id="1" value="require" />
```

Annexe 9

Exemple de fichier de configuration

(adresseip.conf)

Cette annexe représente un exemple de fichier de configuration d'IPSec.

```
#!/usr/local/sbin/setkey -f
# Examples d'implementation de ipsec a l'aide de fichier XML ::LORIN::

delete 152.81.48.21 152.81.48.22 ah 24500 ;
delete 152.81.48.22 152.81.48.21 ah 24700 ;
delete 152.81.48.21 152.81.48.22 esp 24500 ;
delete 152.81.48.22 152.81.48.21 esp 24700 ;

spddelete 152.81.48.21 152.81.48.22 any -P out ;
spddelete 152.81.48.22 152.81.48.21 any -P in ;
#association AH
add 152.81.48.21 152.81.48.22 ah 24500 -m tunnel -A hmac-md5
"1139099911390999";
add 152.81.48.22 152.81.48.21 ah 24700 -m tunnel -A hmac-md5
"1139099911390999";

#association ESP
add 152.81.48.21 152.81.48.22 esp 24500 -m tunnel -E 3des-cbc
"113909991139099911390999";
add 152.81.48.22 152.81.48.21 esp 24700 -m tunnel -E 3des-cbc
"113909991139099911390999";

spdadd 152.81.48.21 152.81.48.22 any -P out ipsec
   esp/tunnel/152.81.48.21-152.81.48.22/require
   ah/tunnel/152.81.48.21-152.81.48.22/require;

spdadd 152.81.48.22 152.81.48.21 any -P in ipsec
   esp/tunnel/152.81.48.22-152.81.48.21/require
   ah/tunnel/152.81.48.22-152.81.48.21/require;
```

Annexe 10

Passerelle XML/IPSEC

(Prototype)

Cette annexe représente la classe java qui réalise l'interfaçage entre l'implémentation d'IPsec et le document XML.

```
package samples.userguide.example15;
import java.io.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;
```

```
publique class Splitex extends HandlerBase {


private  boolean AH    = false;
private  boolean ESP    = false; //defau
private  boolean ahapp    = false;
private  boolean espapp    = false; //defaults
private  boolean setValidation    = false;
publique  StringBuffer associationah;
publique  StringBuffer associationah1;
publique  StringBuffer associationesp;
publique  StringBuffer associationesp1;
publique  StringBuffer policy;
publique  StringBuffer policy1;
publique  StringBuffer confirm;
publique  StringBuffer confirm1;
publique  StringBuffer ass;
publique FileOutputStream fos;
publique OutputStreamWriter osw;
publique String[] rep={         "/bin/setkey","/sbin/setkey",
                        "/etc/setkey","/usr/sbin/setkey",
                                        "/usr/local/bin/setkey","/usr/local/sbin/setkey",
                                        "/usr/src/bin/setkey","/usr/src/sbin/setkey"



                        };

protected boolean canonical;
//protected PrintWriter out;
publique String getip="152.81.48.21";
publique String spiah;
publique String spiesp;
publique String ipsource;

publique String Filename;
private  boolean AHGEN    = false;
private  boolean ESPGEN    = false;
private  boolean POLGEN    = false;
```

```
/** Default constructor. */

publique Splitex(String file,String spi1,String spi2) throws UnsupportedEncodingException {

 Filename=file;
 spiah=spi1;
 spiesp=spi2;

}

publique void startDocument() {

   try
            {
                    fos = new FileOutputStream(Filename);
                    osw = new OutputStreamWriter(fos, "ISO-8859-1");




              associationah.append("add "+ getip +" "+ atts.getValue(1));
             associationah1.append("add "+ atts.getValue(1) +" "+ getip);
              associationesp.append("add "+ getip +" "+ atts.getValue(1));
              associationesp1.append("add "+ atts.getValue(1) +" "+ getip);
              policy.append("spdadd "+ getip +" "+ atts.getValue(1));
              policy1.append("spdadd "+ atts.getValue(1)+" "+ getip);




                        if (name.equals("value"))
           { if(AH)
               {
         associationah.append(" "+atts.getValue(0));

          associationesp.append(" "+atts.getValue(0));
                associationesp1.append(" "+atts.getValue(0))
```

```
            }



             if (name.equals("key"))
            { if(AH)
                {
          associationah.append(" \""+atts.getValue(0)+'\";");
                associationah1.append(" \""+atts.getValue(0)+'\";');



                        }
                        if(ESP)
                {
         associationesp.append(" \""+atts.getValue(0)+'\";");
                associationesp1.append(" \""+atts.getValue(0)+'\";');



                        }


            }
    if (name.equals("direction"))
             {
         policy.append(" any -P out ipsec");
                 policy1.append(" any -P in ipsec");
          POLGEN=true;

                        }

    if (name.equals("AH"))
            { if(atts.getValue(0).equals("1"))
                {ahapp=true;
                        confirm=new StringBuffer();
```