REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université Saad Dahlab Blida1

Faculté des Sciences

Département d'Informatique

Projet de fin d'étude pour l'obtention du diplôme Master

Spécialité :
Sécurité des Systèmes d'Information

Thème

Un outil unifié pour la gestion des communications sécurisées en IPSec

Réalisé par : Amira Fateh et Zerarka Amine

Sujet proposé et encadré par : Dr Mohamed Amine Bouabid

Promoteur : Mr Mohamed Ould Khaoua

Président de jury : Mme Boustia

Jury : Mme Boutoumi

Soutenu le:
2016/2017

Remerciement

Avec l'aide du Bon Dieu le tout puissant que nous remercions tout le temps et infiniment pour tout ce qu'il nous a octroyé. Pour le temps bénis, la santé, la volonté du travail et la patience.

Puis *ilnous plaisant, avant d'entamer l'exposition de ce travail, d'exprimer toute nos gratitudes envers les personnes qui ont, de près ou de loin, contribué à la réalisation de ce projet.*

Nous adressons tout d'abord nos sincères remerciements aux intervenants professionnels responsables à notre formation Mr Bouabid Mohamed Amine et toute l'équipe de la société CERIST, pour leurs soutiens, leurs encadrements de qualité et leurs conseils judicieux tout au long de notre stage.

Nos vifs remerciements vont également à notre promoteur Mohamed Ould Khaoua et aux membres du jury pour nous avoir accordé leur attention.

Nous tiens finalement à exprimer notre adoration et notre respect profond envers nos chers collègues d'USDHB.

Amine & Fateh

Dédicace

Avec un énorme plaisir, un cœur ouvert et une immense joie, que je dédie mon travail et je remercie mes très chers, respectueux et magnifiques parents qui m'ont soutenus tout ou long de ma vie

À mon frère, mes sœurs et toute la famille Zerarka de proche ou de loin.

À mes amis de la vie.

À toute personne qui m'ont encouragé ou aidé au long de mes étude.

Zerarka Amine

Dédicace

Je dédie ce mémoire

À mes chers parents ma mère et mon père

Pour leur patience, leur amour, leur soutien illimitée, et leurs

Encouragements tout au long de mes études.

A mes frères et mes sœurs.

A toute la famille Amira.

A mes amis et mes camarades.

Sans oublier tous les professeurs que ce soit du

Primaire du moyen, du secondaire ou de l'enseignement supérieur.

Amira Fateh

Résumé

La gestion des politiques de sécurité est devenue de nos jours, un enjeu de taille pour les directions de systèmes d'informations. Avec l'évolution exponentielle des attaques, toute négligence risque de causer de grands dégâts, que ce soit sur le plan économique, politique ou même humain. Ainsi, pour garantir un maximum de sécurité, les grands leaders informatiques ne cessent de proposer et d'améliorer des nouveaux systèmes de sécurité.

Dans le cadre de notre projet, nous avons réalisé un système dédié à la gestion de politiques de sécurité au niveau d'un système d'exploitation Linux en se basant sur le standard CIM/WBEM. Notre système permet de modélisé les politiques de IPSec en format CIM et la traduise par la suite à l'aide d'un provider on des actions concrètes IPSec. Notre système repose sur une architecture adaptant les composants WBEM élémentaires à l'architecture standard d'un système de gestion de politiques (comme COPS). La traduction CIM/IPSec est réalisée par le Provider IPSecPro que nous avons développé et qui représente le noyau de notre travail car il assure d'un côté la traduction CIM de/vers IPSec et d'un autre coté le renforcement des politiques modélisées. Nous avons utilisé également un client CIM/WBEM permettant de faciliter la gestion des politiques. Notre contribution représente une brique essentielle dans un écosystème de composants logiciels permettant d'assurer une gestion de politiques de sécurité abstraites et indépendantes des plates-formes en se basant sur un standard de gestion largement adopté et déployé (CIM/WBEM) L'efficacité d'un tel écosystème est tributaire de l'adaptation du même standard pour d'autres systèmes de sécurité (comme AppArmor, GrSecure, XACML, etc.) et ce défi représente la perspective de notre travail.

Mots clés :

CIM/WBEM, IPSec, Gestion, Politiques de sécurité, Cops.

Abstract

Security policy management has become today a major challenge for the supervision of information systems. With the exponential growth of the attacks, any negligence can cause great damages, whether economical, political or even human. Thus, to ensure maximum safety, large IT leaders constantly suggest and improve new security systems.

As part of our project, we realized a system dedicated to managing security policies on operating system Linux based on CIM/WBEM standard. Our system allows modelled IPSec policies into CIM format and translated later with our provider into a concrete action IPSec. Our system is based on an architecture mapping WBEM elementary components to a standard policy management architecture (like COPS) The Provider IPSecPro we have been developed represents the core of our work since it ensures in one hand, CIM/IPSec translation and in the other hand policy enforcement. We have also developed a CIM/WBEM client to facilitate policy management through a simple and user-friendly graphical interface.

Our contribution is an essential building block in an ecosystem of software components ensuring the management of abstract and platform independent policies based on a widely adopted and deployed management standard (CIM/WBEM). The efficiency of such an ecosystem is dependent on the adaptation of the same standard for other security systems (like AppArmor, GrSecure, XACML, etc.) and this challenge represents a future work.

Sommaire

Liste des figures :

Introduction générale

Les réseaux et les systèmes informatiques prennent de plus en plus une place stratégique au sein des entreprises et des organisations. Ainsi l'atteinte à la sécurité de ses derniers est devenue quasi inacceptable. Ceci implique une prise en charge de la sécurité dès la phase de conception du système d'information par l'application de tous les principes, standards et bonnes pratiques liés à la sécurité.

IPSec est l'un des protocoles de sécurité les plus répandus et les plus mâtures pour la sécurisation des communications sur Internet. Conçu au départ comme un ensemble de mécanismes de sécurité intégrés au nouveau protocole IPv6, il a été adapté au protocole IP actuel (version 4) sous la forme d'une extension.

IPSec est présent sur la plupart des équipements de réseaux constituant Internet (les routeurs essentiellement) en plus de la majorité des équipements de sécurité ainsi que les systèmes d'exploitation modernes (Windows, Linux, Unix, *BSD, IOS, etc).

IPSec étant un protocole standardisé et ouvert, les équipements et systèmes qui l'implémentent sont (théoriquement) interopérables, c'est à dire qu'avec une configuration adéquate de part et d'autre, il est possible d'établir des sessions de communications IPSec entre un routeur CISCO et un système Linux (par exemple).

Par contre, l'interface de contrôle d'IPSec diffère d'un équipement à un autre et d'un système d'exploitation à un autre, ce qui rend la gestion d'IPSec dans un environnement hétérogène une activité complexe. De plus, dans de tels environnements hétérogènes, l'administrateur ne dispose pas d'une vue globale, exhaustive et homogène de l'ensemble des politiques IPSec et par conséquent de l'ensemble des associations de sécurités d'IPSec (les SAs) qui définissent les conditions pour l'établissement de sessions sécurisés ainsi que les règles et mécanismes de sécurités à appliquer sur ces sessions (ESP/AH, Transport/Tunnel etc.).

La gestion dirigée par les politiques (Policy Based Management) fondée sur des standards ouverts et reconnus, représente une solution intéressante au problème évoqué plus haut. En effet elle permettrait d'un côté, une gestion centralisée (et donc simplifiée) des réseaux et systèmes distribués et d'un autre côté, la définition de politiques de gestion de haut niveau et leur renforcement sur des systèmes hétérogènes. Une politique étant une règle abstraite qui gouverne un comportement plutôt qu'une configuration spécifique à une plateforme. Dans sa forme la plus élémentaire, une politique est une règle de la forme des politiques plus complexes, peuvent intégrer une combinaison de plusieurs règles élémentaires.

Nous nous intéressons en particulier aux standards ouverts développés par le DMTF, son modèle d'information CIM (Common Model Information) dédié à la gestion des systèmes et des réseaux ainsi que son architecture WBEM (Web Based Enteprise Management) définissant l'ensemble des éléments d'un système de gestion IT.

Le DMTF est une association à but non lucratif réunissant les acteurs IT les plus importants au monde ainsi que des universitaires et chercheurs. Il propose des standards ouverts dédiés à la gestion des réseaux, des systèmes distribués, des applications ainsi que toute entité physique ou virtuelle composant les systèmes d'information modernes.

En particulier, le DMTF propose des modèles et profiles de gestion dédiés aussi bien à la gestion dirigée par les modèles ainsi qu'à la gestion des communications IPSec.

Objectifs :

L'objectif de notre projet est la réalisation d'un Framework permet de suivi des politiques de sécurité IPSec sur des systèmes distribués appartenant au même domaine d'administration et de rendre la gestion d'IPSec dans un environnement hétérogène une activité simple.

Chapitre I

État de l'art

1.1. Introduction :

IPsec se présente sous la forme d'une norme, développée par un groupe de travail du même nom à l'IETF (Internet Engineering Task Force) depuis 1992. Une première version basique de cette extension d'IP a paru, 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 IPsec 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. [1]

Cette présentation débutera par une description des composants d'IPsec, des principes sur lesquels repose la sécurisation et du fonctionnement du système. Dans un second temps, nous verrons dans quelles situations IPsec peut être utilisé et, en particulier, comment il peut être déployé et exploité, en pratique, à l'échelle d'un réseau d'entreprise.

1.2. Présentation du protocole IPSEC :

1.2.1. Définition d'IPSEC :

IPSec (Internet Protocol Security) est un protocole développé par l'IETF (Internet Engineering Task Force) qui vise à sécuriser L'échange de données au niveau de La couche réseau (couche 3 du modèle OSI). Le réseau Ipv4 étant Largement déployé et La migration vers IPV6 étant inévitable, mais néanmoins longue, il est apparu intéressant de développer des techniques de protection des données communes à IPV4 et IPV6. Ces mécanismes sont couramment désignés par le terme IPSec pour IP Security Protocols. Ce n'est pas un remplaçant d'IP mais un complément. Ainsi, il intègre des notions essentielles de sécurité au datagramme IP qui en assureront l'authenticité, l'authentification et le cryptage. Pour cela, il fait largement usage de clé de sessions (Différent Types de clefs), IPSec est basé sur deux mécanismes. Le premier, AH, pour Authentification Header vise à assurer L'intégrité et L'authenticité des datagrammes IP. IL ne fournit par contre aucune confidentialité : Les données fournies et transmises par Ce "protocole" ne sont pas encodées. Le second, ESP, pour Encapsulating Security PayLoad permet aussi l'authentification des données et rajoute en plus le cryptage des informations. Bien qu'indépendants, ces deux mécanismes sont presque toujours utilisés conjointement. Enfin, le protocole IKE permet de gérer Les échanges ou Les associations entre

protocoles de sécurité. Avant de décrire ces différents protocoles, nous commençons par exposer les différents éléments utilisés dans IPSec.

Sa position dans les couches basses du modèle OSI lui permet donc de sécuriser tout type d'applications et de protocoles réseaux basés sur IP.

IPSec est très largement utilisé pour le déploiement de réseau VPN à travers Internet à petite et grande échelle.

Un très grand nombre d'équipements réseaux, en particulier les routeurs et les pare-feu, permettent l'utilisation d'IPsec. De même, les principaux systèmes d'exploitation pour micro-ordinateurs ou smartphones prennent en charge IPsec nativement. Le dialogue IPsec est généralement possible entre ces différents systèmes et équipements.

Dans de nombreux cas, l'utilisation d'IPsec présente un rapport ***"bénéfice en sécurité"/"coût"*** appréciable dans la mesure où cette technologie est prise en charge nativement par la plupart des systèmes clients et des équipements réseau et ne nécessite donc généralement pas d'investissements lourds. Il s'agit par ailleurs d'un protocole arrivé à maturité et bien connu. Sa mise en œuvre peut donc se faire sans charge excessive pour les équipes d'administration.

La figure suivante représente le protocole IPsec dans le modèle OSI

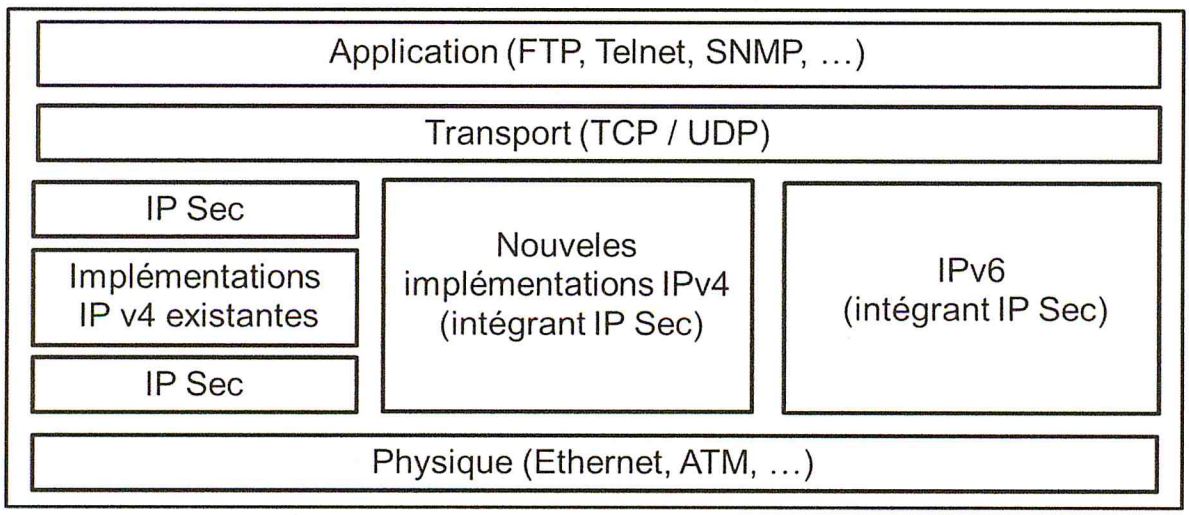

Figure 1 : Le protocole IPSec dans le modèle OSI

1.2.2. Gestion des flux IPSec :

Les flux IPSec sont gérés uni-directionnellement. Ainsi, une communication bidirectionnelle entre deux machines utilisant IPSec sera définie par diverse processus pour

chacun des sens de la communication, Les procédés détaillés ci-dessous respectent tous deux cette lois.

1.2.3. Security Policy :

Une SP définit ce qui doit être traité sur un flux. Comment nous voulons transformer un paquet En général, les adresses source et destination, le protocole et les ports source et destination (pour TCP ou UDP) suffisent pour définir une politique de sécurité («Security Policy") Un SP est stocké dans la « Security Policy Database ou SPD » Cette base sert donc à décider, lors de l'ouverture d'une communication, si elle doit ou non utiliser IPsec. Elle est, en général, configurée par l'administrateur de la machine.

Dans cette base, les communications qui peuvent ou qui doivent utiliser le protocole IPsec sont définies par les informations disponibles dans l'en-tête IP et dans l'entête du niveau transport (TCP, UDP, ICMP) des paquets de la communication.

Il y sera indiqué pour un flux donné :

- Les adresses IP de l'émetteur et du récepteur (unicast, mulitcast ou broadcast) ;
- Par quel protocole il devra être traité (AH ou ESP).
- Le mode IPSec à utiliser (tunnel ou transport).
- Le sens de la liaison (entrante ou sortante).

Notons qu'une SP ne définit qu'un protocole de traitement à la fois. Pour utiliser AH ET ESP sur une communication, deux SP devront être créées donc SP est unidirectionnelle.

1.2.4. Security Association:

Une SA définit comment sera traité le paquet en fonction de sa SP associée. Elle n'est que la "réalisation" des SP. Elle possède l'ensemble des propriétés de la liaison. Ainsi, elle sera représentée par une structure de donnée contenant les informations suivantes :

- Un compteur permettant de générer les numéros de séquence des entêtes AH et ESP.
- Un drapeau (flag) permettant d'avertir qu'en cas de dépassement du compteur précédemment décrit, on doit interrompre la communication.
- Une fenêtre d'anti-répétition dans laquelle le prochain numéro de séquence doit tomber;
- Spécification AH : algorithme d'authentification, clefs, durée de vie, etc.
- Spécification ESP : algorithme d'authentification et de chiffrement, clefs, etc.
- Mode IPSec : tunnel ou transport.

- Durée de vie de la SA.
- MTU.

Une SA est identifiée à un seul et unique flux unidirectionnel grâce à trois champs :

- L'adresse IP de destination (unicast, multicast ou broadcast).
- Le protocole utilisé, AH ou ESP.
- Le SPI (Security Parameter Index).

Comme nous pouvons le voir, une SA ne sera associée qu'a un seul des protocoles AH ou ESP. si nous voulons protéger un flux avec ces deux protocoles, deux SA devront être créé.

LE SPI est un indice (ou ID) sur 32 bits attribué au SA lors de sa création. Nous verrons plus loin que sa génération dépendra du mode de gestion des clés de sessions. Il sert à distinguer les différentes SA qui aboutissent à une même destination et utilisant le même protocole.

1.2.5. Bases de données SPD et SAD :

Tout système implémentant IPSec possède donc 2 bases de données distinctes dans lesquelles il stocke ses SP (ici, SPDatabase) et ses SA (ici, SADatabase).

La SPD (Security Policy Database) est une liste ordonnée d'entrées contenant des critères de contrôle d'accès, similaires à des règles de pare-feux. La SPD est statique par défaut car l'ordre des enregistrements qu'elle contient est très important; en effet, plusieurs règles peuvent s'appliquer à un même paquet en théorie mais seule la première sera réellement effective, d'où l'importance de l'ordre. Il est pourtant possible d'avoir des SPDs dynamiques, mais cela requiert un réordonnancement à la volée des entrées.

Il y a 2 SPDs par interfaces, une pour le trafic entrant, l'autre pour le trafic sortant. Chaque entrée de ces SPDs précise un traitement à appliquer au paquet pour lequel la règle s'applique (quand le critère de sélection ou sélecteur est vrai); ces traitements sont :

- DISCARD, dans ce cas celui-ci sera tout simplement rejeté. Il n'est pas autorisé à sortir de la passerelle ni à la traverser ni à être délivré à une quelconque application.
- BYPASS IPSEC laisse passer le trafic sans traitement IPSec.
- APPLY IPSEC signifie que des services IPSec sont à appliquer à ce trafic.
- Pour chaque trafic soumis à des services IPSec, la base SPD possède une référence vers la SA correspondante dans la base SAD,qui contient toutes les informations requises pour caractériser et échanger des données à protéger. Ainsi, chaque SA contient des éléments permettant de déterminer à quel type de trafic ou paquet elle s'applique. Si

cette entrée n'est pas définie, dans le cas d'une gestion dynamique des clés, celle-ci sera alors créée en accord avec la configuration définie par l'administrateur.

1.2.6. Modes d'IPSec : [2]

Il existe deux modes d'utilisation d'IPSec : le mode transport et le mode tunnel. La génération des datagrammes sera différente selon le mode utilisé.

a) Mode Transport :

Ce mode est utilisé pour créer une communication entre deux hôtes qui supportent IPSec. Une SA est établie entre les deux hôtes. Les entêtes IP ne sont pas modifiés et les protocoles AH et ESP sont intégrés entre l'entête IP et l'entête du protocole transporté. Ce mode est souvent utilisé pour sécuriser une connexion Point-To-Point.

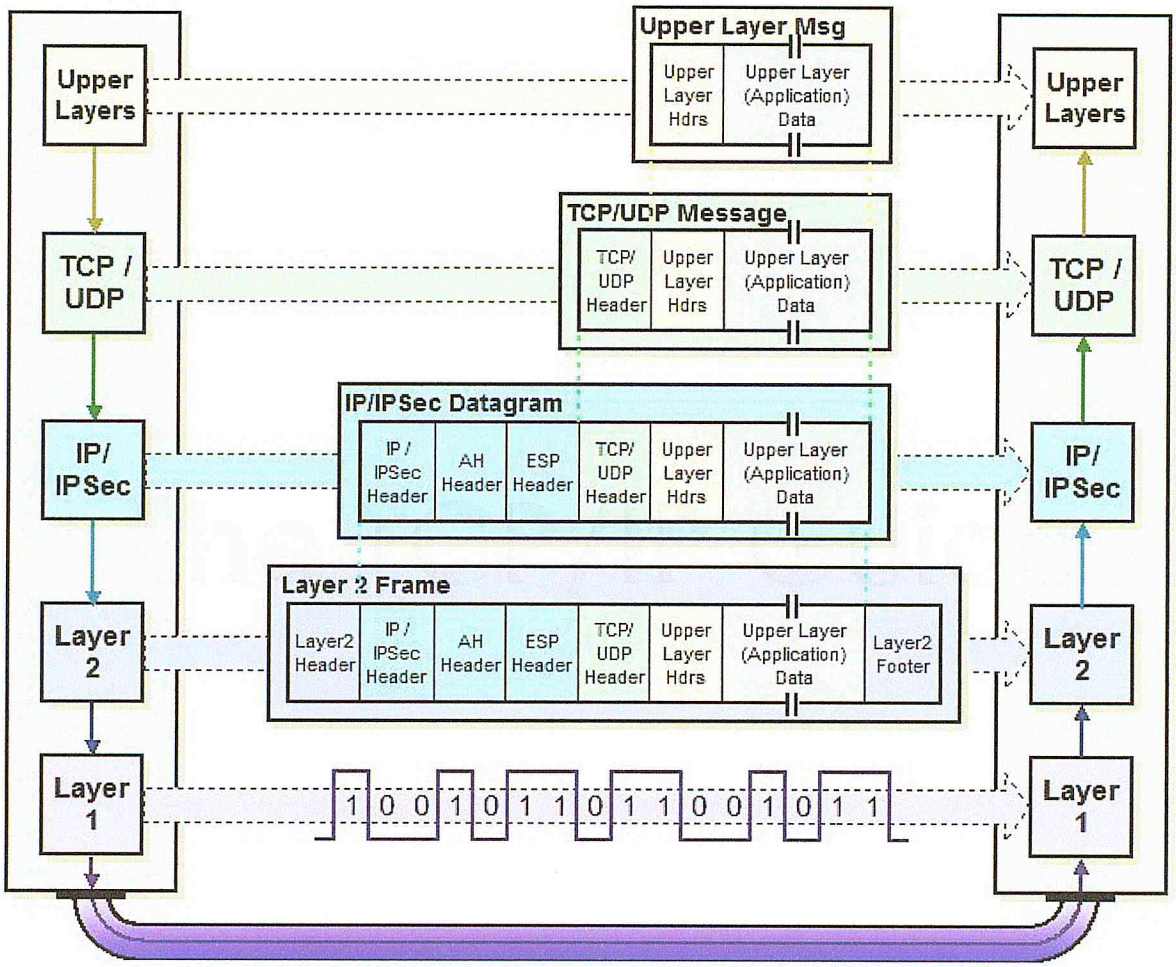

Figure 2 : IPSEC transport mode opération

Lorsque IPSec fonctionne en mode transport, il est intégré à IP et utilisé pour transporter directement le message des couches supérieures (TCP / UDP). Après le traitement, le datagramme possède un seul en-tête IP contenant les en-têtes AH et / ou ESP du IPSec

Mode Tunnel :

Ce mode est utilisé pour encapsuler les datagrammes IP dans IPSec. La SA est appliquée sur un tunnel IP.

Ainsi, les entêtes IP originales ne sont pas modifiés et un entête propre à IPSec est créé. Ce mode est souvent utilisé pour créer des tunnels entre réseaux LAN distants. Effectivement, il permet de relier deux passerelles capables d'appliquer IPSec sans perturber le trafic IP des machines du réseau qui ne sont donc pas forcément prêtes à utiliser le protocole IPSec.

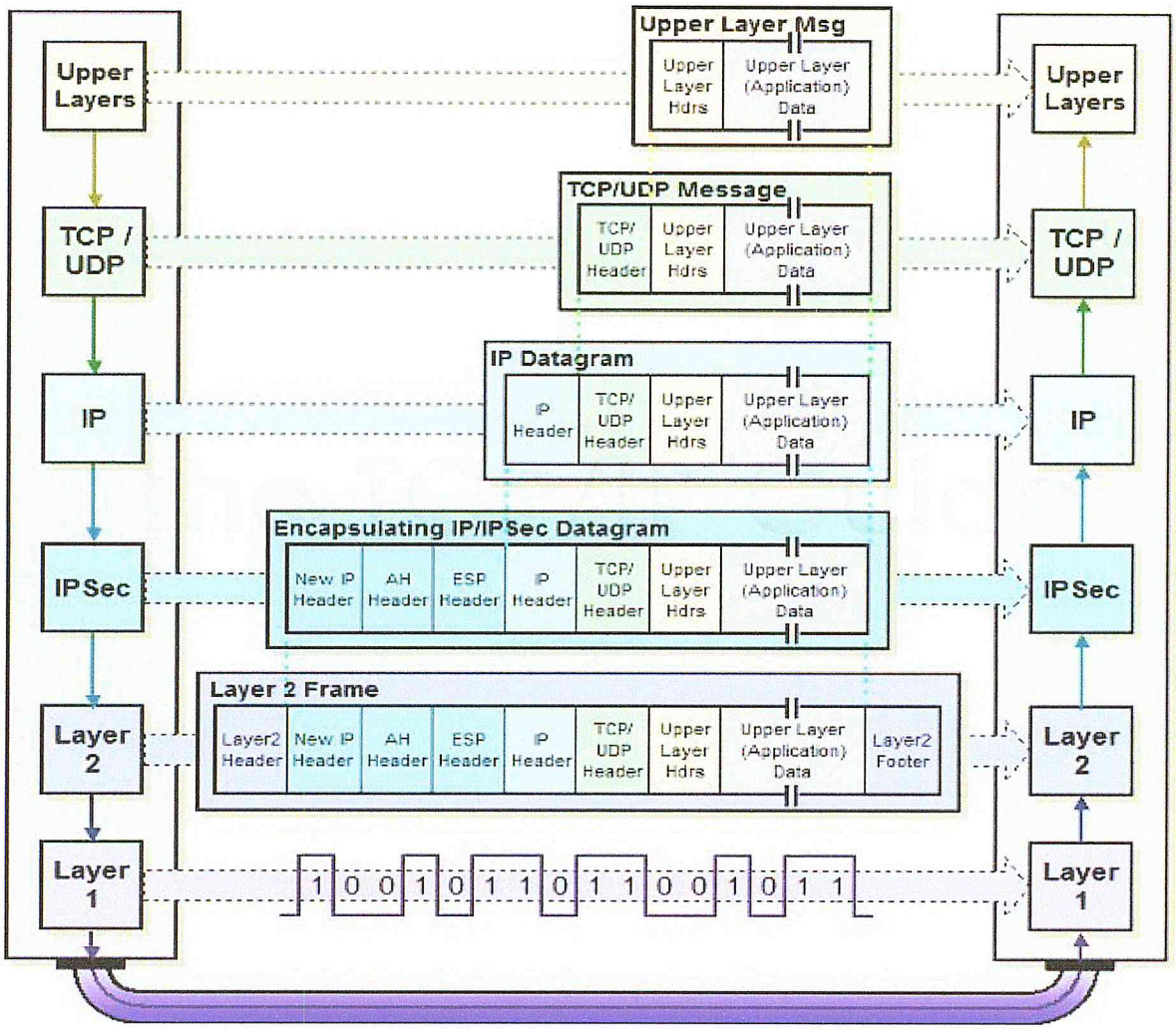

Figure 3 : IPSEC tunnel mode opération

Le mode tunnel IPSec représente une encapsulation d'un datagramme IP complet, formant un tunnel virtuel entre des périphériques compatibles IPSec. Le datagramme IP est transmis à IPSec, où un nouvel en-tête IP est créée avec les extensions AH et / ou ESP.

1.2.7. Détails des protocoles IPSec :

Nous verrons que certains champs sont présents dans les deux protocoles AH et ESP. En cas d'utilisation des deux protocoles pour un même flux de donnée, il n'y aura cependant pas de redondance d'information.

Effectivement, il faut garder à l'esprit que AH et ESP seront gérés séparément par des SA différents.

AH et ESP sont deux protocoles utilisant des clés de sessions utiles à leurs traitement sur le datagramme IP.

a) En-tête AH(Le protocole d'authentification) :

Le protocole AH (Authentification Header) fournit les services de sécurité suivants :

- Intégrité en mode non-connecté
- Authentification des données
- Anti-rejeu (optionnel)

L'en-tête AH se compose de 6 champs comme le décrit l'image suivante :

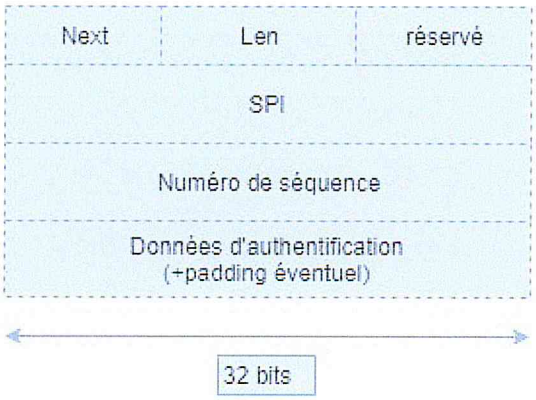

Figure 4 : Composant de L'en-tête AH

Un des plus importants est l'ICV (Integrity Check Value, "Données d'authentification") qui est le résultat d'un procédé cryptographique sur les données à protéger et qui va permettre de vérifier l'intégrité de celles-ci. La taille totale de l'en-tête est de 24 octets; l'ICV est bien entendu rembourré pour atteindre une taille constante (car les algorithmes utilisés peuvent différer).

En conclusion, AH permet de se protéger contre les attaques de type :

- Spoofing et autres modifications des paquets IP
- Rejeux
- DoS quand ils sont basés sur la charge impliquée par Les calculs cryptographiques (la vérification d'intégrité n'intervient qu'après)

Les deux figures suivante représente l'utilisation du protocole AH en mode transport et tunnel

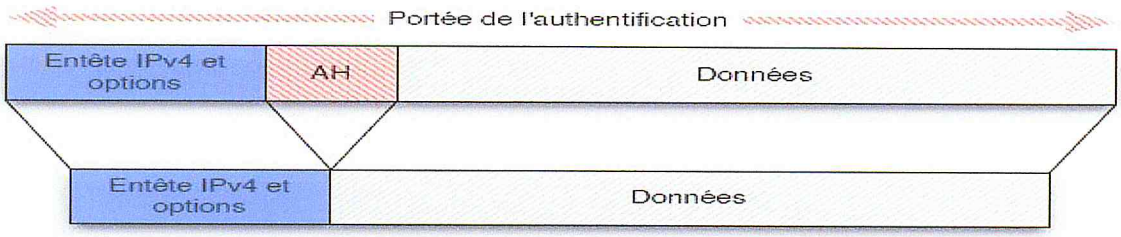

Figure 5 : Utilisation d'AH en mode transport dans un datagramme Ipv4

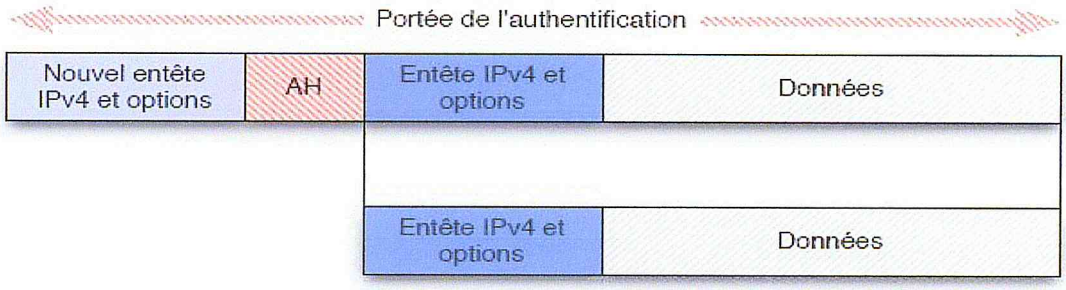

Figure 6 : Utilisation d'AH en mode tunnel dans un datagramme Ipv4

Le tableau suivant illustre les détails des champs de l'en-tête AH :

8	16	32
Entête suivante	Longueur des données	Réservé
Security paramétrés index (SPI)		
Numéros de séquence		
Données Authentification (variable)		

Figure 7: Détails des champs de l'ent-éte AH

- Entête suivante : ce champ permet de spécifier le type du protocole transporté.

- Longueur des données : longueur de l'entête AH par facteur de 32-bit, le minimum étant 2.
- SPI : index unique définissant la SA pour ce paquet.
- Numéros de séquence : compteur utile au mécanisme anti-répétition.
- Données Authentification (variable) : champs contenant les signatures de hachages permettant d'authentifier l'émetteur et l'authenticité des données. La taille de ce champ dépend des protocoles de hachage utilisés.

b) Champs ESP(Le protocole de confidentialité) :

Le protocole ESP (Encapsulating Security Payload) fournit les services de sécurité suivants :

- Confidentialité
- Protection contre l'analyse du trafic
- Intégrité en mode non-connecté (comme AH)
- Authentification des données (comme AH)
- Anti-rejeu (comme AH)

On peut voir tout de suite qu'ESP couvre les services offerts par AH, et on peut se demander pourquoi AH est utilisé et pourquoi l'on complique ainsi un protocole qui l'est déjà bien assez. C'est en effet une question qui est d'actualité mais néanmoins il faut souligner le fait qu'AH offre des services plus complets qu'ESP car il est le seul à protéger l'en-tête du paquet (en-tête original en mode Transport, en-tête IPSec en mode Tunnel). ESP ne protège que les données (c'est-à-dire tout au plus l'en-tête original en mode Tunnel).

L'en-tête ESP se compose de 7 champs (dont 2 optionnels) comme le décrit l'image suivante :

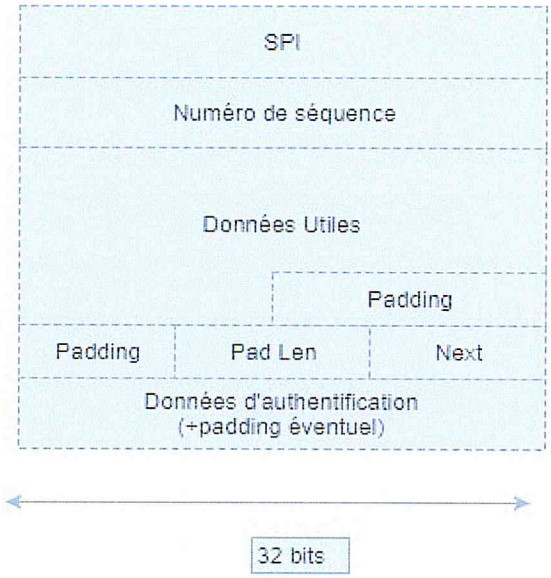

Figure 8 : Composant de L'en-tête ESP

Nous ne rentrerons pas dans les détails de chaque champ ici, mais nous invitons les lecteurs à consulter la RFC ((Request For Comments) correspondante pour plus d'informations sur le sujet. Très brièvement, le chiffrement sera appliqué aux données utiles jusqu'au champ NEXT inclus (ce champ contient un identifiant du protocole supérieur, au niveau 4). Comme les données utiles ne sont pas prédéfinies, leur longueur peut varier grandement et un padding assez important est requis. Il est obligatoire et sa longueur est explicitée dans le champ PAD LEN. Les données d'authentification protègent les données du champ SPI au champ NEXT inclus; en cas d'utilisation des 2 mécanismes simultanément, le chiffrement est effectué en premier suivi du calcul des données d'intégrité. Cela a pour résultat d'éviter des attaques par déni de service au, ainsi que de permettre d'effectuer les 2 opérations en parallèle (à la réception).

En conclusion, ESP permet de se protéger contre les attaques de type :

- espionnage et autres divulgations d'informations
- rejeux (optionnel)
- analyse de trafic (optionnel)

Les deux figures suivantes représentent le fonctionnement du protocole ESP avec le mode tunnel et transport

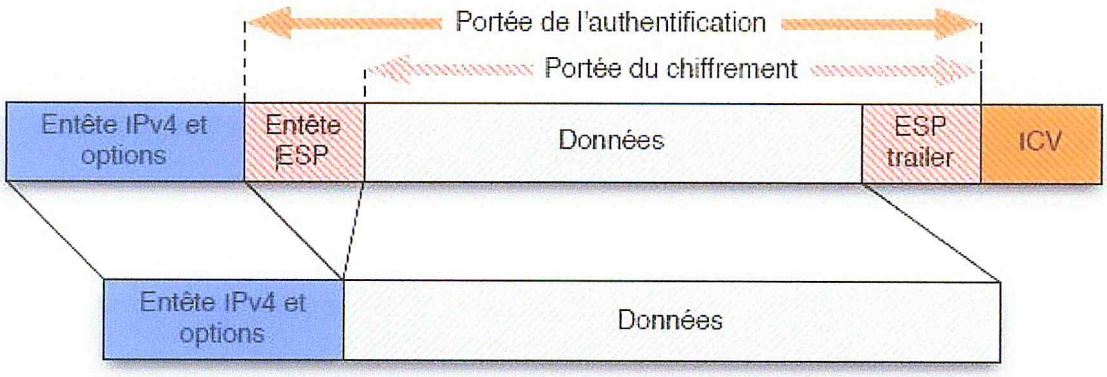

Figure 9 : Utilisation d'ESP en mode transport dans un datagramme Ipv4.

ICV désigne « Integrity Check Value », valeur utilisée par le mécanisme de contrôle d'intégrité.

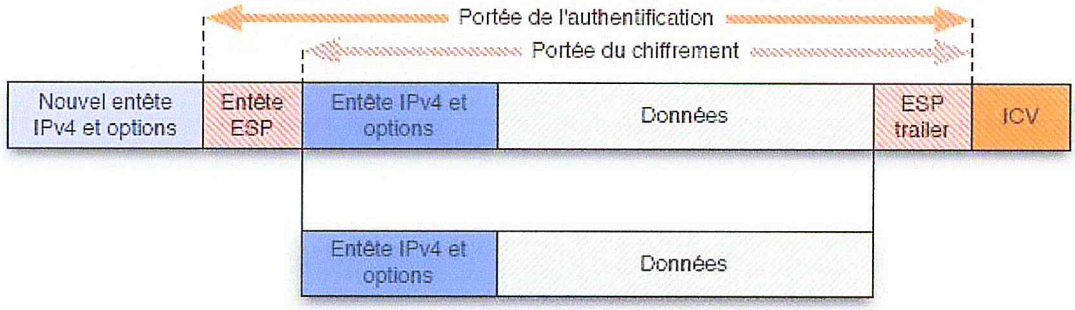

Figure 10 : Utilisation d'ESP en mode tunnel dans un datagramme Ipv4.

Le tableau suivant illustre les détails des champs du protocole ESP :

16	24	32
Security association identifier (SPI)		
Numéros de séquence		
Données		
Bourrage (0-255 octets)		
	Taille du Bourrage	Entête suivante
Données Authentification (variable)		

Figure 11: Détails des champs du protocole ESP

- SPI : index unique définissant la SA pour ce paquet.
- Numéros de séquence : compteur utile au mécanisme d'anti-répétition.

- Données : données du protocole de couche supérieure.
- Bourrage : sert au cryptage des données. Certains protocoles nécessitent une certaine taille afin d'être plus efficace et/ou applicable.
- Taille du bourrage : indique la taille du bourrage.
- Entête suivante : ce champ permet de spécifier le type du protocole transporté.
- Données Authentification (variable) : champs contenant les signatures de hachages permettant d'authentifier l'émetteur et l'authenticité des données. La taille de ces champs dépend des protocoles de hachage et du cryptage utilisés. [3]

Tous les processus que nous venons de découvrir son donc basés sur des mécanismes de cryptage et de hachage complexes qui utilisent des clés de longueurs variables. Avant d'utiliser un VPN (Virtual Protocol network), il faut donc s'intéresser à la gestion de ses clés.

1.2.8. Etablissement d'un lien IPsec :

Les mécanismes cryptographiques utilisés pour la protection en intégrité ou en confidentialité sont paramétrés par une ou plusieurs clés. Ces éléments doivent être partagés par les différents hôtes.

En employant IPSec deux approches sont possibles : mettre en place manuellement des clés sur chaque hôte ou utiliser le mécanisme d'échange de clés IKE pour que les hôtes puissent négocier ces clés.

Le schéma suivant représente les différentes étapes qui permet de créé des communications IPSec

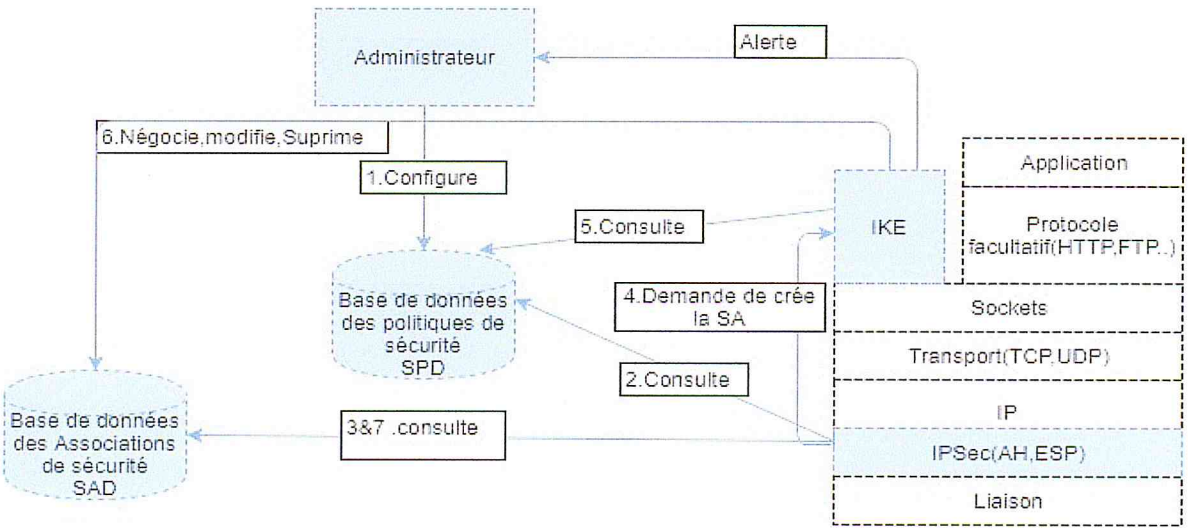

Figure 12 : Composants d'IPsec et actions à l'émission de données

1.3. Gestion des clefs IPSec : [1]

1.3.1. Les différents types de clés :

- **Clefs de chiffrement de clefs :**

Ces clefs sont utilisées afin de chiffrer d'autres clefs et ont généralement une durée de vie longue. Les clefs étant des valeurs aléatoires, l'utilisation d'autres clefs pour les chiffrer rend les attaques par cryptanalyse (tentatives de déchiffrement du message) plus difficiles. La cryptographie à clef publique est souvent utilisée pour le transport de clefs, en chiffrant la clef à transporter à l'aide d'une clef publique.

- **Clefs maîtresses :**

Les clefs maîtresses sont des clefs qui ne servent pas à chiffrer mais uniquement à générer d'autres clefs par dérivation. Une clef maîtresse peut ainsi être utilisée, par exemple, pour générer deux clefs : une pour le chiffrement et une pour la signature.

- **Clefs de session ou de chiffrement de données :**

Ces clefs, contrairement aux précédentes, servent à chiffrer des données.

1.3.2. Systèmes de gestion des clés :

a) Mise à la clé manuelle :

Les algorithmes et les clés peuvent être paramétrés manuellement sur chaque équipement. On parle généralement du «Manual Keying »; il est important de ne pas confondre cette méthode avec l'authentification « Pre-Shared Key » d'IKE abordée au chapitre suivant.

La mise à la clé manuelle est fortement déconseillée. Elle exige en effet une configuration fastidieuse, la clé étant idéalement différente pour chaque couple d'hôtes. Par ailleurs, il est difficile en pratique de renouveler les clés à une fréquence suffisante pour être conforme tant aux bonnes pratiques cryptographiques (usure de la clé) que de SSI (cryptopériode). En outre, elle ne permet pas de bénéficier de la propriété de « Perfect Forward Secrecy ».

Enfin, elle ne permet pas de mettre en œuvre des mécanismes d'authentification cryptographiques.

En définitive, la mise à la clé manuelle doit être réservée à des procédures de tests ou de diagnostics ou à des systèmes très particuliers ayant fait l'objet d'une étude de sécurité approfondie, notamment pour s'assurer que le cycle de vie des clés est bien géré et qu'il y a bien une clé différente pour chaque usage.

permettre de sécuriser les échanges de la phase 2. La phase 2 est en revanche appelée plusieurs fois. En effet, les clés qui servent à chiffrer deviennent vulnérables avec le temps ou quand on s'en sert beaucoup. Cette phase est donc régulièrement ré-effectuée pour changer certaines clés de sessions.

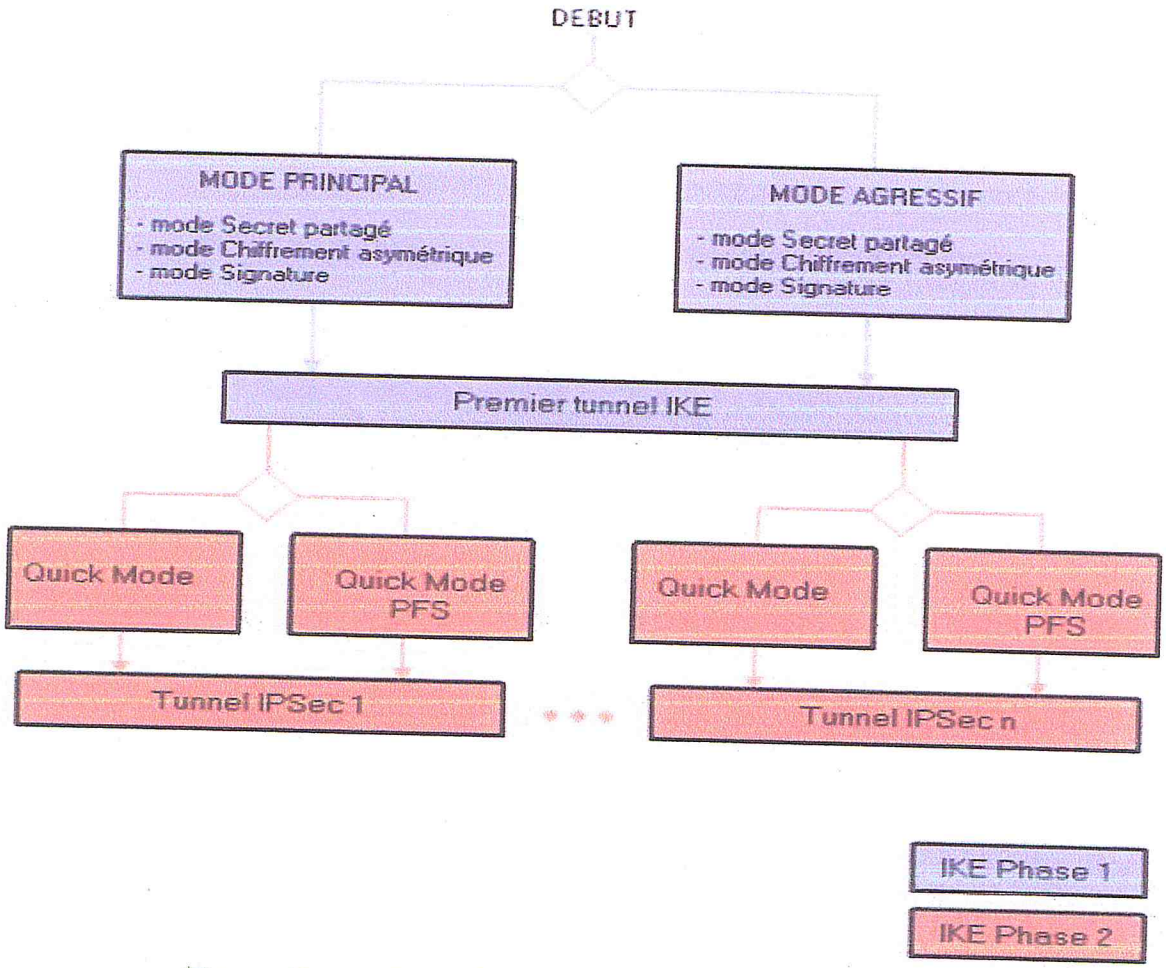

Figure 13 : les phases de IKE employé dans le cadre d'IPSec.

1.4. Les services proposés par IPSec :

Nous avons cité précédemment quelles étaient les propriétés basiques des tunnels de chiffrement, et ce indépendamment du protocole utilisé. Citons à présent quels sont les services de sécurité proposés par IPSec:

a) **Authentification des extrémités :** Cette authentification mutuelle permet à chacun de s'assurer de l'identité de son interlocuteur. Rappelons tout de même qu'IPSec est un protocole de niveau 3 et qu'il ne fournit donc qu'une authentification de niveau égal,

c'est-à-dire une authentification des machines mettant en œuvre le protocole plutôt que des personnes utilisant réellement la machine. Nous verrons techniquement comme l'authentification est effectuée dans les paragraphes suivants.

b) **Confidentialité des données échangées :** IPSec permet si on le désire de chiffrer le contenu de chaque paquet IP pour éviter que quiconque ne le lise.

c) **Authenticité des données :** IPSec permet de s'assurer, pour chaque paquet échangé, qu'il a bien été émis par la bonne machine et qu'il est bien à destination de la seconde machine.

d) **Intégrité des données échangées :** IPSec permet de s'assurer qu'aucun paquet n'a subi de modification quelconque (attaque dite active) durant son trajet.

e) **Protection contre les écoutes et analyses de trafic :** IPSec permet de chiffrer les adresses IP réelles de la source et de la destination, ainsi que tout l'en-tête IP correspondant. C'est le mode de tunneling, qui empêche tout attaquant à l'écoute d'inférer des informations sur les identités réelles des extrémités du tunnel, sur les protocoles utilisés au-dessus d'IPSec, sur l'application utilisant le tunnel (timing-attacks et autres) ...

f) **Protection contre le rejeu** : IPSec permet de se prémunir contre les attaques consistant à capturer un ou plusieurs paquets dans le but de les renvoyer à nouveau (sans pour autant les avoir déchiffrés) pour bénéficier des mêmes avantages que l'envoyeur initial. [5].

1.5. Définition d'un VPN (Virtual Private Network) :

Décomposons l'expression VPN.

Network : un réseau est constitué de plusieurs machines qui peuvent communiquer entre elles d'une façon ou d'une autre. Ces machines peuvent être dans un même endroit physiquement ou dispersées, et les méthodes de communication sont diverses.

Private : privé veut dire que les communications entre deux ou plusieurs machines sont secrètes et donc inaccessibles pour une machine ne participant pas à la communication privée.

Virtual : dans le concept de virtuel, nous retiendrons l'émulation d'une fonction d'un objet qui n'est pas vraiment là. Un réseau virtuel n'est pas un réseau physique, mais émuler pour faire croire à un réseau physique.

Un réseau privé virtuel est donc l'association de ces trois concepts. Une fois en place, il vous offre la possibilité d'utiliser un réseau public non sécurisé pour créer un réseau privé (données cryptées) et y faire circuler des données. [25]

1.6. Différents cas d'usage d'IPsec :

1.7. Conclusion :

IPsec est un système très complet qui peut répondre à beaucoup de besoins en matière de sécurité et s'adapter à de nombreuses situations. Sa conception en fait un système très sûr et sa nature de norme garantit l'interopérabilité entre les équipements de différents fournisseurs. Ces avantages, couplés à la prédominance grandissante du protocole IP, vont certainement faire d'IPsec un acteur important de la sécurité des réseaux informatiques. Il lui manque encore, pour être utilisé à grande échelle, un peu de maturité et surtout un système de gestion centralisée et dynamique des politiques de sécurité. Les avancées actuelles dans ce domaine laissent à penser qu'il ne s'agit que d'une question de temps avant qu'un tel système ne voie le jour. L'apparition d'infrastructures à clefs publiques fonctionnelles et reconnues est également indispensable pour une utilisation pratique et répandue d'IPsec.

Chapitre II

Étude des standards CIM et WBEM du DMTF

2.1. Introduction :

L'environnement de l'informatique ubiquitaire est très complexe et varie facilement, en raison de la présence de supports variés et nombreux (mobile, réseaux sans fils,...) et de leurs configurations.

Pour réaliser une application distribuée sur cet environnement, il faut donc pouvoir connaitre à tout moment le contexte dans lequel cette application évolue, afin d'exploiter les différentes ressources de manière optimisée, et savoir s'il est nécessaire d'entreprendre des actions d'adaptation à ce contexte. Il est donc utile de pouvoir récupérer et gérer des informations de contexte sur cet environnement, telles que les systèmes d'exploitation, les préférences utilisateurs…

Le standard CIM (Common Information Model) et l'architecture WBEM (Web Based Enterprise Management) offrent des possibilités qui se rapprochent des besoins du notre projet. En effet, ces technologies permettent de récupérer et gérer facilement, au sein d'un réseau, des informations sur le système matériel et logiciel.

Donc nous proposons dans ce chapitre une étude détaillée des standards CIM et WBEM du DMTF (Distributed Management Task Force), en particulier les profiles dédiés à la gestion dirigée par les politiques (Policy profiles) et une architecture de gestion afin de démontrer son utilité dans le contexte de gestion.

2.2. Pourquoi WBEM ET CIM ? :

Le choix de l'architecture que nous utilisons a été principalement motivé par les facilités d'expression de l'information de gestion qu'elle offre. En effet, notre démarche focalise sur la nature de l'information de gestion à chacun de ces niveaux d'abstraction ainsi que les interdépendances entre ces niveaux. Nous ne traitons pas les éléments liés à l'aspect fonctionnel, architectural, organisationnel ou protocolaire de la solution de gestion.

Il existe plusieurs approches pour la modélisation des ressources de gestion. Nous pouvons citer les MIB (Management Information Base)/SMI [RFC 1155, RFC 2578, RFC 3216] pour l'architecture SNMP (Simple Network Management Protocol), les MIB/GDMO [ISO 10165-4] pour les architectures OSI et TMN (Telecommunication Management Network) ou encore les PIB (Policy Information Base)/SPPI (Security Support Provider Interface) (SSPI [RFC 3159] pour l'architecture de gestion à base de politique. Cependant, ces modèles d'information ne représentent que des technologies spécifiques et ne prennent pas en compte nativement ce type d'abstraction.

Au contraire, l'initiative CIM/WBEM du DMTF, qui propose une approche pour unification de la gestion des réseaux et des systèmes distribués par un modèle d'information. Elle intègre pour cela le méta-modèle CIM (Common Information Model [DMTF 1999]) qui permet l'intégration des différents modèles d'information existants (comme SNMP).

Nous avons choisi d'implémenter notre travail avec l'initiative WBEM parce que :

1. **D'un point de vue conceptuel** : CIM permet une approche spécifique et indépendante des plates-formes pour les modèles d'information. Cette capacité permet d'intégrer de nouveaux modèles d'information dans CIM sans avoir à se préoccuper du protocole de distribution sous-jacent. De plus, étant orienté objet, l'ajout de nouveaux modèles est simplifié. Il permet aussi différents niveaux d'abstraction.
2. **D'un point de vue pratique** : CIM est de plus en plus retenu et intégré dans les nouvelles versions de produit de gestion (comme OpenView de Hewlett-Packard, OpenMaster de Bull) et pour de nouveaux environnements (Cisco, Sun ou encore Microsoft). Les spécifications des ressources gérées sont augmentées régulièrement. [10]

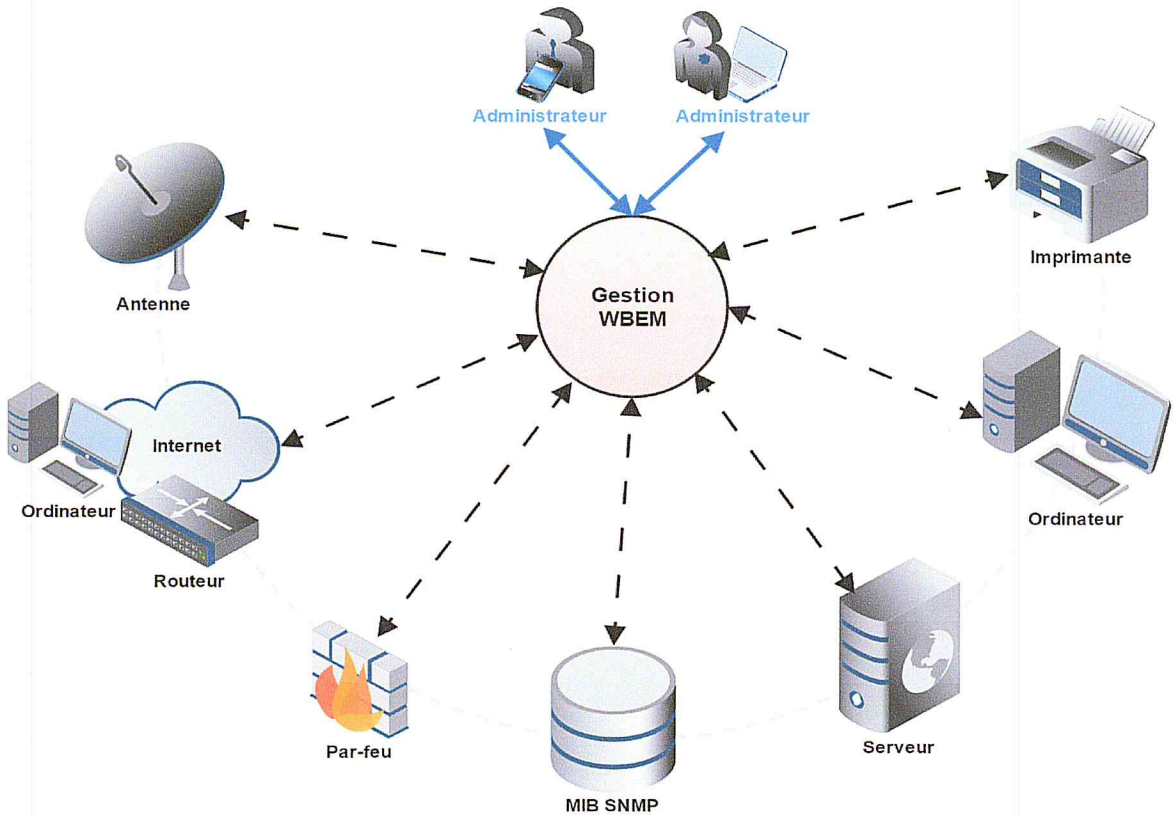

Figure 14 **:** Gestion WBEM homogène d'un réseau hétérogène. [11]

2.3. L'initiative WBEM du DMTF : [10]

2.3.1. Définition :

WBEM (Web Based Enterprise Management) Est un ensemble de technologies et de standards Internet de gestion développé pour unifier la gestion des environnements informatiques distribués. WBEM inclus des protocoles, langages de requêtes, et autres outils nécessaires à l'échange d'informations en format CIM.

Donc il permet d'exploiter la puissance d'expression offerte par CIM pour permettre la gestion d'un réseau d'un point de vue matériel et logiciel, et de façon automatisée. Comme toute autre approche de gestion, WBEM propose une architecture se basant sur un ensemble d'entités. Il contient 4 types d'éléments :

- **Un serveur WBEM :** qui héberge le schéma CIM complet ainsi que les instances fournies par les Providers (entités WBEM présentées ci-après). Il fonctionne comme une base de données objets: le schéma CIM fourni les «colonnes » (arguments et structures des classes et méthodes) que les Providers remplissent. Le serveur contient 3 parties essentielles (figure 2) :
  - Le CIMOM (CIM Object Manager) C'est le moteur de gestion des flux de requêtes et de réponses venant et/ou destinées aux autres entités de l'architecture. En particulier il contrôle l'intégrité des objets CIM échangés eu égard au schéma CIM enregistré dans le référentiel CIM
  - Une base de données objets qui stocke le schéma CIM et les informations instanciés de ce schéma, aussi appelée «référentiel CIM (repository) »
  - Divers protocoles pour permettre la communication avec les autres éléments de l'architecture : CIM-XML et WS-MAN par exemple.

  L'approche WBEM définit ses composants de base de la façon suivante :

  1. **Une architecture fonctionnelle :** qui définit les acteurs intervenants dans la gestion, leur répartition, leurs fonctions et leurs rôles.
  2. **Un modèle informationnel :** qui représente une approche de modélisation et un formalisme pour la spécification de nouveaux modèles de gestion (syntaxe et sémantique). Il intègre aussi un modèle fonctionnel générique (se basant sur le modèle informationnel) comportant un ensemble d'objets gérés de base.
  3. **Un modèle de communication :** qui comprend un service et un protocole assurant la communication entre les entités.

- **L'application de gestion :** aussi appelée client CIM, qui récolte et exploite les informations disponibles dans le serveur WBEM, elle constitue l'interface entre le domaine géré et le gestionnaire. Elle présente à ce dernier un accès aux informations qu'elle récupère auprès du gestionnaire d'objets CIMOM.
- **Le gestionnaire d'objets CIMOM (CIM Object Manager) :** Aussi appelé serveur CIM, il représente l'entité principale dans l'architecture WBEM. Il assure le maintien de la base d'informations de gestion (MIB) qui est appelée CIMOR (CIM Object Repository) dans la terminologie WBEM. Le rôle du CIMOM consiste à donner aux applications de gestion l'accès aux informations contenues dans cette CIMOR, et qui donnent une vue globale et homogène des ressources gérées, puisqu'elles sont toutes modélisées avec CIM et suivant des modèles de références.
- **Le provider** : Il assure la liaison entre le gestionnaire d'objets CIMOM et les ressources gérées. C'est un composant essentiel dans l'architecture WBEM car, il permet d'appliquer les opérations CIM, qu'un gestionnaire invoque sur les éléments gérés. Un provider peut être vu comme un traducteur, puisqu'il permet de traduire les informations CIM qu'il communique avec le CIMOM en actions ou en valeurs concrètes relatives aux ressources qu'il gère. Un provider possède donc deux interfaces.

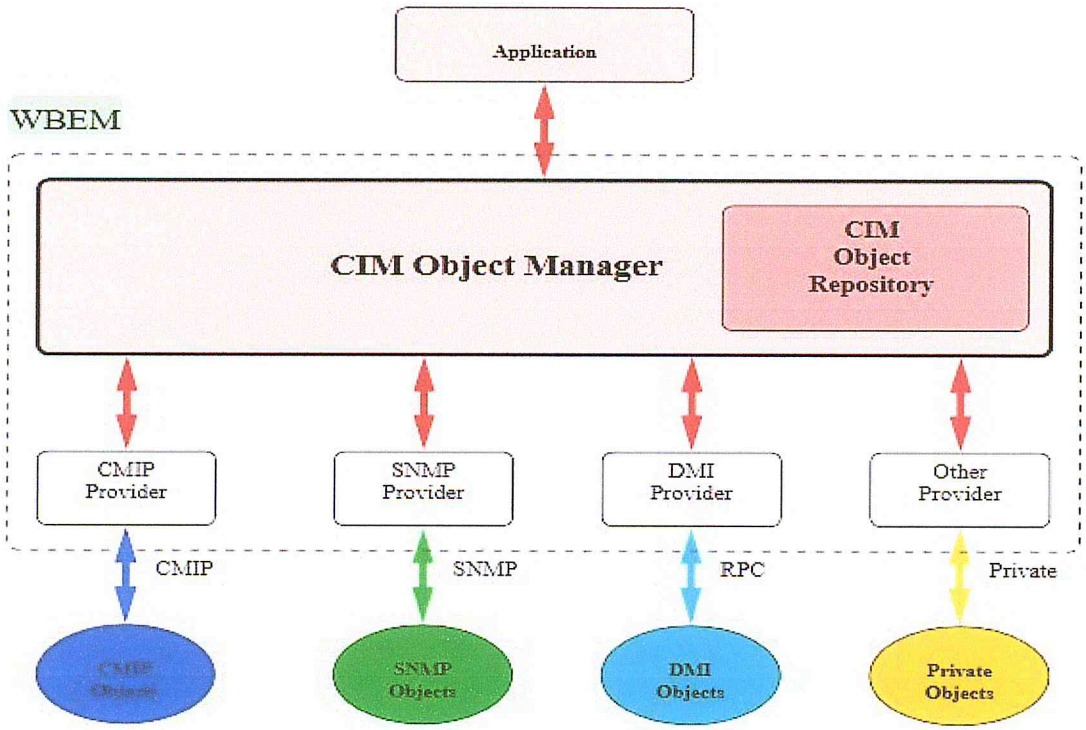

Figure 15: *Composants d'une architecture [12]*

Les Listeners sont qui réagissent à un type particulier de Providers, nommés « Indication Providers ». Ces Providers sont associés à un événement (l'extinction de l'ordinateur par exemple) et communique au serveur WBEM toute réalisation de cet événement. Les listener sont ensuite informés par le serveur WBEM de cet événement. (Voir les figures).

L'architecture définie offre une couche d'abstraction permettant d'assurer une gestion homogène centralisée d'un ensemble de ressources hétérogènes distribuées, tout en conservant les technologies existantes.

L'architecture WBEM présentée est purement théorique. Le choix de la distribution de cette architecture sur le réseau a été laissé à l'implémentation. Généralement, dans une implémentation réelle, les entités WBEM sont réparties de la façon suivante (figure) :

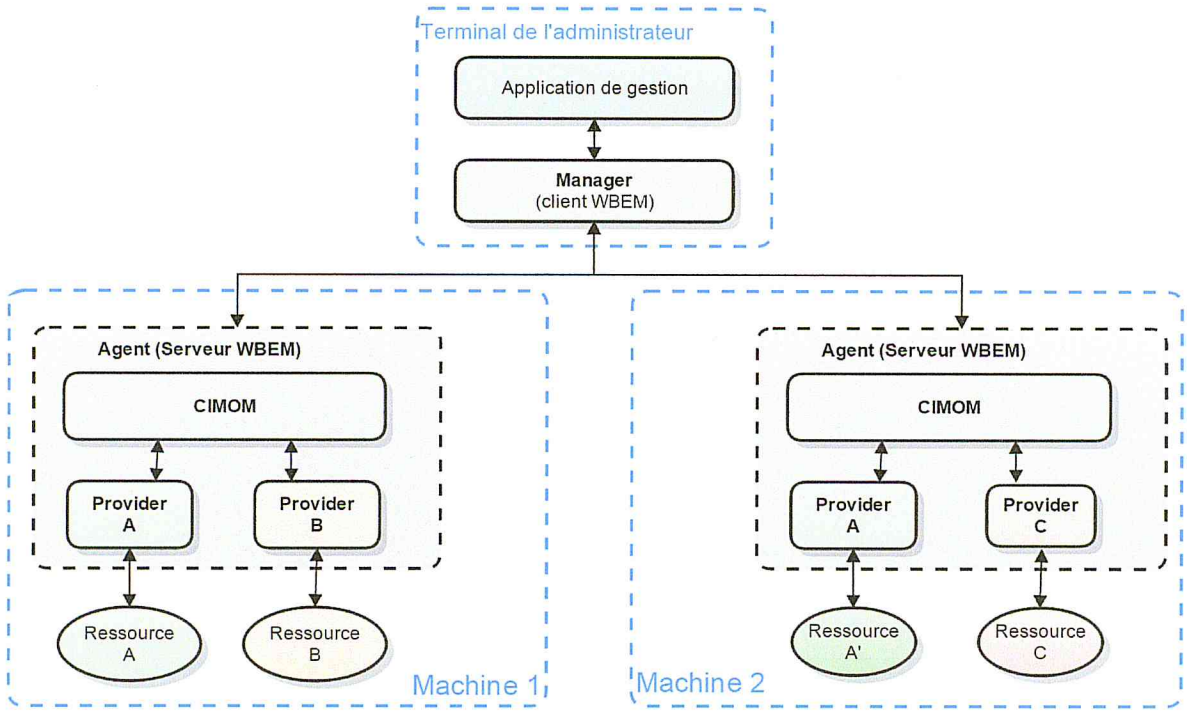

Figure 16 : Distribution pratique de l'architecture CIM.

Le serveur WBEM est l'entité qui regroupe généralement le gestionnaire d'objets CIMOM, le référentiel CIM (CIM Repository), les Providers, ainsi que des modules qui assurent la communication.

2.4. Le standard CIM :

2.4.1. Définition :

CIM (Common Information Model) est un standard ouvert publié par le DTMF en 1999 sous l'initiative WBEM qui définit comment les éléments d'un système sont modélisés, ainsi que les relations entre eux. C'est un méta modèle extensible qui permet d'organiser logiquement

les informations de gestion de manière unifiée et cohérente, même si l'environnement géré est hétérogène. CIM est un méta-modèle orienté objet qui tente d'unifier et d'étendre les standards de gestion existants, comme SNMP et CMPI, en les représentants avec le même méta-modèle. C'est à dire, qu'après la modélisation en CIM des systèmes (matériel et logiciels) issus de différents constructeurs, leur gestion ne nécessitera que des connaissances CIM au lieu d'apprendre les différentes API d'instrumentation offertes par les constructeurs et les éditeurs, ainsi tous les systèmes qui assurent les mêmes fonctions (OS, Routeurs, Switchs etc.) puissent être gérées de la même manière.

CIM est un élément clé qui permet aux développeurs d'applications et administrateurs de réseaux et de systèmes distribués de représenter et gérer les politiques à travers un éventail de domaines techniques, y compris le réseautage, la sécurité et l'administration du système.

Nous étudions dans cette section les différents concepts intégrés dans CIM lui permettant d'uniformiser l'information de gestion. [13]

2.4.2. Les éléments de base de CIM :

CIM offre une approche pour la surveillance et le management des équipements indépendamment du constructeur. CIM a pour but d'étendre et unifier les différents modèles de données comme SNMP, DMI, etc. Ci-dessous, une vue détaillée de CIM qui présente principalement quatre éléments :

1. **Un méta-modèle** décrivant les composants de base du modèle qui permettent la définition des entités gérées.
2. **Un schéma de nommage** des composants gérés.
3. **Un langage pour la spécification** des objets gérés.
4. **Un schéma de base et un schéma commun** qui sont des modèles génériques, devant être étendus pour la spécification de nouveaux systèmes.

2.4.2.1. Le méta-modèle :

Étant un méta-modèle orienté objet, CIM présente l'ensemble des éléments de base permettant de construire des spécifications. Ces éléments représentent les briques de base du modèle, dont certains sont similaires à des concepts utilisés dans la programmation orientée objet [DMTF]. Ces éléments sont :

- **Le schéma:** c'est un ensemble de spécifications constituées de classes, d'instances et d'associations modélisant un domaine de gestion. Un schéma peut être vu comme un module SNMP puisqu'il permet d'avoir une vue abstraite du domaine.

- **La classe:** permet de déclarer les propriétés communes d'un type d'objet. Elle est définie par des attributs dont les valeurs représentent l'état des objets de cette classe, ainsi que des méthodes qui représentent son interface d'interaction.
  Elle s'écrit de cette façon :

  class nom_de_la_classe
  {
  }

  La notion d'héritage existe en CIM et se symbolise de la façon suivante :

  class nom_classe_fille : nom_classe_mere
  {
  };

- **L'attribut:** représente une propriété de l'objet dans lequel il est défini. Chaque attribut est caractérisé par son type de données (entier, caractère, booléen, etc.) et un ensemble d'opérations (lire, affecter, etc.) que l'on peut exécuter sur lui.
- **La méthode:** définit une opération que l'on peut invoqué (exécuter) sur une instance d'objet. Une méthode comporte des paramètres et une valeur de retour.
  Elle s'écrit de la façon suivante :

  type_sortie nom_méthode (type entrée)

  Par exemple :

  uint32 AddNode ([IN] CIM_CP ref CS);

- **Les arguments de classe et de méthode :** Ils se déclarent de manière très simple :

  type_arg nom_arg;

Les types existants dans la spécification sont :

Nom du type	Sens
uint8	Entier 8-bit non signé
sint8	Entier 8-bit signé
uint16	Entier 16-bit non signé
sint16	Entier 16-bit signé
uint32	Entier 32-bit non signé
sint32	Entier 32-bit signé
uint64	Entier 64-bit non signé
sint64	Entier 64-bit signé
string	Chaîne de caractères UCS-2
boolean	Booléen
real32	Réel IEEE 4-byte
real64	Réel IEEE 8-byte
datetime	Chaîne de caractères contenant la date
<classname> ref	Référence
char16	Caractère UCS-2 sur 16 bit

Figure 17 : Les types existants (Tableau)

- **L'association :** Une association est une classe particulière permettant de spécifier des relations entre les ressources gérées. Une référence est un attribut particulier qui permet de pointer vers une instance d'une classe particulière. Une association possède donc au minimum deux attributs de type « référence ».Une association est caractérisée par le qualificateur « Association ». Elle représente un lien entre deux objets. Exemple : l'association « InstalledOS » crée le lien entre le matériel et le système d'exploitation.
  Les associations sont des éléments clés dans CIM, en effet ils permettent d'atteindre des objets CIM inconnus à partir d'objets connus.
- **L'instance:** c'est une occurrence particulière d'une classe ou d'une association. Plusieurs instances peuvent être issues de la même classe, on parle d'instanciation de classe.
- **Le qualificateur :** c'est une métadonnée qu'on associe à un composant (classe, attribut, méthode ou instance) afin de lui ajouter certaines spécificités. Parmi les qualificateurs citons:
  1. **Description :** Peut être associé à tous les composants. Il permet de leur ajouter une description textuelle.
  2. **Key :** Associé aux attributs clé d'une classe, permettant de définir d'une manière unique les instances issues de cette classe.
  3. **Valuemap/Values :** Deux listes contenant toutes les valeurs possibles d'un attribut.

4. **Association :** Indique que la classe est une association
5. **Version :** Précise la version du schéma dont la classe fait partie
6. **« Override » :** Indique qu'il s'agit d'une surcharge d'un argument hérité d'une classe mère
7. **Required :** Indique que l'argument doit obligatoirement être renseigné lors de l'instanciation de la classe.
8. **La référence:** Est un attribut pointeur dans une association. Il indique le rôle de chaque objet dans une association. Elle est indiquée par le mot REF. Par exemple : **CIM_OperatingSystem REF Antecedent**

Voici un exemple, donné par la DMTF, d'une classe et d'une instance. On y retrouve la majorité des éléments décrits ci-dessus :

Exemple de classe	Example d'instance pour cette classe

[Version ("2.7.0"), Experimental, Description (
"A CIM is a type of CIM_WBEMService "
"that instruments one or more aspects of the CIM Schema. "
"A CIM_Provider operates at the request of the "
"CIM_ObjectManager to perform operations on CIM objects. "
"The properties CreationClasName, SystemCreationClassName "
"and SystemName can be set to empty strings. In this case, "
"the CIM Object Manager must interpret the properties with "
"the local system information.")]
class CIM_Provider :CIM_WBEMService {
 	
[Override ("Name"), Description (
"A human-readable name that uniquely "
"identifies the provider within a system.")]
string Name;
 	
[Required, Description (
"An implementation specific string that identifies the "
"handle to the provider.")]
string Handle;};
	instance of CIM_Provider {
	Name = "ACME_OperatingSystemProvider";
	Handle = "ACME_OperatingSystemProvider";
	};
	
	instance of CIM_ProviderCapabilities {
	ClassName = "CIM_OperatingSystem";
	ProviderType = { 2 };
	SupportedProperties = NULL;
	SupportedMethods = NULL;
	};

Figure 18 : Exemple d'une classe et une instance (Tableau) [13]

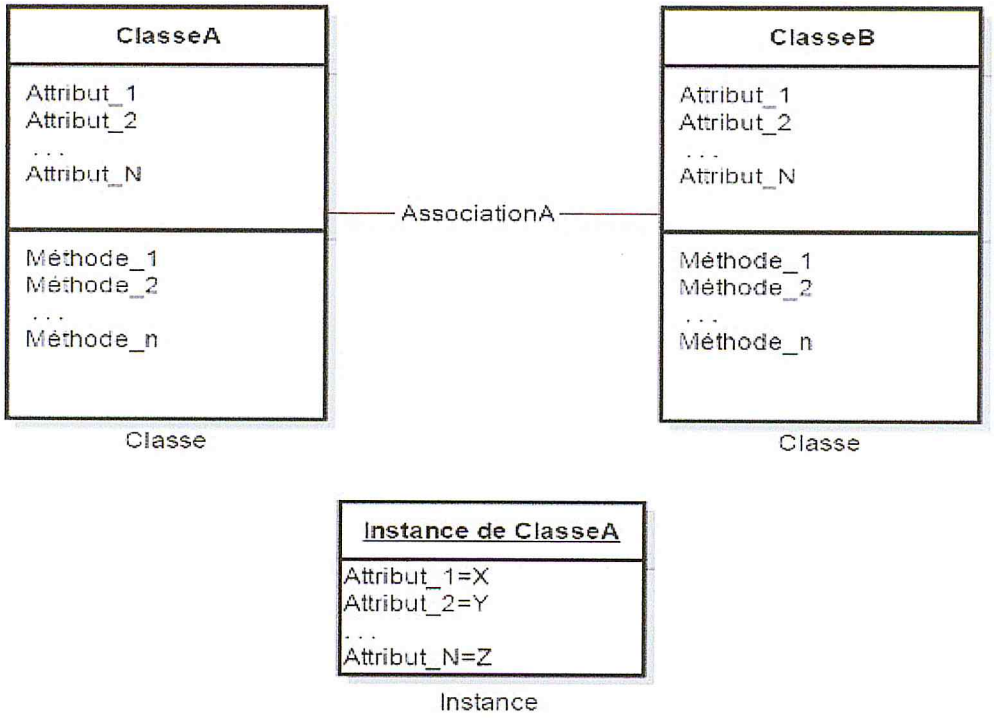

Figure 19 : Les éléments de base du méta-modèle CIM

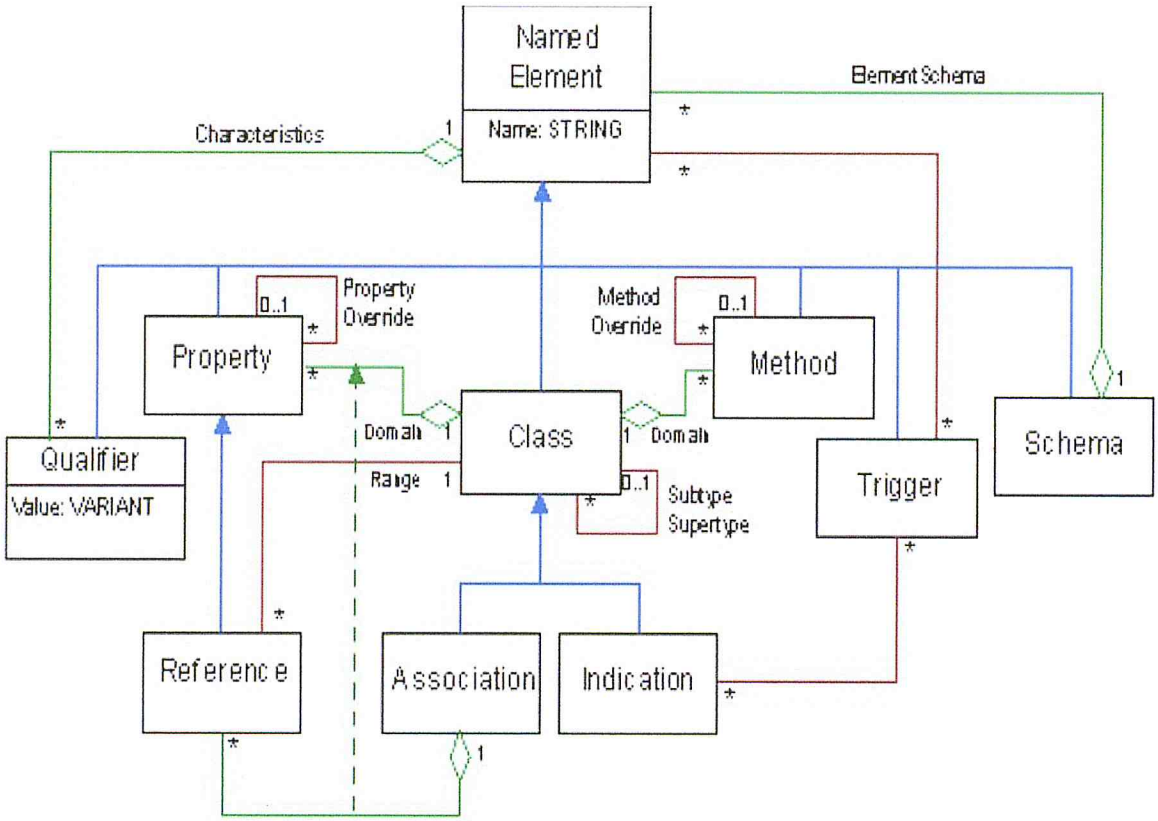

Figure 20 : Le méta modèle CIM [13]

Enfin, un déclencheur est une opération invoquée lors du changement d'état d'un objet géré.

Une indication est une classe particulière dont les instances sont créés par des déclencheurs.

Les éléments du méta-modèle sont spécifiés à l'aide d'un langage de spécification textuel appelé MOF (Managed Object Format) qui sera détaillé dans la section 4.2.3 Par ailleurs le langage visuel UML est adopté afin de représenter d'une manière synthétique (et tout aussi expressive que MOF) les détails d'un modèle en précisant les associations et les qualificateurs de chaque élément. [13]

2.4.2.2. L'espace de nommage et l'architecture de la MIB :

Dans l'architecture CIM, les objets gérés, représentés à l'aide d'un ensemble de classes et de relations entre elles, sont maintenues par le gestionnaire d'objets CIMOM. Ces objets représentent la base d'information MIB appelée CIMOR dans le contexte WBEM.

L'architecture de cette base d'information repose sur le principe d'espace de nommage. Un espace de nommage est un conteneur d'objets (classes, instances et relations) dans lequel chaque composant est identifié d'une manière unique. Un espace de nommage peut contenir d'autres espaces de nommage, ce qui donne une arborescence de nommage similaire à l'arborescence des répertoires sous UNIX.

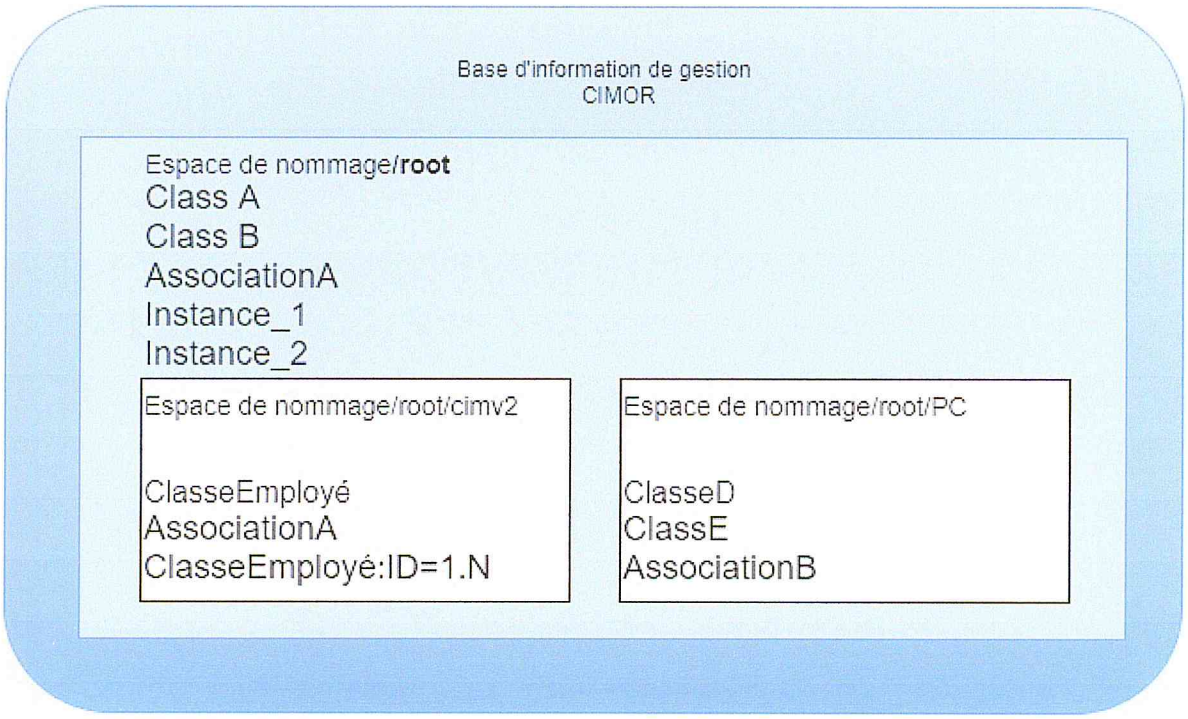

Figure 21 : Exemple des composants d'un espace de nommage. [14]

2.4.2.3. Le langage de spécification :

Dans l'architecture WBEM, le langage utilisé pour la spécification est le *MOF (ManagedObject Format)*. Ce langage fournit un support textuel pour la formalisation des modèles de l'information et propose une syntaxe et une sémantique permettant de spécifier les composants d'un modèle CIM en se basant sur des éléments du méta-modèle (classes, associations, etc.).

MOF est un langage de spécification orienté objet dérivé du langage IDL (Interface Definition Language) défini par l'OMG (Object Management Group) principalement utilisé par les applications CORBA. La syntaxe et la sémantique ne sont pas détaillées dans ce chapitre.

Voici un exemple de code MOF représentant la classe ClasseEmployé :

```
[Description ("classe représentant un employé")]                    // qualificateur description

class ClasseEmployé {

[Key, Description ("Identifiant de l'employé")]                     // qualificateur clé et
description

uint16 ID;                                                          // entier non signé de taille
16 bits

[Key, Description ("Nom de l'employé")]

string Nom;                                                         // chaîne de caractères

[Description ("'Méthode pour changer l'ID de l'employé")]

boolean SetNouveauID (uint16 NouveauID);                            // méthode qui prend un ID en

// argument et qui retourne un

// booléen
};
```

Figure 22 : *Exemple du langage MOF*

Un schéma est un ensemble de spécifications de classes, d'instances, etc. regroupées par rapport à une thématique. Par exemple, System est le schéma correspondant à la modélisation du domaine des systèmes informatiques.

Les classes, propriétés, méthodes et instances ont la signification classique de l'approche objet.

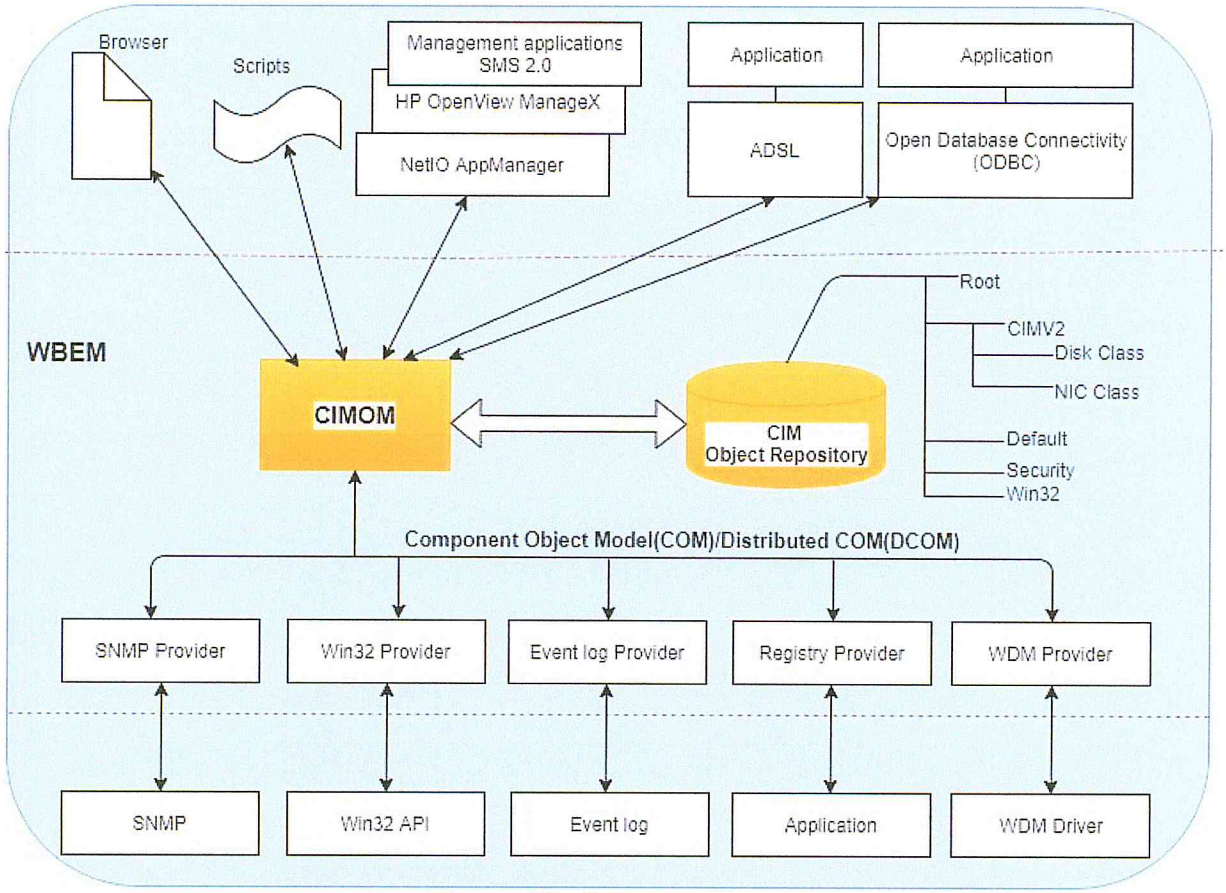

Figure 23 : *Structure CIM*

2.4.2.4. Le schéma CIM :

Le standard CIM contient 2 parties : les spécifications (la structure du schéma, et le langage avec lequel il est écrit) et le schéma CIM de base.

Le schéma CIM est un schéma orienté objet. Il contient un nombre assez important de classes donnant des informations matérielles et logicielles sur le système, ainsi que sur les relations entre ces éléments. Il représente ainsi aussi bien les systèmes sur le réseau, que les bases de données, les processus,... et ce de façon générique, évitant ainsi tout problème de compatibilité.

Il est découpé en trois modèles : un **modèle de cœur** qui contient les classes de bases, il permet de définir les classes et les associations génériques pour toutes les classes

domaine de gestion, un **modèle commun** qui englobe les classes représentant des aspects logiciels ou matériels précis, héritant des classes du modèle de cœur, et un **modèle d'extension**, qui contient des classes spécifiques à un environnement précis ou à un contexte de travail, et qui sont vouées à long terme à entrer dans le modèle commun (figure 8). Il est possible de rajouter des classes à ce schéma en respectant les spécifications.

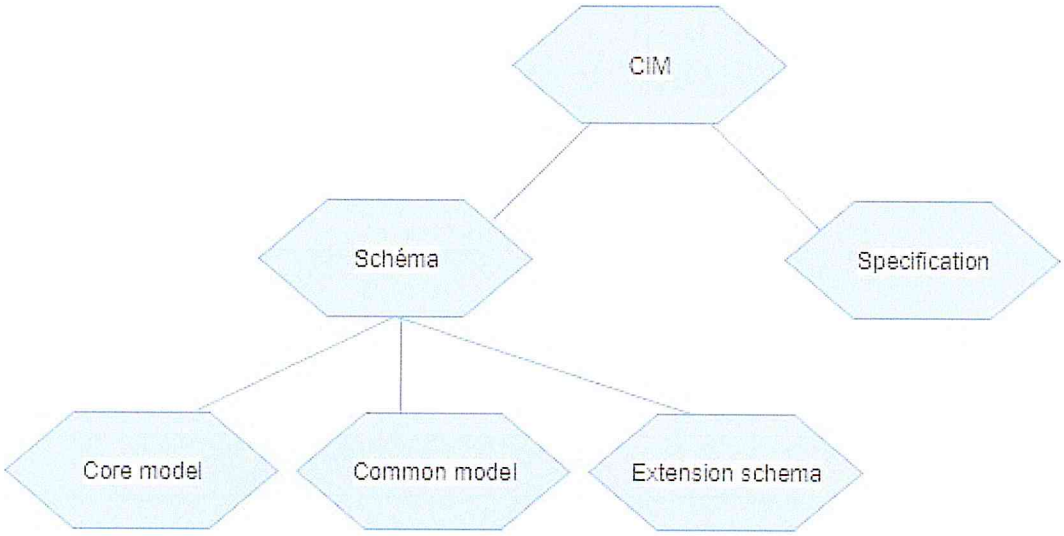

Figure 24 : Structure du standard CIM

Le modèle commun

Afin d'unifier la gestion des ressources gérées et d'assurer leur interopérabilité avec l'architecture WBEM, le DMTF a défini deux schémas de référence sur lesquelles doit s'appuyer tout travail de modélisation en CIM. Ces deux schémas, dont chacun représente un certain niveau d'abstraction, sont : le schéma de base (Core Schema) et le schéma commun (Common Schema).

2.4.2.5. Le modèle de base :

Le modèle de base donne une modélisation minimale des ressources de l'environnement dont l'objectif est d'être adaptée à toute ressource gérée.

Toute classe CIM hérite de la classe abstraite CIM_ManagedElement qui représente le niveau d'abstraction le plus haut dans CIM.

Les éléments gérés sont réellement considérés à partir de la classe abstraite CIM_ManagedSystemElement qui spécialise CIM_ManagedElement.

Un élément géré peut être considéré comme un composant physique avec une réalité matérielle ou un composant logique dans le cadre de la gestion grâce aux classes abstraites CIM_PhysicalElement et CIM_LogicalElement qui spécialisent CIM_ManagedSystemElement.

Les objets sont représentés dans des fichiers au format MOF (Managed Object Format).

Ses différents éléments sont les suivants :

- **Le modèle commun :**

  Qui est une extension du modèle de base. Il rassemble un ensemble de spécifications regroupées par domaines de gestion. Il comprend 13 schémas. Parmi lesquels nous pouvons citer :

  - Le modèle physique permet de modéliser la composition physique des éléments gérés;
  - Le modèle politique qui est raffiné par le modèle politique IPsec ;
  - Le modèle utilisateur représente en tant qu'objet géré un utilisateur en incluant entre autre le modèle RBAC.

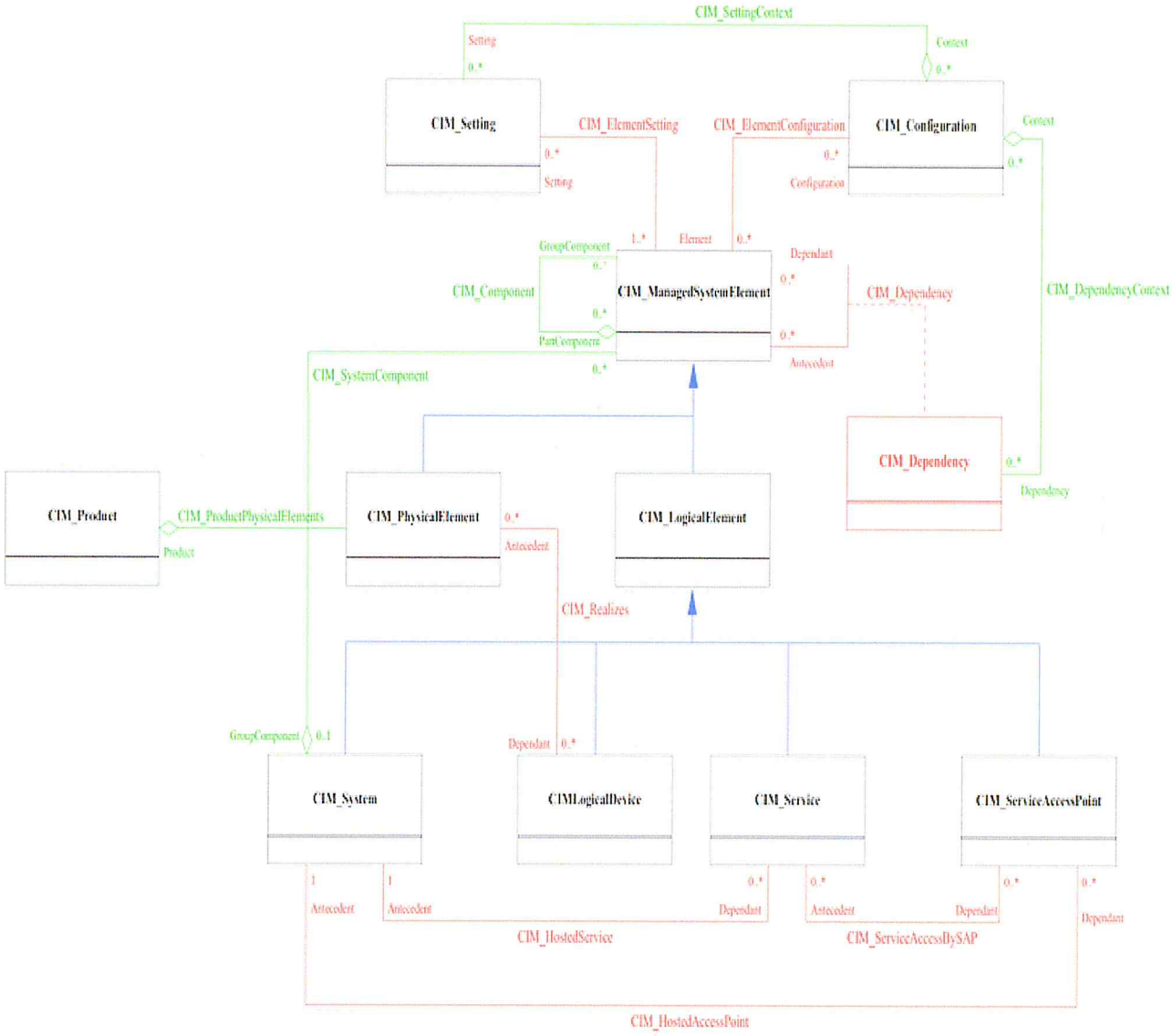

Figure 25 : Diagramme de classes UML définissant le modèle de base [13]

Ce méta-modèle a été proposé par le DMTF pour que tous les méta-modèles dérivés soient unifiés en offrant un maximum d'homogénéité. Pour cette raison, toute modélisation d'une entité doit obligatoirement être dérivée de ce schéma ou d'un de ses dérivés.

2.4.2.6. Le schéma commun

Ce schéma est une extension du schéma de base. Il maintient un niveau d'abstraction inférieur à celui de ce dernier car, il présente des spécifications applicables à des domaines spécifiques. Ce schéma se décompose en 13 sous-schémas associés aux domaines de gestion suivants [13]

2.4.3. Le modèle de communication :

Afin de véhiculer les informations de gestion CIM entre les différentes entités de l'architecture WBEM, le DMTF propose un protocole de communication fondé sur deux standards du web : le langage XML (eXtensible Markup Language) et le protocole http (Hyper-Text Transport Protocol).

En plus de ces deux standards, le DMTF propose un ensemble standard d'opérations de gestion appelées opérations CIM permettant l'accès et la manipulation des données CIM. [13]

2.4.3.1. Le protocole de communication XML/http :

Ce protocole est un couplage des deux standards XML et HTTP. Le premier est utilisé pour encoder l'information CIM alors que le second est utilisé pour le transport proprement dit.

La figure suivante représente une communication client/serveur entre deux entités WBEM

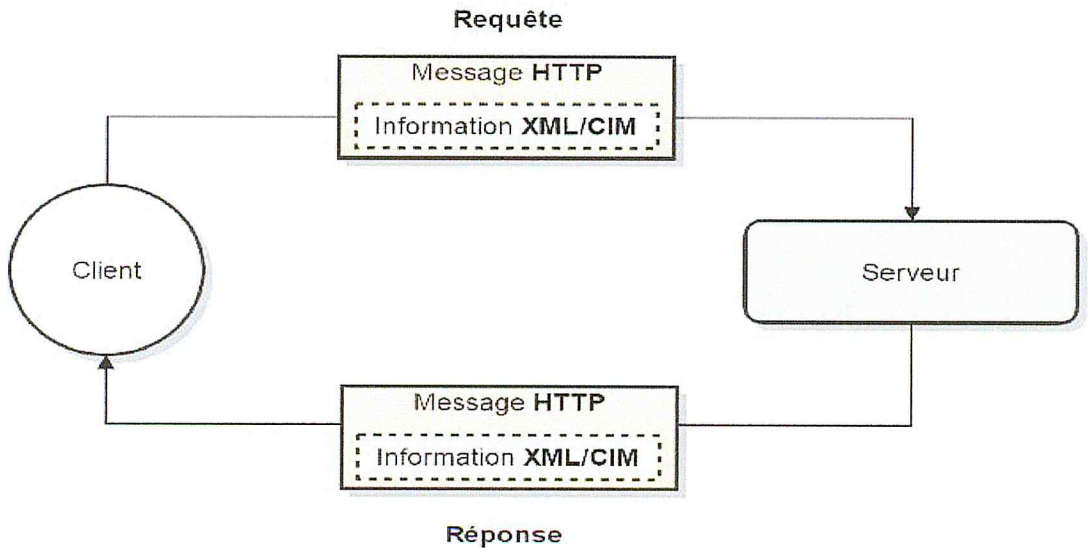

Figure 26 : Communication client/serveur WBEM [14]

Au niveau de l'implémentation, cette communication est assurée par des modules dédiés qui sont :

- Le serveur HTTP et l'encodeur/décodeur CIM au niveau du serveur CIM.
- Le client HTTP et l'encodeur/décodeur **CIM au niveau du client CIM.**

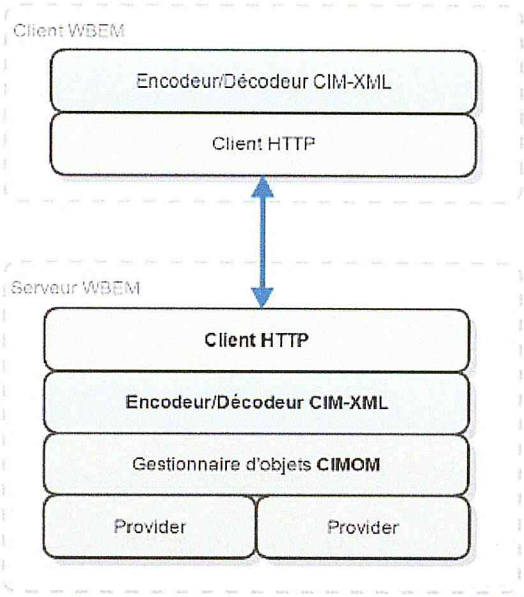

Figure 27 : Modules de communication WBEM.

2.4.3.2. Les opérations CIM :

Les opérations CIM sont définies comme des appels de méthodes sur des objets gérés. Bien entendu, ces appels seront traduits par des requêtes de bas niveau sur les véritables entités gérées. [13]

Deux types d'opérations CIM sont distingués : **les opérations intrinsèques et les opérations extrinsèques. [14]**

- **Les opérations Intrinsèques :** Permettent au client CIM de demander au serveur CIMOM, d'obtenir ou de modifier l'état des objets de l'espace de nommage cible (classes, instances, etc.). Conceptuellement, ces opérations correspondent aux opérations de gestion classique de type M-GET et M-SET dans le protocole CMIP proposé par l'ISO. Les opérations intrinsèques sont classifiées en 7 familles, leur implémentation dépend du modèle fonctionnel. C'est à dire, chaque modèle fonctionnel d'un système géré nécessite l'implémentation d'au moins une famille. Les familles d'opérations sont :

1. **Les opérations de lecture de base (Basic Read) :** elles regroupent les opérations de lecture simple des données maintenues par un gestionnaire d'objets CIMOM. Parmi les opérations les plus importantes on retrouve :
   - **GetClass :** retourne la définition d'une classe implantée dans l'espace de nommage cible. Elle prend en argument obligatoire le nom de la classe souhaitée.
   - **EnumerateClasses :** retourne la liste de toutes les classes définies dans l'espace de nommage cible, ou seulement les classes dérivées d'une classe donnée comme argument.
   - **GetInstance :** retourne la définition de l'instance donnée comme argument.
   - **EnumerateInstances :** retourne l'énumération de toutes les instances d'une classe.
2. **Les opérations d'écriture de base (Basic Write) :** cette famille d'opérations contient une seule opération qui est la modification de la valeur d'une propriété d'une instance donnée. Elle doit être implémentée dans tous les systèmes utilisant les opérations de lecture de base. Cette opération est :
   - **SetProperty :** permet de d'affecter une nouvelle valeur à une propriété d'une instance donnée, dans **l'espace de nommage cible.**
3. **Les opérations de manipulation de schémas (Schema Manipulation) :** Regroupent les opérations qui permettent de créer, supprimer ou modifier une classe dans l'espace de nommage cible. Cette famille contient 3 opérations :
   - **CreateClass :** permet de créer une nouvelle classe dans l'espace de nommage cible.Elle requière comme argument la définition en langage MOF de la nouvelle classe.
   - **ModifyClass :** permet de modifier la définition d'une classe dans l'espace de nommage cible. Elle requière comme argument la définition en langage MOF de la classe modifiée.
   - **DeleteClass :** permet de supprimer la définition d'une classe dans l'espace de nommage cible. Elle requière donc comme argument le nom de cette classe.
4. **Les opérations de manipulation d'instances (Instance Manipulation) :** Similaire à la famille précédente, cette famille contient les opérations : **CreateInstance, ModifyInstance** et **DeleteInstance**, qui sont invoquées

respectivement dans le but de créer, modifier ou supprimer des instances d'objets dans l'espace de nommage cible.

5. **Les opérations de parcours d'associations (Association Traversal)** : Elles permettent de lancer des requêtes sur des définitions et des instances d'associations (relations) entre les objets CIM. Il s'agit ici d'opérations très puissantes qui font toute la force de l'approche CIM/WBEM. En effet elles permettent à partir d'une instance définie et du nom d'une association, données en argument, d'identifier la classe et toutes instances dérivées associées à l'instance de départ. Comme exemple, il est possible d'identifier tous les composants physiques et logiques d'un ordinateur (instance de CIM_ComputerSystem) à travers les associations qu'il possède avec les différentes classes (CIM_Processor, CIM_Memory, CIM_NetworkPort etc.). L'opération la plus importante est Associators : elle permet de retourner l'ensemble des classes et des instances en relation avec l'objet (classe ou instance) donné en argument. Un autre paramètre peut être donné pour spécifier l'association qui assure les relations. C'est à dire, ce paramètre permet d'afficher les objets qui ont l'association donnée en paramètre, avec l'objet concerné.
6. **Les opérations d'exécution de requêtes (Query Execution) :** Cette famille, un peu spéciale, comporte une seule opération qui est une demande d'exécution d'une requête d'interrogation CIM sur la base CIMOR. Il s'agit là encore d'un aspect très puissant de l'approche CIM/WBEM qui permet de sélectionner des objets CIM (classes ou instances) de la même manière qu'une requête SQL sélectionnera des lignes de tables sur les bases de données relationnelles. Ce sont des requêtes multicritères sur tout type d'attributs (clé ou simple) qui vont permettre ensuite d'exécuter des opérations de base sur les résultats. Cette opération est ExecQuery : elle permet à un client de lancer des requêtes sur les objets maintenus dans l'espace de nommage cible. Cette opération prend comme arguments le nom du langage de la requête suivi de la requête de sélection elle-même. WBEM définit pour le moment un seul langage de requêtes appelé le CQL (CIM Query Language).
7. **Les opérations de déclaration de qualificateurs (Qualifiers Declaration)** : elles permettent de lancer des requêtes sur des qualificateurs. Parmi ces opérations on retrouve :

- **GetQualifier :** permet de demander la définition d'un qualificateur dans l'espace de nommage ciblé.
- **SetQualifier :** permet de demander la création ou la modification d'une définition de qualificateur dans l'espace de nommage ciblé.
- **DeleteQualifier :** permet de demander la suppression de la définition d'un qualificateur dans l'espace de nommage ciblé.

8. **Les opérations Extrinsèques** : Elles représentent les méthodes pouvant être invoquées sur n'importe quel type d'objets. Ce sont les méthodes définies dans les classes CIM.

2.5. Conclusion :

Dans ce chapitre, nous avons présenté les différents concepts du standard de gestion homogène CIM/WBEM dédié à la gestion d'entités hétérogènes complexes (composants physiques, services, utilisateurs, etc.). D'abord, nous avons présenté l'architecture fonctionnelle de WBEM. Ensuite, nous avons abordé les composants de base du modèle de l'information commun (CIM). Après, nous avons étudié le modèle de communication utilisé dans ce standard. L'aspect sécurité ainsi que la gestion des politiques ont été discuté.

L'étude faite dans ce chapitre nous a permis de comprendre la puissance d'expression du standard WBEM/CIM qui est considéré comme l'un des standards les plus prometteurs du domaine. Nous avons choisi WBEM pour la réalisation de notre projet, à cause de sa capacité de gérer des entités complexes, vu que la gestion des politiques n'est pas un aspect supporté par n'importe quelle approche. De plus, l'interopérabilité offerte par WBEM répond à notre besoin de réalisation d'un système de gestion pouvant être intégré à d'autres systèmes.

Chapitre III

Conception du système

3.1. Introduction :

Nous rappelons que l'objectif de notre travail est la réalisation d'un outil normalisé pour la gestion des communications sécurisées en IPSec. Pour atteindre cet objectif, WBEM a été choisi pour satisfaire nos besoins en terme de standards de gestion (modélisation des règles, leur distribution dans le réseau, etc.).

Bien que WBEM est aujourd'hui l'un des standards de gestion les plus prometteurs, l'implémentation de Providers permettant de gérer n'importe quel composant d'un système reste son plus grand défi. Nous avons donc décidé de réalisé notre propre provider « IPsecPro » qui va nous permettre de traduire le schéma CIM vers des actions concrètes qui vont permettre la gestion des différentes politiques de IPsec.

3.2. Système de gestion des politiques :

3.2.1. Définition :

La gestion basée sur les politiques est une technologie qui peut simplifier la tâche complexe de gestion des réseaux et des systèmes distribués. Dans ce paradigme, un administrateur peut gérer différents aspects d'un réseau ou d'un système distribué de manière souple et simplifiée en déployant un ensemble de politiques qui régissent son comportement, ces politique sont des règles indépendantes de la technologie visant à améliorer les fonctionnalités codées de gestion En introduisant une logique interprétée qui peut être modifiée dynamiquement sans modifier la mise en œuvre sous-jacente. Cela permet un certain degré de programmation sans qu'il soit nécessaire d'interrompre le fonctionnement du système géré ou du système de gestion lui-même. La gestion basée sur les politiques peut augmenter considérablement les aspects autogérés de tout système ou du réseau distribué, ce qui conduit à un comportement plus autonome démontré par les systèmes informatiques autonomes.[16]

3.2.2. La gestion à base de politique :

L'objectif premier de l'approche de gestion à base de politique est de rendre dynamique un certain nombre de tâches de gestion sans stopper ou recoder le système. L'intérêt principal de cette démarche consiste dans la séparation des règles gouvernant un système des fonctionnalités offertes par ce système. Il y a donc clairement une distinction entre la gestion du système et le système lui-même.

La démarche générale est de partir des objectifs généraux et en découler des règles compréhensibles par les équipements. Nous décrivons donc dans cette section les différents points de vue existants sur ce concept ainsi que l'architecture proposée par l'IETF pour son déploiement.

3.2.3. Gestion des politiques dans WBEM :

Contrairement aux anciens standards de gestion, qui permettent généralement de gérer des entités relativement simples, WBEM apporte non seulement la possibilité de gérer des systèmes complexes mais aussi, la possibilité de gérer des entités abstraites. Parmi ces entités, la gestion des politiques pour laquelle le DMTF propose le modèle CIM policy déjà vu dans la section 2.4.2.4. Les politiques modélisées peuvent être vues comme des configurations du système permettant de contrôler son comportement. Comme le cas des politiques de contrôle d'accès et les politiques de qualité de service [10].

3.3. L'architecture de notre système:

En s'inspirant des principes de l'architecture et du protocole de gestion de politiques appelé COPS (Common Open Policy Service) un standard de l'IETF, pour la définition et l'application de nos politiques avec ces quatre composants [17] [18]:

- **Policy Administration Point:** appelé PAP et représenté par un client CIM/WBEM, il fait appel aux services du PDP (ci-après) afin de définir, distribuer et appliquer des politiques
- **Policy Decision Point**: appelé PDP, représenté par le serveur CIMOM qui maintient les politiques (IPsec) à appliquer dans le système géré. Concrètement, les règles sont définies par l'administrateur via un client CIM, et peuvent être activées et désactivées par la suite.
- **Policy Respository Point:** appelé PRP représente l'entité qui stocke les politiques et permet de les lire. Dans notre cas c'est le serveur CIMOM qui constitue le point d'entrée au référentiel CIMOR contenant les politiques sous la forme d'instances CIM.
- **Policy Enforcement Point:** appelé PEP représente l'entité responsable de l'application des règles définies par le PDP, dans notre cas c'est le Provider WBEM IPsecPRO.

Dans cette architecture, le client WBEM (ou le PAP) définit une politique (ensemble de règles) et la stocke sous format CIM dans la base CIMOR maintenue par le serveur WBEM (le PDP). Avant son application par IPsecPRO (le PEP), cette politique est traduite en règles IPSec

brutes. La traduction CIM/IPSEC est assurée par le serveur WBEM, plus exactement par le Provider IPsecPRO (PEP aussi) que nous avons développé. Ce provider assure aussi la traduction inverse IPSEC /CIM afin de pouvoir superviser l'état de IPsecPRO.

Comme tout autre provider WBEM, le nôtre possède deux interfaces :

- **Une interface CIM** qui permet d'interagir avec le CIMOM afin de lire les politiques définies et stockées dans la base CIMOR, et de présenter l'état du système sous format CIM.
- **Une interface IPsec** qui permet d'interagir avec le système IPSEC sous-jacent afin de lui transmettre les règles IPSEC brutes appliqué. Elle permet aussi de récupérer l'état d'IPsecPRO ainsi que celui des ressources qu'il contrôle.

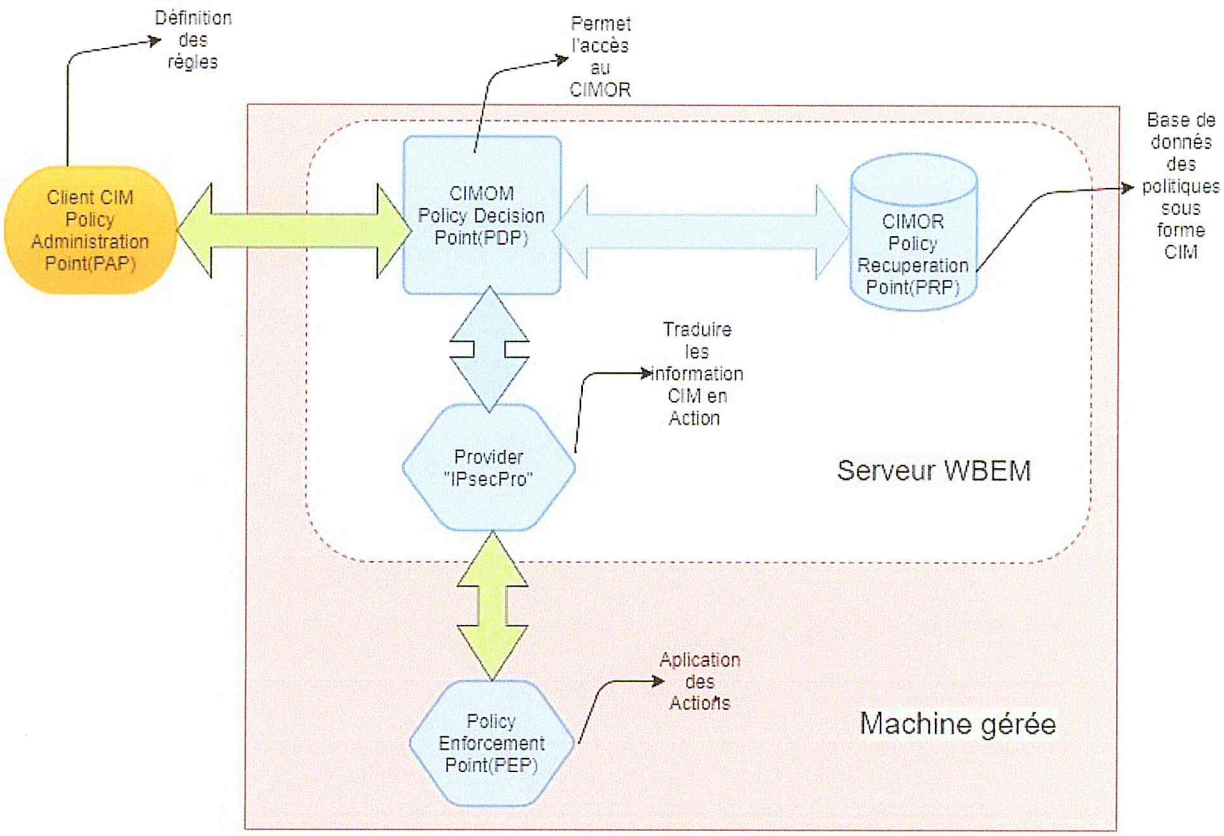

Figure 28 : *Architecture de notre système de gestion.*

Le provider IPsecPRO est en réalité un ensemble de modules dont chacun maintient une classe de notre modèle CIM. C'est à dire, chaque module assure l'exécution des méthodes d'une classe donnée ainsi que la gestion de ses instances (modifier et retourner les valeurs des attributs).

3.4. Modélisation fonctionnelle :

La modélisation fonctionnelle nous permettra de définir une abstraction du système d'un point de vue fonctionnel en identifiant les acteurs, leurs différentes façons d'utiliser le système, d'établir précisément les frontières du système et de déterminer les cas d'utilisation et leurs descriptions.

Dans cette section, nous allons présenter les différents scénarios d'utilisation de notre Provider en utilisant des diagrammes de cas d'utilisations.

3.4.1. Digramme de cas d'utilisation « créé une politique IPSec » :

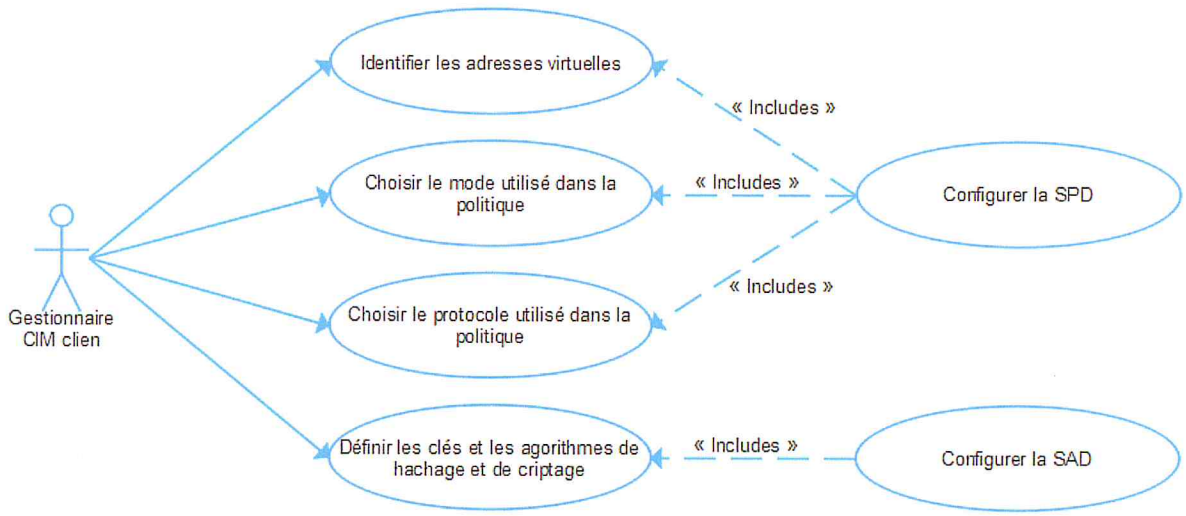

Figure 29 : *Les paramètres nécessaires pour créer une politique IPSec*

Ce diagramme de cas d'utilisation permet de définir les taches principales qui permettent la création d'une politique IPSec, donc pour une nouvelle politique le client doit définir les composant nécessaires pour configurer la SPD qui inclue les adresses virtuels ,le mode transport ou tunnel utilisé dans la politique, le protocole utilisé AH ou ESP ,par la suite il doit définir une configuration pour la SAD si la SA correspondant à la politique enregistrée dans la SPD n'existe pas encore, cette configuration qui inclue les algorithme de cryptage et de hachage qui sont aussi choisis par le client.

1.4.2. Digramme de cas d'utilisation «gérer les bases de données»

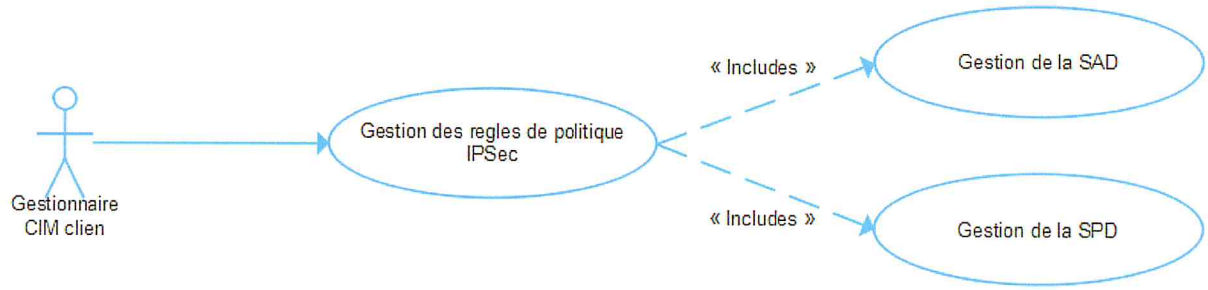

Figure 30 : *Les bases de données gérées par le Client CIM*

Dans notre travail le Client CIM est responsable de la configuration des deux bases de données SPD et SAD, car nous nous sommes limité à la configuration statique (manuelle) pour des SAs qui est d'habitude gérée par le protocole IKE et le client ne définie que la configuration de la SPD.

1.4.3. Digramme de cas d'utilisation « Modifier une configuration politique IPSec » :

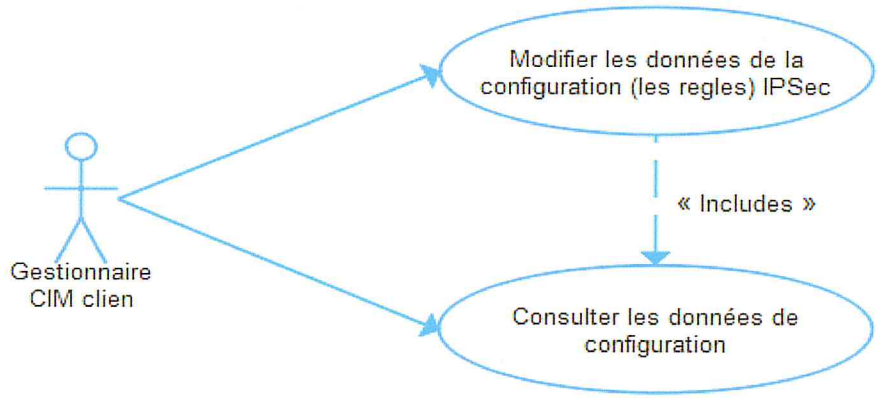

Figure 31 : *Gestion de la configuration d'IPSec*

Le Client CIM peut consulter et modifier les données de la configuration IPSec qui se trouvent dans la SAD et la SPD.

1.5. Schéma illustratif pour le fonctionnement de notre protocole IPSec :

Le schéma suivant représente le fonctionnement du protocole IPsec sans prendre en considération la négociation IKE.

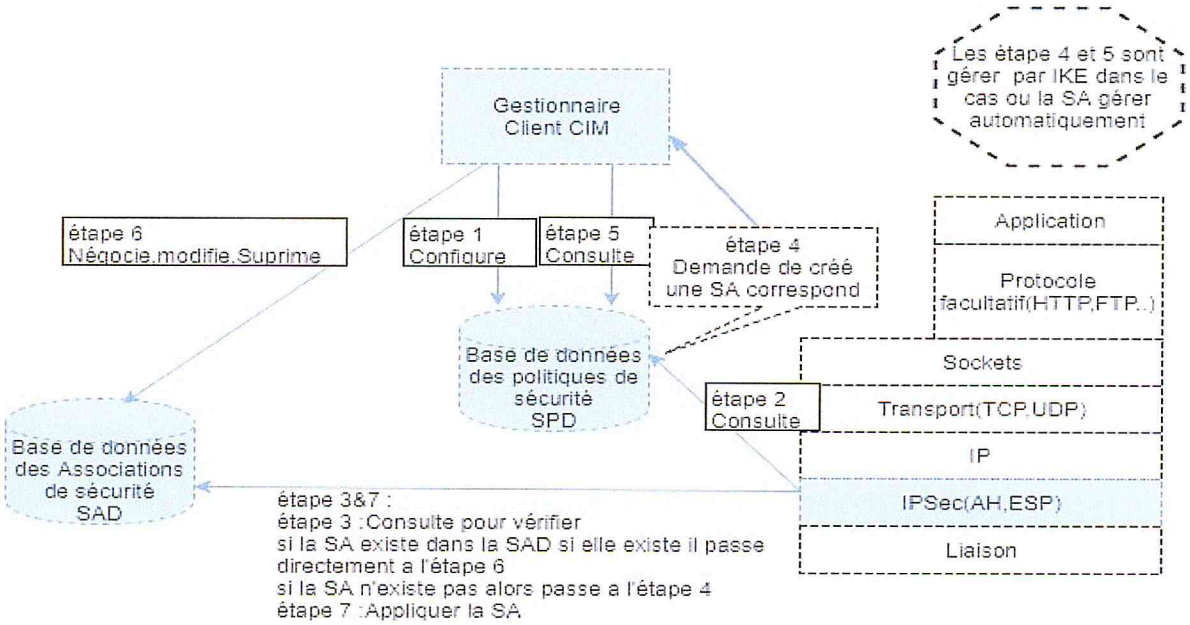

Figure 32 : *Schéma détaillé pour la définition d'une nouvelle politique IPSec*

Donc pour établir une communication IPSec pour un trafic sortant notre client CIM commence par configurer la SPD, cette base de données permet de décider, pour chaque paquet, s'il se verra apporter des services de sécurité (étape1), s'il sera autorisé à passer ou rejeté(étape1), 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 (étape2), Si cette base lui indique que le trafic doit se voir appliquer des mécanismes de sécurité, elle récupère les caractéristiques requises pour la SA correspondante et va consulter la base des SA (SAD) (étape 3). Si la SA nécessaire existe déjà, elle est utilisée pour traiter le trafic en question (et donc les étapes 6 at 7 s'applique). Dans le cas contraire, l'administrateur doit créer une nouvelle SA correspond à la SPD dans la SAD avec les caractéristiques (l'étape4 ensuite l'étape 5 et 6) il termine par appliquer la SA sur le trafic.

Dans le cas d'un trafic entrant lorsque la couche IPSec reçoit un paquet en provenance du réseau, elle examine l'entête pour savoir si ce paquet s'est vu appliquer un ou plusieurs services IPSec et si oui, quelles sont les références de la SA et répéter les étapes précèdent.

1.6. Modélisation CIM :

Suivant les recommandations du DMTF exigeant que toute modélisation de système géré doivent être conforme à l'un des profils proposés, nous avons repris et étendu un profil CIM répondant à nos besoins

Dans la terminologie CIM, on appelle un schéma, tout modèle (ou un ensemble de modèle) proposé par une organisation pour satisfaire ses besoins en terme de gestion. Tous les noms des classes appartenant à un schéma donné, comportent un préfixe commun appelé schéma spécifique. Par exemple, les noms des classes appartenant aux modèles CIM proposés par le DMTF commencent tous par le schéma spécifique "CIM_". Ainsi, pour notre modèle, nous avons défini "IPsec_" comme schéma spécifique. Ceci implique que toutes nos classes commencent par le préfixe : IPsec_.

Avant d'établir un tunnel IPSEC, il faut commencer par le configurer (ou le modéliser) en fournissant toutes les informations nécessaires à l'implémentation IPSEC sous-jacente pour le réaliser. Ceci passe par la création d'un ensemble d'instances de classes CIM reliées entre elles par des associations adéquates, voici donc l'ensemble des classes utilisée dans notre modèle:

3.6.1. Les Classes de politique (Policy Class) :

Les classes de politique IPsec représentent l'ensemble des stratégies qui permettent de définir les politiques définies dans le PDP/PRP.

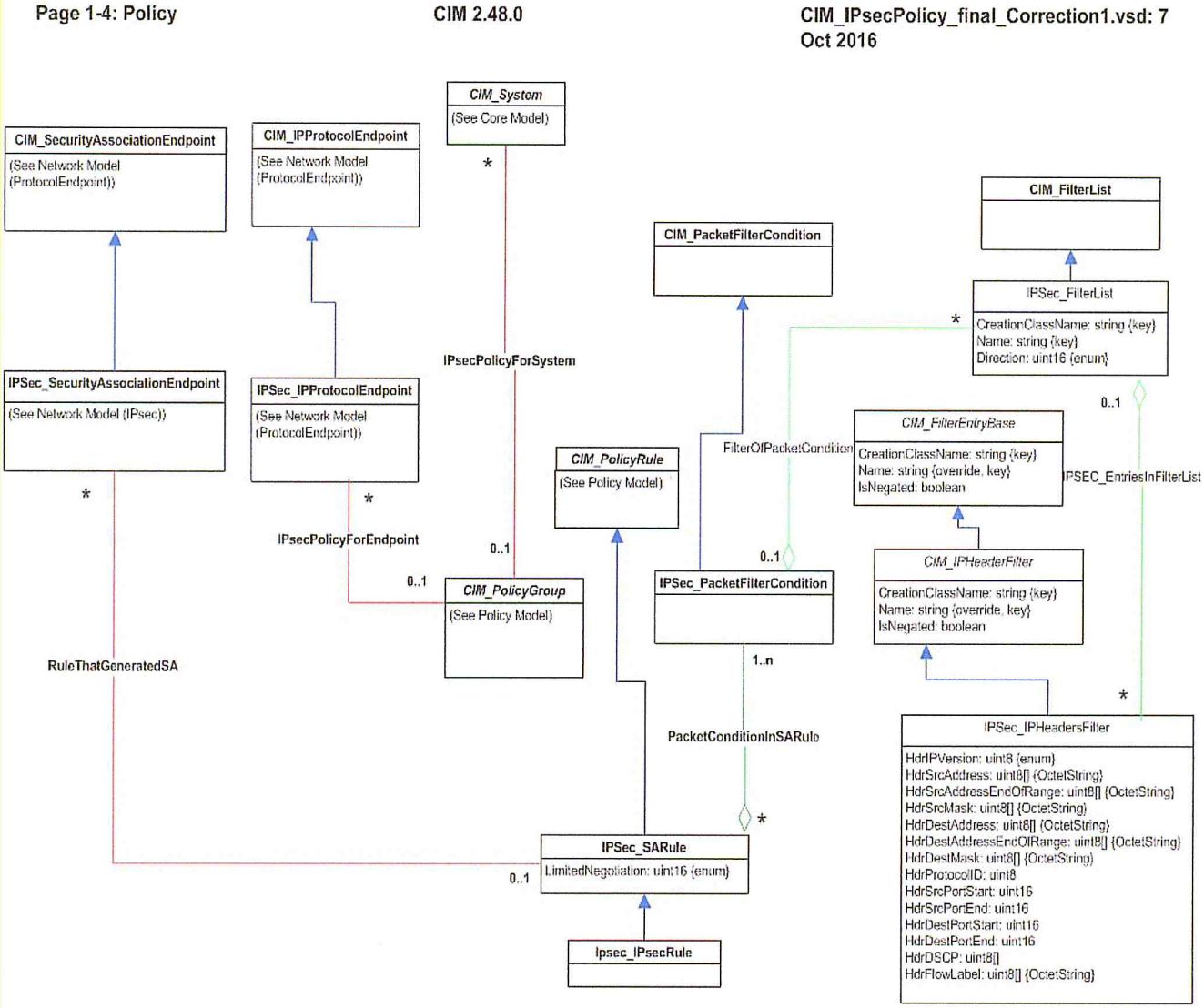

Figure 33 : *Diagramme de classes UML représentant les classes de politique (Policy Class)*

- CIM_ManagedElement est une classe abstraite qui représente la classe mère (ou la partie supérieure de l'arbre d'héritage) pour les classes non associées dans le schéma CIM.
- Les entités contrôlées par IPSec sont représentées par la classe abstraite IPSec_ManagedSystemElement

- La classe principale du Policy Class est la classe IPsec_SARule (hérite du CIM_SARule) qui représentent la classe de base pour IPsec_IPSecRule (qui hérite du CIM_IPSecRule).
- Le point de démarrage est la création d'une instance IPSec_PolicyRule qui est représentée par une instance de type (de la classe) IPSEC_PolicyGroup (hérite de CIM_PolicyGroup). Elle doit regrouper deux règles pour chaque direction de trafic (une politique pour le trafic entrant et une autre pour le trafic sortant). L'association IPSEC_PolicyRuleInPolicyGroup (hérite de CIM_PolicyRuleInPolicyGroup) est utilisée pour exprimer cette composition entre IPSEC_PolicyGroup et IPSec_IPsecRule.
- Une instance de CIM_PolicyGroup représente l'ensemble des stratégies utilisées sur une interface. Les instances de PolicyGroup sont définies et nommées par rapport au système CIM_System qui fournit leur contexte. Ce PolicyGroup doit être associé soit directement Avec l'instance de la classe IPProtocolEndpoint qui représente l'interface réseau (via l'association IPsecPolicyForEndpoint) ou indirectement (Via l'association IPsecPolicyForSystem) associée au système Qui héberge l'interface.
- Une politique IPSEC_PolicyGroup est donc composée de deux instances de la classe IPSec_IPsecRule qui hérite de CIM_IPSecRule (voir ci-après)

  Chaque instance de la classe IPSec_IPsecRule correspond à une direction du trafic (entrant ou sortant)
- Une instance de IPSec_IPsecRule permet de spécifier les actions statiques (telles que le rejet) elle est composée de deux instances des classes : IPSEC_PacketFilterCondition (hérite de CIM_PacketFilterCondition) qui représente la condition de la politique et de IPSec_PreconfiguredSAAction (hérite de IPSec_SAStaticCondition) qui représente l'action de la politique.
- L'instance de IPSEC_PacketFilterCondition (la condition de la politique) permet de spécifier le type de trafic qui doit passer par le tunnel IPSEC. Elle est reliée à l'instance IPSec_IPsecRule via l'association IPSEC_PolicyConditionInPolicyRule (hérite de CIM__PolicyConditionInPolicyRule)

  Il s'agit d'un ensemble de caractéristiques extraites généralement de l'entête IP du trafic:

- Adresses IP source
- Adresse(s) IP destination
- Numéro(s) de port(s) source
- Numéro(s) de port(s) destination
- Protocole particulier (IP, UDP, TCP, etc.)
- Type particulier de trafic retrouvé via le champ d'entête IP DSCP….. etc.

3.6.2. Les Classes d'actions (Policy Actions):

Les classes d'action sont utilisées pour modéliser les différentes actions que pourrait prendre un dispositif IPSec lorsque l'évaluation de la condition associée entraîne une correspondance.

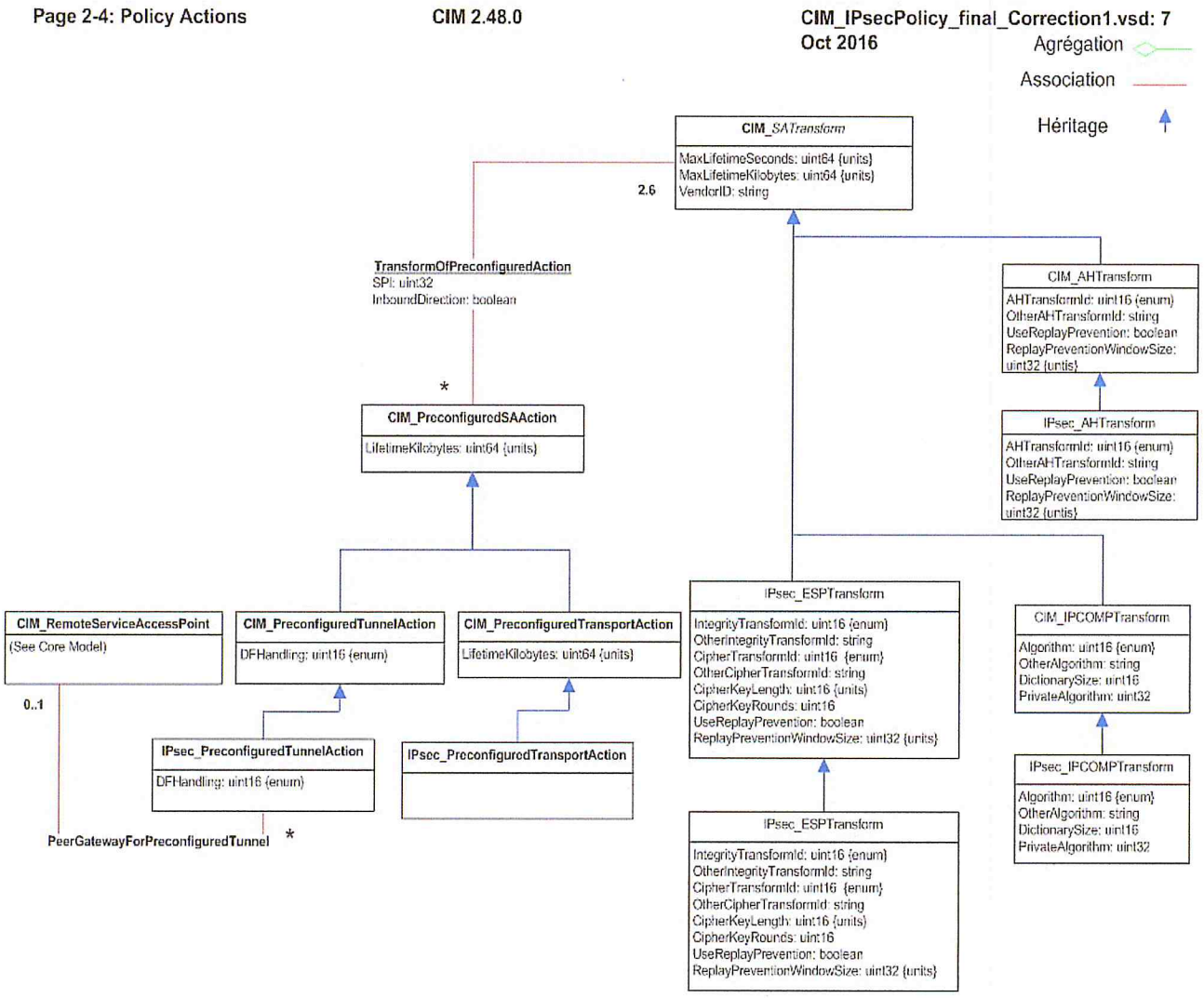

Figure 34 : *Diagramme de classes UML représentant les classes d'actions (Policy Actions)*

- L'action d'une politique IPSEC est représentée par la classe CIM_PreconfiguredSAAction. Cette dernière correspond au cas où la SA IPSEC est configurée manuellement et non pas négociée via le protocole IKE (dans ce cas une autre classe devrait être utilisée et qui sort de la portée de ce projet)
- La classe CIM_PreconfiguredSAAction est la classe ascendante de deux classes concrètes: IPSEC_PreconfiguredTransportAction (hérite de CIM_PreconfiguredTransportAction) et IPSec_PreconfiguredTunnelAction (hérite de CIM_PreconfiguredTunnelAction) correspondants respectivement à un tunnel IPSEC en mode Transport et un tunnel IPSEC en mode tunnel.
- Dans les deux cas, chaque instance IPSec_PreconfiguredSAAction est associée à une instance de la classe IPSec_SATransform via l'association TransformOfpreconfiguredAction cette association contient le paramètre SPI (Security Parameter Index) dans l'attribut SPI et spécifie si le trafic est de type entrant (InboundDirection = True) ou sortant (InboundDirection =False).
- La configuration des paramètres de cryptage est essentiellement contenue dans plusieurs instances de la classe IPSec_SharedSecret (hérite de CIM_SharedSecret) dont plusieurs instances doivent être créées. Cette classe contient les attributs suivants:
  - **Secret :** La clé secrète de session (la clé de cryptage)
  - **Protocol :** contient selon le cas de la valeur «ESP-encrypt», «ESP-auth» ou «AH»
  - **Algorithm :** contient l'algorithme utilisé pour protéger la clé secrète (Secret) qui peut être plaintext.
  - **Remoteid :** est la concaténation de l'adresse IP IPSEC du paire, suivi du caractère «/», suivi de IN ou OUT (selon la direction du trafic), suivi du caractère « / » et enfin de la valeur hexadécimal du SPI

3.7. Conclusion :

Tout au long de ce chapitre, nous avons présenté et expliquer l'approche de conception de notre système standardisé (car il est conforme au Policy Profile proposé par le DMTF) de gestion de politiques de sécurité. Nous avons présenté l'architecture de gestion que nous avons implémentée et qui s'inspire du protocole de gestion de politiques **COPS**. Concernant, la modélisation des politiques IPSec, nous avons proposé un modèle CIM permettant de définir des politiques IPSec pour les appliquer par la suite, avec les diagrammes de cas d'utilisation qui permet d'une meilleure compréhension de notre travail.

Dans le chapitre suivant nous présentons la dernière étape de ce projet qui consiste à implémenter ce modèle concrètement sur un système LinuxCentOS (7.0). Cette implémentation se traduit par l'installation d'une plateforme de gestion WBEM (client/serveur) ainsi que des outils nécessaires pour le développement des providers chargés de gérer les classes du modèle.

Chapitre IV

Réalisation du système

4.1 Introduction

Ce chapitre présente les différentes étapes d'implémentation de notre framework. Cette implémentation permet tout d'abord, de prouver la possibilité de gérer des politiques de sécurité abstraites en format CIM en les traduisant en politiques concrètes IPSec, ce qui rend la gestion d'IPSec dans un environnement hétérogène une activité simple, elle permet aussi aux administrateurs de systèmes hétérogènes d'avoir d'une vue globale, exhaustive et homogène de l'ensemble des politiques IPSec et par conséquent de l'ensemble des associations de sécurités de IPSec (les SAs) qui définissent les conditions pour l'établissement de sessions sécurisées ainsi que les règles et mécanismes de sécurités à appliquer sur ces sessions (ESP/AH, Transport/Tunnel etc.) Notre système devra permettre aussi d'offrir une plateforme pouvant être exploitée par un administrateur afin de composer, diffuser et appliquer, via un client WBEM, des politiques de sécurité sur des systèmes en réseau.

Ce chapitre retrace les phases de développement suivies que sont :

1. Les choix techniques adoptés qui ont permis la programmation et la mise en place de l'ensemble du système.
2. L'implémentation de notre Provider CIM/WBEM appelé IPSecPro, dédié à la gestion du IPSec et des politiques associées aux tunnels VPN.
3. Les détails de la de programmation d'un module particulier de notre Provider
4. Un plan de tests.

4.2 Les choix techniques

Pour l'implémentation de notre Framework, nous avons utilisé les technologies suivantes qui incluent : Le système Linux, le serveur CIM/WBEM qui permet la gestion des objets CIM et la communication avec d'un côté les ressources gérées et d'un autre coté les clients CIM, un ensemble de providers CIM couvrant un grand nombre de domaines de gestion et enfin le framework open source dédié au développement du provider CIM de notre projet:

4.2.1 Le système d'exploitation Linux CentOS 7 :

Pour l'implémentation de notre système, nous avons choisi comme plateforme la distribution Linux CentOS 7.0, qui est principalement destinée aux serveurs. CENTOS est un clone au système Linux commercial RedHat et en même temps très similaire au système communautaire Linux Fedora.

L'intérêt de CENTOS dans notre projet est qu'il intègre dans ses dépôts officiels de logiciels supportés, la plupart des package mettant en œuvre l'ensemble de l'écosystème CIM/WBEM à savoir : le serveurs CIM/WBEM (OpenPegasus et SFCB), plusieurs clients WBEM (notamment CIMCLi et WBEM-CLI) Un grand nombre de Providers CIM/WBEM couvrant un bon nombre de domaines de gestion standardisés par le DMTF, issues essentiellement des projets SBLIM (supporté par IBM) et OPENLMI(supporté par REDHAT) enfin des librairies facilitant l'exploitation des objets CIM et le développement aussi bien de Providers que d'applications clientes CIM/WBEM (notamment les projets PYWBEM, KONKRETE-CMPI et OPENLMI) [19]

4.2.2 Le projet OpenPegasus :

Le projet OpenPegasus développé et maintenu par l'OpenGroup [20], est une implémentation open-source des standards CIM et WBEM du DMTF, plus précisément il implémente le serveur CIM Object Manager (CIMOM) et le CIM Object Repository (CIMOR) et il offre un client CIM et une API en langage C++ dédiée au développement de Providers et d'applications CIM. Il est conçu pour être portable (multi-OS) et modulaire. Il est codé en C++ afin qu'il traduise efficacement les concepts des objets CIM dans un modèle de programmation, mais tout en conservant la performance et l'efficacité d'un langage compilé. OpenPegasus implémente l'ensemble des composants de l'architecture WBEM qui sont appariés aux différents points de gestion de politiques de notre architecture (PAP, PDP, PRP) à l'exception des Providers CIM qui représentent les PEP. Nous avons choisi ce projet pour sa stabilité, sa maturité et pour l'exhaustivité de ses fonctionnalités. [21]

4.2.3 Le framework CIMPLE :

Pour le développement de notre provider CIM, nous avons choisi le Framework de développement CIMPLE dédié à la programmation de providers et de clients CIM. Cet outil facilite énormément l'implémentation par la génération de squelettes de programmes en C++ à partir de la définition en MOF des classes CIM relatives aux ressources cibles. La puissance de ce Framework réside dans son approche qui consiste à générer des classes C++ qui correspondent (et qui sont identiques) aux classes MOF avec une puissante gestion des types de données et la génération des squelettes des méthodes intrinsèques et extrinsèques. Le travail du développeur consiste à coder les méthodes en fonction de la (des) ressource(s) visée(s) par le Provider ou le client [22]

4.2.4 L'outil cimcli :

Est un outil de test à la ligne de commande pour l'exécution des opérations d'un client CIM. Il implémente toutes les opérations CIM de DMTF à l'exception la modification et de la création des opérations de classe / instance et comprend plusieurs autres opérations qui sont utiles pour tester les serveurs CIM et pour tester notre Provider (le responsable de la gestion des politiques IPSec par l'invocation des méthodes des classes CIM correspondante). [23]

4.2.5 L'outil ip xfrm :

«ip xfrm» est un outil offert par le package iproute2 qui offre une interface de gestion complète de la pile TCP/IP sous Linux. Cette commande permet principalement la gestion des tunnels IPSec, voici les commande utilisé pour xfrm : [24]

ip xfrm state add Ajouter une nouvelle politique dans la base des associations de sécurité SAD, une politique pour la SAD comprend les adresses source et destination, le protocole, le SPI, le mode, l'algorithme d'authentification et l'algorithme de cryptage.

ip xfrm state flush Supprimer une politique de la SAD.

ip xfrm policy add Ajouter une nouvelle politique dans la base des politiques de sécurité SPD, une politique pour la SPD comprend les adresses virtuelles (locale et distante) et les adresses réelles (source et destination) pour chaque extrémité, avec la direction, le protocole et le mode utilisé.

ip xfrm policy flush Suppri3mer une politique de la SPD.

Par la suite il faut définir l'adresse destination virtuelle et la route pour chaque extrémité avec les deux commandes :

ip addr add Ajouter une adresse destination virtuelle.

ip route add Définir les adresses source et destination (virtuelle) du tunnel.

4.3 Implémentation du Providers

Avant de présenter cette partie du travail, rappelons qu'un Provider CIM est un composant intégré au serveur CIMOM qui permet d'interagir avec la(les) ressource(s) gérée(s) afin de traduire les requêtes exprimées en langage CIM en commandes de bas niveau exploitables par

l'entité ciblée et inversement, traduire les retours et les informations produites par ces entités en objets conformes au standard CIM.

Un provider implémente donc un ensemble de classes et associations CIM qui forment un modèle ou profil CIM de préférences parmi ceux standardisés par le DMTF (comme celui choisi pour la réalisation de notre projet)

La principale fonction de notre Provider est d'établir (et de supprimer) des tunnels VPN IPSEC, en prenant comme argument les instances CIM qui permettent de retrouver tous les paramètres d'une politique IPSEC ainsi que l'association de sécurité associée (SA)

Notre Provider ne prend pas en compte le service IKE qui permet la négociation entre les extrémités d'un tunnel IPSEC afin de choisir les algorithmes et les différentes clés de signature numérique et de cryptage ainsi que les différents identifiants (comme le SPI). Ces paramètres spécifiés manuellement par l'administrateur.

Une politique IPSEC est modélisée par un diagramme d'instances conforme au diagramme de classes détaillée dans le chapitre Conception (voir figure 41)

4.3.1 Diagramme d'instance :

En utilisant les deux diagrammes d'actions et de politiques définies dans le chapitre conception en arrive a défini le diagramme de classe suivant qui va nous permettre de la création d'un diagramme d'instance.

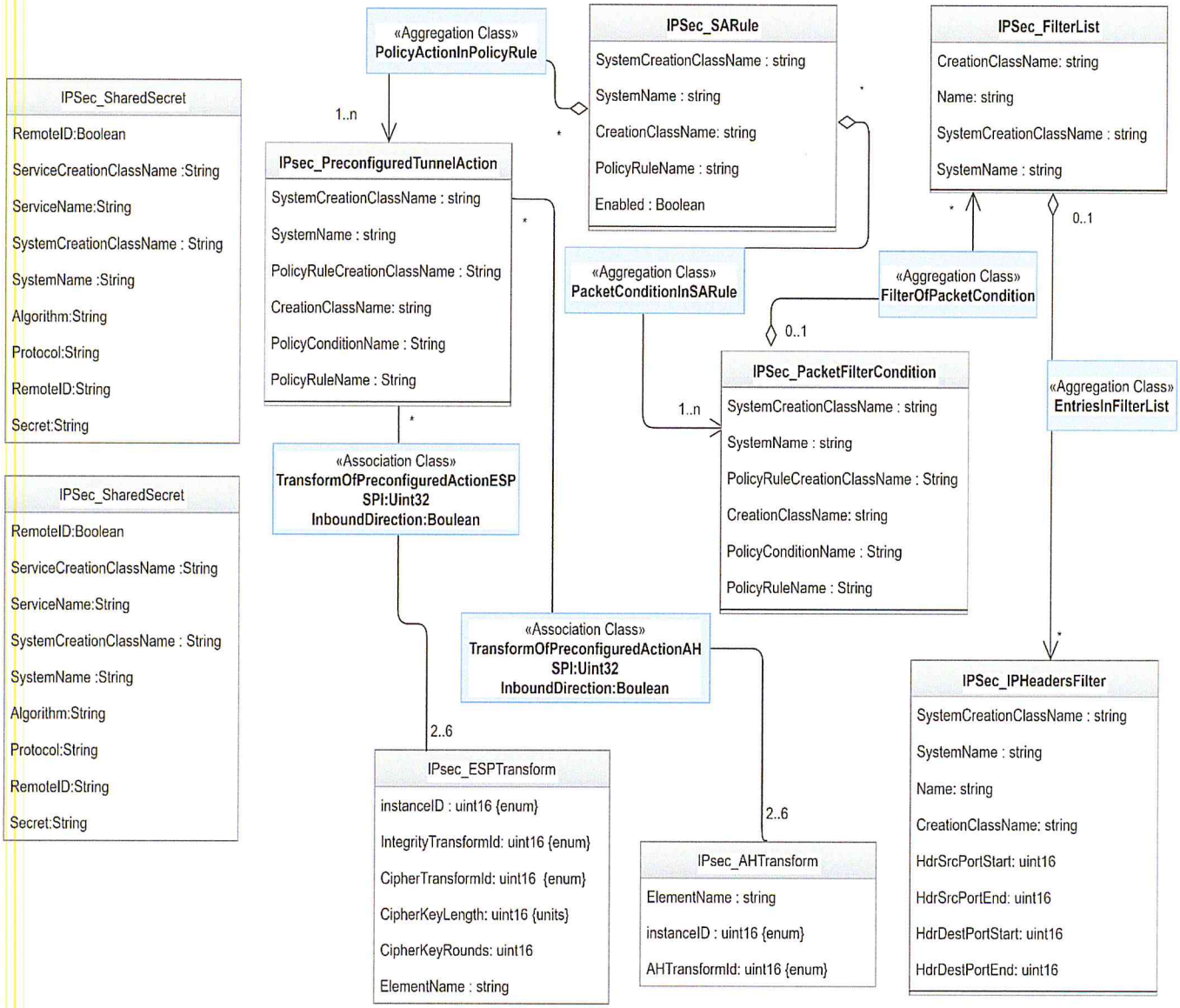

Figure 35 : *Diagramme de classes représentant les politiques/SA IPSEC*

La figure 41 présente un exemple de diagramme d'objet exprimé en langage UML qui représente une instance du diagramme de classes et qui illustre l'état de notre système à un moment donné.

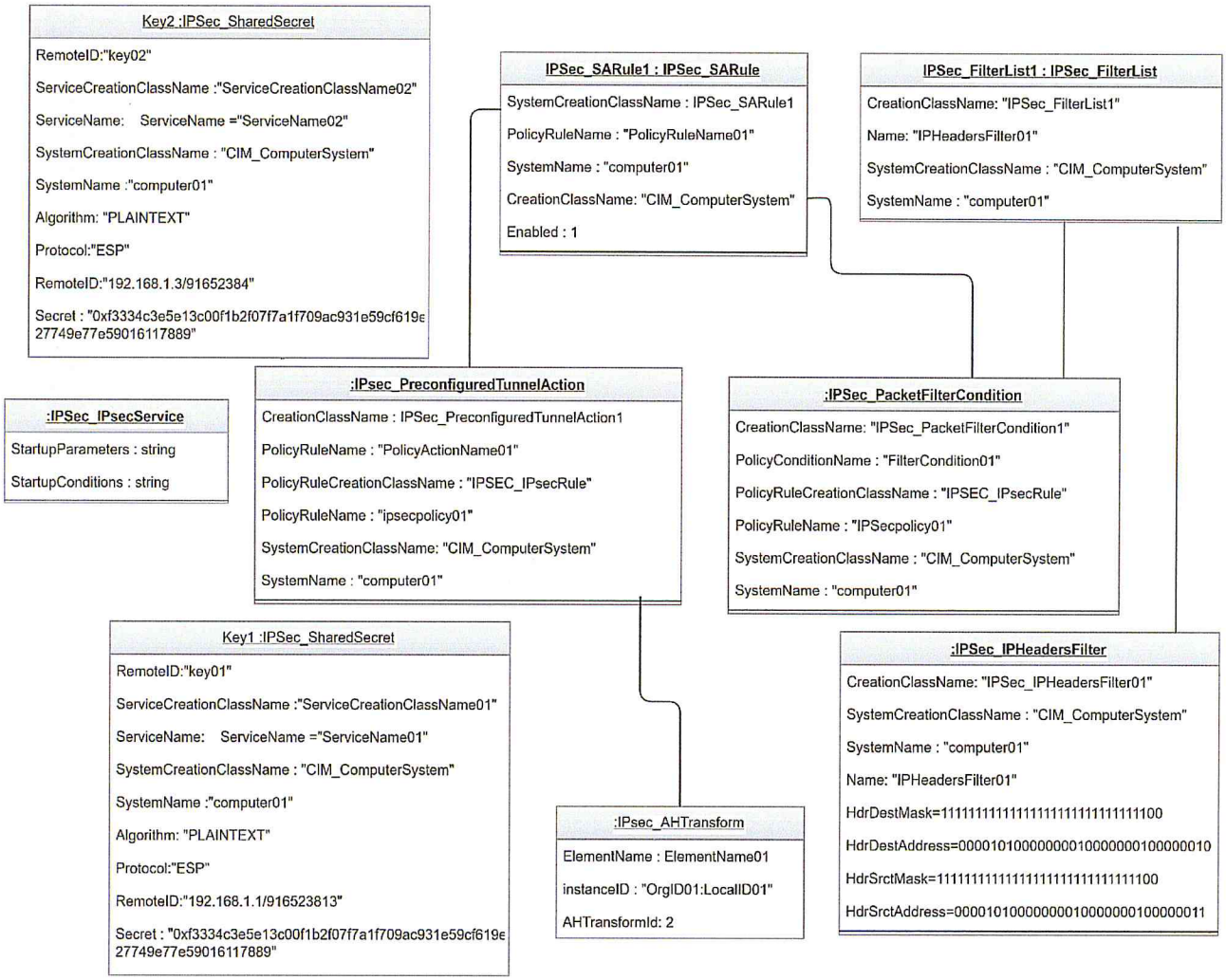

Figure 36 : *Exemple de diagramme d'objet (instances) modélisant une politique/SA IPSEC*

Le point d'entrée au modèle est l'instance de la classe IPSec_IPsecRule qui hérite de la classe principale IPSec_SARule. IPSec_IPsecRule permet à travers les différents instances des associations utilisées dans le modèle des classes, de traverser toutes les classes et récupérer les instances des classes modélisant les différents aspects d'une politique/SA IPSEC :

- La condition de la politique qui permet de «déclencher» l'aiguillage du trafic ciblé vers le tunnel IPSEC
- La condition est associée au filtre qui permet de sélectionner le trafic ciblé
- L'action de la politique qui détermine les transformations que va subir le trafic devant passer par le tunnel

- Les classes modélisant les différents paramètres de signature numérique et cryptage de données
- etc.

Un tel diagramme d'instances devrait être traduit en format textuel conforme au langage MOF (Managed Object Format) pour qu'il devienne exploitable par un serveur CIM-OM, mais également par les différents programmes.

Chaque instance du diagramme (qu'il s'agisse d'une classe ou d'une association) doit être exprimée en format MOF puis compilée et intégrée dans le serveur CIM-OM pour qu'elle puisse être stockée dans la base des objets CIM (qui représente le PRP dans notre architecture)

4.3.2 Création des instances

Nous utilisons un client CIM (CIMCLI, WBEM-CLI ou CIMMOF) afin de créer les instances de notre politique CIM-IPSEC.

Rappelons que les instances utilisées dans notre modèle peuvent être des instances d'une classe simple, une classe d'agrégation ou une classe d'association.

4.3.2.1 Instance d'une classe simple

Exemple de création d'une instance de la classe IPSec_ESPTransform :

```
instance of IPSec_ESPTransform

{

InstanceID = "OrgID01:LocalID01";
ElementName = "ElementName01";
.
.
CipherKeyLength            = 101;
  CipherKeyRounds          = 5;
  CipherTransformId        = 6;
  IntegrityTransformId     = 3;

};
```

Figure 37 : *Capture d'écran pour une instance de la classe IPSec_ESPtransform.*

La classe est composée de deux catégories d'attributs, la première catégorie inclue les attributs clés de cette classe qui sont InstanceID (héritée de la classe CIM_SettingData) et ElementName (héritée de la classe CIM_SATransform) permettant d'identifier de manière unique cette instance, la deuxième catégorie regroupe les autres attributs qui renseignent d'autres paramètres. La syntaxe générale pour créer l'instance d'une classe simple est le suivant :

Instance of IPSec_ClasseName

{

Atributs Clé ;

Atribut de la classe courant;

};

4.3.2.2 Instance d'une classe d'agrégation :

Exemple d'une instance pour la classe d'agrégation IPSec_PolicyActionInPolicyRule :

```
instance of IPSec_PolicyActionInPolicyRule

{

GroupComponent=IPSec_SARule.CreationClassName="IPSec_SARule",PolicyRule
Name="PolicyRuleName01",SystemCreationClassName="CIM_ComputerSystem",Sy
stemName="computer01";
.
.
PartComponent=IPSec_PreconfiguredTunnelAction.CreationClassName="IPSec_
PreconfiguredTunnelAction",PolicyActionName="PolicyActionName01",Policy
RuleCreationClassName="IPSEC_IPsecRule",PolicyRuleName="ipsecpolicy01",
SystemCreationClassName="CIM_ComputerSystem",SystemName="computer01";

};
```

Figure 38 : *Capture d'écran pour une instance de la classe d'agrégation IPSec_PolicyActionInPolicyRule*

Pour définir l'instance d'une classe d'agrégation :

1. Il faut identifier les deux classes qui sont reliées avec cette agrégation, l'attribut GroupComponent représente une référence vers la classe englobante de la relation et l'attribut PartComponent représente une référence vers la classe contenue dans l'agrégation.
2. Identifier les attributs clés pour les références PartComponent et GroupComponent en respectant la syntaxe illustrée dans la figure ci-haut (figure 43).

4.3.2.3 Instance d'une Classe d'association :

Exemple d'une instance pour la classe d'association IPSec_TransformOfPreconfiguredActionESP :

```
instance of IPSec_TransformOfPreconfiguredActionESP

{

    InboundDirection = "true";
    SPI        = 916523813;
    .
    .
    Antecedent=IPSec_ESPTransform.InstanceID="OrgID01:LocalID01",ElementName="Element
    Name01",CipherKeyLength="101",CipherKeyRounds="5",CipherTransformId ="6",Integrit
    yTransformId="3";

    Dependent=IPSec_PreconfiguredTunnelAction.CreationClassName="IPSec_PreconfiguredT
    unnelAction",PolicyActionName="PolicyActionName01",PolicyRuleCreationClassName="I
    PSEC_IPsecRule",PolicyRuleName="ipsecpolicy01",SystemCreationClassName="CIM_Compu
    terSystem",SystemName="computer01"

};
```

Figure 39 : *Capture d'écran pour une instance de la classe d'association IPSec_ TransformOfPreconfiguredActionESP*

Pour définir l'instance d'une classe d'association, on suit les mêmes étapes que celles suivies dans le cas précédent sauf que qu'on va plutôt spécifier les attributs Dependent et Antecedent en plus des autres attributs de la classe d'association s'il y en a.

L'ensemble de la politique IPSEC exprimé en format CIM devrait être compilé et intégré à la base des objets CIM du serveur pegasus en créant toutes les instances des classes, des agrégations et autres associations impliquées dans la politique.

Une fois la politique est déclarée, elle peut être mise en œuvre par notre Provider CIM dédié à la classe IPSec_IPsecService. Cette classe expose deux méthodes importantes : IpsecPolicyActivate() et IpsecPolicyDeactivate().

La première méthode, prend en argument la référence de l'instance IPSec_IPsecRule relative à la politique IPSec qu'on voudrait mettre en œuvre. Cette référence va permettre au provider de retrouver toutes les autres instances de la politique et pouvoir construire la configuration IPSec compréhensible par le système IPSec sous-jacent. Une fois la configuration construite, le provider procède à la création du tunnel IPSec au niveau de l'extrémité dans laquelle il est installé et ce par la création et l'installation de règles de politique et d'association IPSec.

Le client CIM devrait bien entendu exécuter cette méthode au niveau des deux extrémités pour que le tunnel puisse s'établir entre elles.

La seconde méthode permet de désactiver une politique identifiée par une référence vers l'instance IPSec_IPsecRule. Le travail inverse va être réalisé afin de supprimer le tunnel et toute configuration y afférant.

4.3.3 Développement du provider CIM/WBEM IPSecPro

Afin de développer notre provider CIM IPSecPro, nous avons utilisé un framework dédié à ce genre de développement : CIMPLE. CIMPLE est un outil qui permet de simplifier le développement de Providers CIM en langage C++. La puissance de CIMPLE est sa capacité à traduire des déclarations de classes en format MOF en classes équivalente en langage C++ et en introduisant de nouveaux types C++ correspondant aux types de données MOF.

De plus, CIMPLE offre des outils qui facilitent la génération du fichier Makefile qui permet la compilation appropriée et la génération de modules prêt à l'utilisation, ciblant trois interfaces de gestion CIM: Pegasus, CMPI et Microsoft WMI.

Enfin CIMPLE offre la possibilité d'intégrer facilement les modules (Providers CIM) générés au serveur CIM-OM ciblé.

Le développement de notre Provider IPSECPRO en utilisant le framework CIMPLE passe par les étapes suivantes :

1. **Rédaction des fichiers MOF correspondant à l'ensemble du modèle :**

Cette étape consiste en la traduction des classes avec un préfix « IPSec_» des modèles conçues dans le chapitre de la conception (sous la forme de diagrammes de classes UML) vers des fichiers nommés IPSec_Nom_classe.mof qui sont écrits en langage MOF, de la manière suivante :

```
    [UMLPackagePath ( "CIM::Network::IPsec" ),
    Description (
        "IPSec_AHTransform inherited from CIM_AHTransform "
       "AHTransform defines the parameters used for a phase 2 AH "
       "(Authentication Header) Security Association." ),
    MappingStrings { "IPSP Model.IETF|AHTransform" }]
    class IPSec_AHTransform : CIM_AHTransform {
      [Description (
          "AHTransformId is an enumeration that specifies the hash "
          "algorithm to be used. The list of values was generated "
          "from RFC2407, Section 4.4.3." ),
       ValueMap { "1", "2", "3", "4" },
       Values { "Other", "MD5", "SHA-1", "DES" },
       MappingStrings { "IPSP Model.IETF|AHTransform.AHTransformID",
          "RFC2407.IETF|Section 4.4.3" },
       ModelCorrespondence { "CIM_AHTransform.OtherAHTransformId" }]
      .
      .
      .
      [Description (
          "ReplayPreventionWindowsSize specifies, in bits, the "
          "length of the sliding window used by the replay "
          "prevention mechanism. The value of this property is "
          "meaningless if UseReplayPrevention is false. The window "
          "size MUST be a power of 2." ),
       Units ( "Bits" ),
       MappingStrings {
          "IPSP Model.IETF|AHTransform.ReplayPreventionWindowSize" }]
      uint32 ReplayPreventionWindowSize;
};
```

Figure 40 : *Capture d'écran pour la classe IPSec_AHTransform.mof*

Par la suite l'ensemble des fichiers .mof seront incluse dans un fichier nommé repository.mof en utilisant la directive #pragma include (" /Directory/IPSec_Nom_classe.mof") on respectant l'ordre de l'héritage de la manière suivante :

```
//Classes IPSec policy classes
#pragma include("IPSecclasses/IPSecpolicyclasses/IPSec_FilterList.mof")
...
//Association
#pragma include("IPSecclasses/IPSecpolicyclasses/IPSec_EntriesInFilterList.mof")
...
//End Classes IPSec Policy classes
//Classes IPSec Action classes
#pragma include("IPSecclasses/IPSecActionclasses/IPSec_PreconfiguredTransportAction.mof")
...
//Association
#pragma include("IPSecclasses/IPSecActionclasses/IPSec_PeerGatewayForPreconfiguredTunnel.mof")
...
//End Classes IPSec Action classes
//IPSecService
#pragma include("IPSecclasses/IPSec_IPSecService.mof")
...
//end IPSecService
```

Figure 41 : *Capture pour une partie le fichier repository.mof*

2. Génération des fichiers .cpp :

Dans cette étape, nous utilisons CIMPLE qui nous permet la traduction des fichiers (.mof) en fichiers C++ (.cpp) et (.h). En utilisant la commande genclass fournie par CIMPLE.

Les résultats de cette commande donnent une définition des classes de langage C++ qui utilisent des structures de données standards définies par CIM qui permet de faciliter le travail des programmeurs, voici un exemple pour la commande **genclass** :

```
genclass -r IPSec_IPSecService IPSec_FilterList ....
```

3. Génération du squelette du Provider :

CIMPLE génère des squelettes du provider sous forme de fichiers (.h) et de code C++ (.cpp) automatiquement à partir des classes MOF.

La commande suivante par exemple, génère des squelettes du provider pour la classe IPSec_IPsecService, Ce squelette comprend les définitions ainsi que les corps des méthodes CIM du provider.

```
genprov IPSec_IPsecService
```

La séparation des fichiers de définition des classes CIM de la deuxième phase et les fichiers des providers de cette phase va permettre de rendre indépendant la définition des classes CIM des opérations sur celles-ci. Ce qui permet au programmeur d'effectuer des changements dans son modèle MOF et régénérer les classes C++ correspondants avec la commande genclass, comme il pourrait opérer des modifications sur le code des opérations CIM sans toucher au modèle associé. Pour générer les squelettes des providers, nous exécutons la commande genprov du Framework CIMPLE.

4. L'ajout des codes des méthodes :

Après la génération des squelettes des providers, nous pouvons maintenant procéder à l'implémentation des méthodes définies de la classe IPSec_IPsecService.

5. Génération du module :

A cette étape, la commande **genmod** est utilisée afin de permettre la génération du fichier module.cpp contenant le code et les informations permettant l'enregistrement du provider auprès du serveur CIMOM.

6. Compilation du Provider :

Avant de procéder à la compilation de nos programmes, nous devons générer le fichier Makefile avec la commande **genmak** du Framework CIMPLE. Ce fichier décrit l'ordre de la compilation des différents fichiers sources ainsi que d'autres informations nécessaires à cette

opération comme les chemins des bibliothèques d'OpenPegasus de CIMPLE (libcimple, libcommon et libpegadaptater).

Nous procédons ensuite à la compilation de notre projet en lançant la commande **make**. Cette opération va générer des fichiers objet intermédiaires (ayant l'extension « .o») qui seront compilés à leur tour pour générer notre bibliothèque partagée IPSecPro.so, le résultat de notre travail.

7. Enregistrement du Provider :

L'enregistrement du provider constitue la dernière étape. Cette étape permet d'instruire le serveur CIMOM d'associer toutes les requêtes sur les classes que nous avons définies dans le chapitre conception aux programmes implémentés par notre module construit grâce au Framework CIMPLE par la commande **regmod**.

Le Schéma suivant explique les étapes principales pour la réalisation de notre Provider.

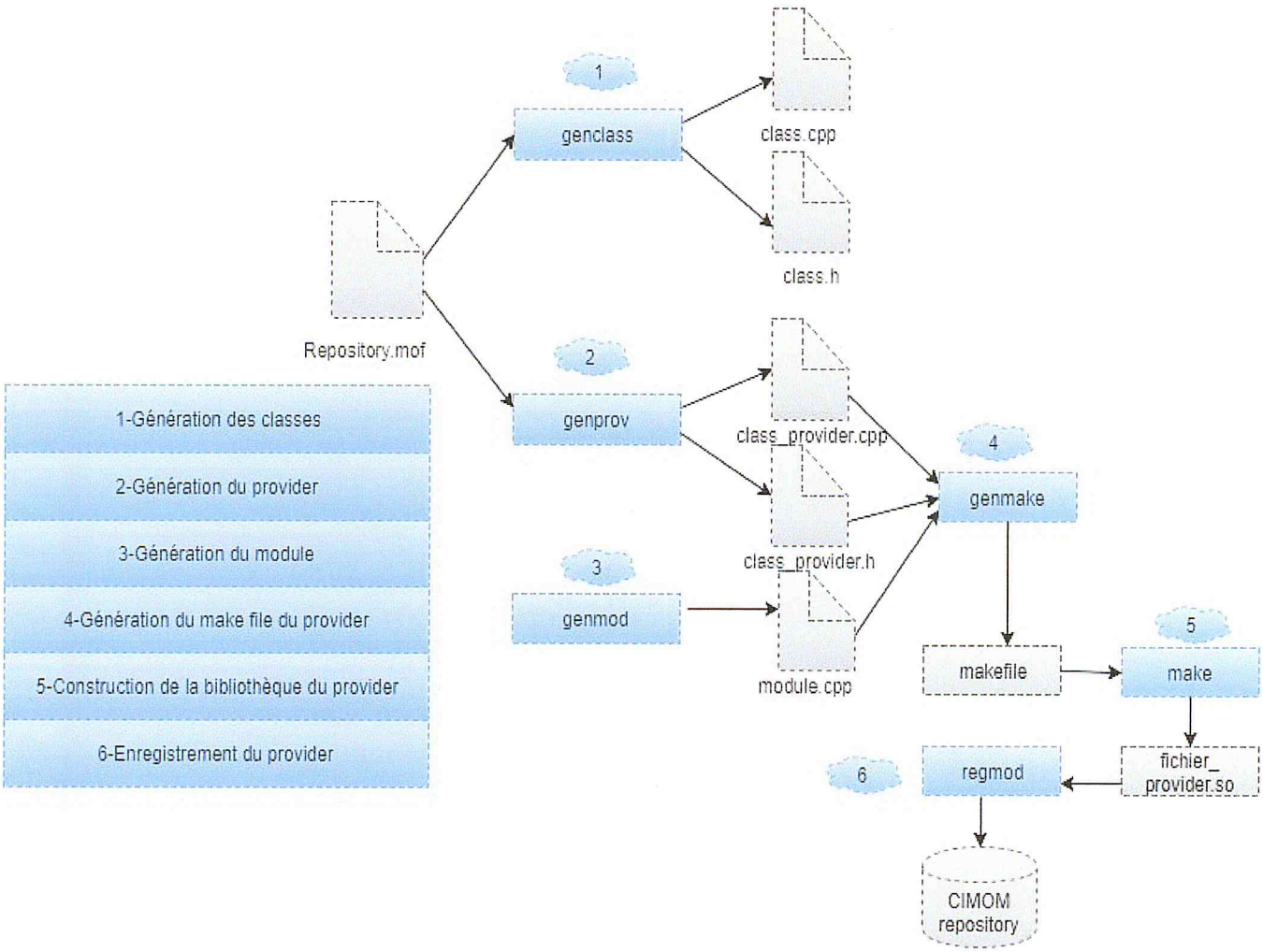

Figure 42 : Etapes de programmation du Provider

4.4. Codage des méthodes IPsecPolicyActivate() et IPsecPolicyDeactivate():

Après la génération des squelettes des providers, nous pouvons maintenant procéder à l'implémentation des méthodes définies pour notre class IPSec_IPsecService.

La méthode IPsecPolicyActivate() permet d'activer une politique IPSec, cela passe par la récupération des instances qui représente les paramètres de configuration de notre tunnel IPSec et les paramètres nécessaires de la commande ip xfrm qui permet d'agir directement sur les bases de données des SA (SAD) et de politiques (SPD) (on invoque dans nos programmes des appels de commande shell ip xfrm avec les arguments qu'il faut pour chaque situation), en parcourant toutes les instances participant à construire la politique.

Le parcours donc permet de récupérer :

1. Les adresses sources et destination à partir de la classe **IPSec_IPHeadersFilter.**
2. Le mode utilisé pour créer les SAs en utilisnt de la classe **IPSec_PreconfiguredTunnelAction** via l'agrégation IPSec_PolicyActionInPolicyRule pour le mode tunnel qui est le mode qu'on va utiliser pour notre test (pour le mode Transport il faut utiliser la classe IPSec_PreconfiguredTransportAction).
3. Le SPI (Security Policy Index) et la direction du trafic sont récupérés de l'association **IPSec_TransformOfPreconfiguredActionAH.**
4. Le protocole utilisé (AH ou ESP) est récupéré de la classe **IPSec_SharedSecret**.
5. les clés partagés est les algorithmes de cryptage et de hachage utilisés pour le chiffrement sont récupérés de la classe **IPSec_SharedSecret**.

Après avoir récupéré ces paramètres, il est possible de mettre en place le tunnel IPSEC en utilisant la commande ip xfrm.

Notons que notre provider doit lui manipuler les objets CIM qui constituent la politique IPSEC (décrites plus haut) et pour faire, il a besoin d'une interface CIM cliente. Le projet CIMPLE offre parmi ces fonctionnalités une telle interface appelé BREVETY.

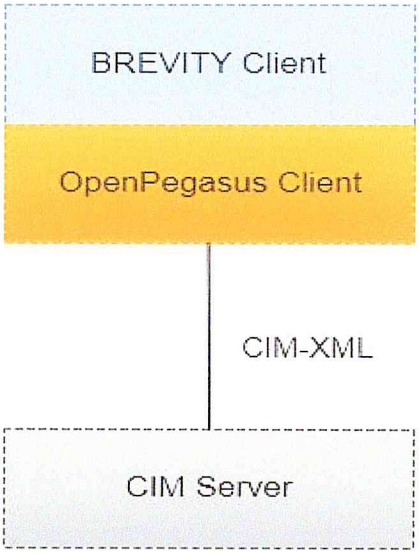

Figure 43 : L'API BREVITY qui permet la création des clients CIM

Un certain nombre d'étapes doivent être suivi afin de pouvoir interagir avec des objets CIM. Il s'agit essentiellement de générer des fichiers en C++ correspondants aux classes MOF ciblés par le client CIM.

En plus de la commande genclass présentée plus haut, BREVITY offre la command genhnd qui permet de générer des classes C++ permettant de manipuler les instances CIM existantes (contrairement au provider qui permet de produire des instances CIM).

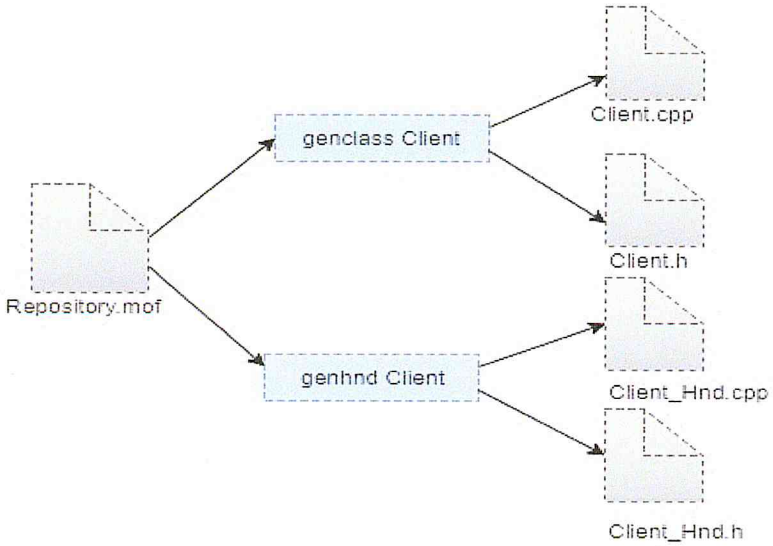

Figure 44 : *Génération des classes concrète*

4.5. Tests du Provider IPSecPro :

Dans cette section, nous présentons un plan de tests qui couvre les fonctionnalités clés de notre provider à suivre : la création d'une politique IPSec et l'activation /désactivation de cette politique, (l'activation d'une règle IPSec provoque la création de notre tunnel entre deux extrémités).

Au niveau de la première extrémité on effectue les étapes de 1 à 4 suivantes :

Etape 1: Edition et intégration d'une politique IPSEC en format CIM/MOF

Cette étape consiste en la définition des paramètres cités précédemment qui permettent de la configuration d'un tunnel IPSec, ces paramètres sont définis dans un fichier (instances_ipsec.mof) qui contient des instances pour tous les classes nécessaires.

Pour stocker ces instances au niveau de CIMOR il faut exécuter la commande **cimmof** qui prend en paramètre un fichier en format mof contenant l'ensemble des instances CIM avec le bon format.

cimmof instances_ipsec.mof

On prend comme exemple la politique IPSec avec les paramètres de configuration suivant :

Exemple de paramètres des deux SAs (destinés au SAD):

Adresse source : 192.168.1.10.

Adresse destination : 192.168.1.15.

Protocole : AH.

SPI : 0x53fa0fdd.

Mode : tunnel.

Algorithme de hachage: sha256

Algorithme de cryptage: AES

$KEY1: 0xf3334c3e5e13c00f1b2f07f7a1f709ac931e59cf619e27749e77e59016117889

$KEY2: 0x9f71281789684cb6fde2ebe6df01739d49330eb4f9fbe76ddc2cc09b63ad3923

Paramètres des deux politiques IPSEC (destinés au SPD):

Direction: IN.

Adresse locale (virtuelle) : 10.10.10.1

Adresse Remote (virtuelle) : 10.10.10.2

Avec les adresses sources et destination, le reqid qui prend la même valeur du SPI, le protocole et le mode qui sont avec les mêmes valeurs comme celle définie pour la SAD.

Etape 2 : Tester l'existence des instances qui constituent une politique IPSEC cible

Les tests présentés ici ont été réalisés à l'aide de l'interface ligne de commande du système Linux, nous avons donc veillé à décortiquer les sorties afin d'interpréter les résultats.

Pour tester que les instances sont bien créés en exécute la commande **cimcli ei** « IPSec_Nom de la classe » qui permet de retourner l'instance de la classe ciblé.

```
[root@localhost IPSecPro]# cimcli ei IPSec_SARule

// path= IPSec_SARule.CreationClassName="IPSec_SARule",PolicyRuleName="PolicyRuleName01"
,SystemCreationClassName="CIM_ComputerSystem",SystemName="computer01"

instance of IPSec_SARule
{
    InstanceID = NULL;
    Caption = NULL;
    Description = NULL;
    ElementName = NULL;
    Generation = NULL;
    CommonName = NULL;
    PolicyKeywords = NULL;
    PolicyDecisionStrategy = NULL;
    PolicyRoles = NULL;
    Enabled = 1;
    SystemCreationClassName = "CIM_ComputerSystem";
    SystemName = "computer01";
    CreationClassName = "IPSec_SARule";
    PolicyRuleName = "PolicyRuleName01";
    ConditionListType = 1;
    RuleUsage = NULL;
    Priority = 0;
    Mandatory = NULL;
    SequencedActions = 3;
    ExecutionStrategy = NULL;
    LimitNegotiation = NULL;
};
```

Figure 45 : *Capture d'écran pour l'instance de la classe IPSec_SARule*

Etape 3: Récupérer l'instance de la classe IPSec_IPsecService

On peut récupérer l'instance de la classe IPsec_IPsecService avec la commande cimcli ei.

cimcli ei IPsec_IPsecService

Le résultat de cette commande permet de récupérer tous les informations nécessaires pour la configuration du tunnel.

Etape 4 : Invoquer la méthode IPsecPolicyActivate() en lui fournissant la référence de l'instance de la classe IPSec_IPsecRule

À partir du résultat de la commande cimcli ei, on invoque la méthode IPsecPolicyActivate en lui donnant comme argument l'identifiant de l'instance IPsec_IPsecRule (PolicyRuleName) qui permet de parcourir toutes les classes présentées dans notre diagramme d'instances est donc

récupérer les paramètres de configuration pour une politique spécifiée par l'identifiant de IPSec_IPsecRule.

La commande ciminvoke est exécutée de la manière suivante :

ciminvoke IPsec_IPsecService.Name=...,SystemName=...,CreationClassName=IPSec_IPsecService,SystemCreationClassName=CIM_ComputerSystem IPsecPolicyActivate param=value IPSec_IPsecRule cible.

Etape 5 : On refait la même séquence précédente au niveau de la deuxième extrémité IPSec (la deuxième machine) en inversant les paramètres de la source et de la destination.

Etape 6 : Au niveau de chaque machine, on doit vérifier que les politiques ont été correctement configurées et installées avec les commandes ip xfrm.

Ils s'agit de verifier que des SAs et des politiques IPSEC sont bien installés et avec les paramètres choisis.

Les commandes qui permettent de vérifier la SAD :
ip xfrm state show
ip xfrm state get src addr **dst** addr **dir** direction
Les commandes qui permettent de vérifier la SPD :
ip xfrm policy get src addr **dst** addr **dir** direction
ip xfrm policy show

Etape 7 : Envoyer du trafic et vérifier qu'il est routé à travers le tunnel IPSEC

On peut vérifier que le tunnel est bien routé avec :

1. Un ping vers l'adresse virtuelle de l'autre extrémité.
2. Utiliser la commande tcpdump pour vérifier que le trafic est encapsulé dans des paquets IPSec.
3. Utiliser la commande ip xfrm monitoring pour voir quelles sont les politiques mises en place.

4.5 Conclusion :

En résumé, pour implémenter notre système standardisé de gestion de politiques de sécurité, nous avons installé le serveur OpenPegasus ainsi que le client wbemcli dans une plateforme Linux Centos. Ensuite, nous avons installé tous les outils nécessaires pour le développement du Provider qui représente le noyau de notre travail d'implémentation. Nous avons détaillé les étapes qui permettent le développement de notre Provider IPSecPro en utilisant le framework CIMPLE ainsi que les étapes qui permettent la création et la récupération des instances, ces derniers permet d'établir un tunnel IPSec on utilisant l'outil ip xfrm qui permet d'agir directement sur les bases de données des SA (SAD) et de politiques (SPD), On a finalisé notre travail avec un test pour vérifier que le tunnel est bien établit.

Nous rappelons que le résultat attendu de ce travail est une bibliothèque (.so) qui représente le Provider permettant d'opérationnaliser notre modèle. Cette librairie peut être enregistrée dans n'importe quel serveur WBEM pour pouvoir gérer des politiques IPSec abstrait en format CIM.

Conclusion générale et perspectives :

Notre travail présente la conception et réalisation d'un outil normalisé dédié à la gestion des politiques de sécurité, qui permet la gestion de politiques IPSec en utilisant le standard CIM/WBEM. La conception de ce système montre la possibilité de modéliser, composer, appliquer ou retirer des politiques IPSec en format CIM et à travers un environnement de gestion CIM/WBEM.

Ce travail se positionne dans le domaine de la gestion des réseaux et systèmes En effet la gestion des politiques de sécurité est un aspect de la gestion de la sécurité qui ne peut être isolée des autres composants participants à la bonne gestion IT (configuration, installation, performances, audit etc.)

Selon l'approche CIM, toutes les entités informatiques sont représentées sous la forme d'objets abstraits conçus selon les concepts de l'OMG (Object Managament Group) le DMTF qui est une association regroupant un grand nombre d'industriels et éditeurs informatiques, conçoit et publie également des profils CIM spécialisés pour des domaines de gestion précis, parmi lesquels on trouve : CIM Security Profile, CIM Policy Profile, CIM Integrated Access Control Policy Management Profile, CIM Simplified Policy language. Ce dernier a été adopté par l'IETF, l'organisation chargé de normaliser les protocoles Internet.

Dans le cadre de notre projet, nous avons adapté le protocole IPSec au standard CIM/WBEM en prenant en compte les spécificités de ce protocole et en se conformant aux exigences du DMTF en ce qui concerne la réutilisation de profiles existants. Une bonne étude et maitrise du protocole IPSec était nécessaire pour atteindre cet objectif et dont une synthèse est présentée au chapitre 3.

Nous avons ensuite repris les modèles CIM adéquats pour les étendre avec les éléments permettant d'exprimer les caractéristiques et règles IPSec. Une fois notre modèle CIM spécifique validé, nous avons procédé à son implémentation en commençant par l'installation de l'environnement de gestion WBEM (via le projet OpenPegasus) et puis nous avons apporté notre contribution à ce système par le développement de deux composants logiciels intégrées à l'environnement existant :

Le provider WBEM qui implémente les classes et associations de notre modèle de gestion de politiques appliquées à IPSec. Il s'agit du coeur de notre système réalisé conformément à la spécification CMPI (Common Manageability Programming Interface) pour lui permettre de

s'intégrer naturellement et d'une manière transparent à n'importe quel serveur CIM/WBEM existant.

Notre provider utilise les objets CIM dérivés de notre modèle pour modéliser des politiques de sécurités au lieu d'utiliser les règles natives de l'outil. D'un autre côté, et vu le caractère distribué de l'environnement CIM/WBEM, notre application permet de gérer, naturellement, des systèmes distants qui utilisent IPSec même si les systèmes sont hétérogène pourvu qu'un serveur WBEM soit aussi installé sur ces systèmes au sein duquel nos modèles sont compilés et notre Provider installé.

L'aspect standardisé de ce travail lui permet de s'intégrer dans un écosystème concourant à l'unification de la gestion de l'intégralité d'une infrastructure IT, notamment via une politique de sécurité globale exprimée d'une manière homogène (en utilisant le métamodèle CIM).

Comme perspective à notre travail, nous recommandons d'améliorer les fonctionnalités de notre Provider afin de traiter la totalité des politiques de IPSec (par exemple prend en considération la négociation IKE), il est possible aussi de réaliser une interface qui permet une meilleur exploitation des fonctionnalités offert par notre provider.

Acronymes:

IPSec: Internet Protocol Security.

SAD: Security Association Database.

SPD: Security Policy Database.

DMTF: Distributed Management Task Force.

WBEM: Web Based Enterprise Management.

CIM: Common Information Model.

CIMOM: CIM Object Manager.

CMIP: Common Management Information Protocol.

VPN: Virtual Protocol Network.

DMI: Desktop Management Information.

HTTP: Hyper-Text Transport Protocol.

MIB: Management Information Base.

SNMP: Simple Network Management Protocol.

TMN: Telecommunication Management Network.

UML: Unified Modelling Language.

USB: Universal Serial Bus.

WMI: Windows Management Interface.

XML: eXtensible Markup Language.

XACML: eXtensible Access Control Markup Language.

OMG: Object Managament Group

CMPI: Common Manageability Programming Interface.

CIM-SPL: CIM Simplified Policy Language.

NAT: Network Address Translation.

SCADA: Supervisory Control and Data Acquisition

PPTP: Point-to-Point Tunneling Protocol

TLS: Transport Layer Security

SSPI : Security Support Provider Interface

Références bibliographique :

[1] Labouret, G., & HSC, H. S. C. (1999, avril). LA SÉCURITÉ RÉSEAU AVEC IPSEC (NETWORK SECURITY WITH IPSEC) Consulté le: 26/04/2017, sur : https://www.google.dz/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=0ahUKEwjrotCVmdDWAhXGWCYKHYhbA28QFggmMAE&url=http%3A%2F%2Fwww.hsc.fr%2Fressources%2Farticles%2Fipsec-tech%2Findex.html.fr&usg=AOvVaw3wjto-JD7PUodWU5EZwVVR

[2] Charles M. Kozierok. IPSec Architectures and Implementation Methods. The TCP/IP Guide, consulté le 25/05/2017 sur : https://www.google.dz/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwiXzff5m9DWAhUH8CYKHaRlBGYQFggkMAA&url=http%3A%2F%2Fwww.tcpipguide.com%2Ffree%2Ft_IPSecArchitecturesandImplementationMethods-2.htm&usg=AOvVaw2mpUJLHywPbl1H2Us9cdxT

[3] Labouret, G., & HSC, H. S. C. (2000). IPSEC: Présentation technique. *Hervé Schauer Consultants (HSC). URL: www. hsc. fr*.
(http://www.univ-tebessa.dz/fichiers/master/master_1499.pdf)

[4] S. Frankel & S. Krishnan. (2011, fevrier). IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap consulté le 25/05/2017 sur: https://tools.ietf.org/html/rfc6071

[5] SecutiteInfo.com. (2 Septembre 2002). IPSEC : Internet protocol security, consulté le 26/05/2017 sur : https://www.researchgate.net/profile/Patrick_Martineau/publication/238664001_Integration_du_protocole_IPsec_dans_un_reseau_domestique_pour_securiser_le_bloc_des_sous-reseaux_FAN/links/543fabda0cf21227a11a6845/Integration-du-protocole-IPsec-dans-un-reseau-domestique-pour-securiser-le-bloc-des-sous-reseaux-FAN.pdf

[6] MINISTRE, P. Note technique Recommandations pour un usage sécurisé d'(Open) SSH.

[7] IP SECURITY (IPSEC) PROTOCOLS. Consulté le: 26/04/2017, sur: https://www.google.dz/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwjL_tHumNDWAhUJNiYKHajmAs4QFggkMAA&url=http%3A%2F%2Fwww.tcpipguide.com%2Ffree%2Ft_IPSecArchitecturesandImplementationMethods-3.htm&usg=AOvVaw3fqApDABJl_lGGPsxdRvPd.

[8] Cisco. (aout, 2007). Introduction to Cisco IPsec Technology Consulté le : 26/04/2017, sur :

https://www.cisco.com/c/en/us/td/docs/net_mgmt/vpn_solutions_center/2-0/ip_security/provisioning/guide/IPsecPG1.html#wp1002608

[9] IPsec par Brieuc Jeunhomme disponible sur l'url : http://www.frameip.com/ipsec/

[10] Laborde, R. (2005). Un cadre formel pour le raffinement de l'information de gestion de sécurité réseau : Expression et Analyse. Université Toulouse 3. Consulté le: 26/04/2017, sur: https://www.irit.fr/~Romain.Laborde/memoires/TheseRomainLaborde.pdf

[11] Bouabid.M.L, Semmar.M, Malek.M. (2015, juin). Conception et réalisation d'un système standardisé de gestion de politiques de sécurité. Consulté le: 15/05/2017, sur: https://www.researchgate.net/figure/281819821_fig9_Figure-2-1-Gestion-WBEM-homogene-d%27un-reseau-heterogene

[12] CIM Tutorial. (2003). Distributed Management Task Force, Consulté le: 15/05/2017, sur: http://www.dmtf.org/documents/user-and-security-model-white-paper-270

[13] Distributed Management Task Force, CIM Tutorial. Consulté le: 15/05/2017, sur: http://www.wbemsolutions.com/tutorials/DMTF/dmtftutorial.pdf, 2003.

[14] Festor, O., & Youssef, N. B. (2000). *Wbem* (Doctoral dissertation, INRIA).

[15]Distributed Management Task Force, DSP0221, Managed Object Format (MOF), Version: 3.0.0. Consulté le : 04/06/2017, sur : https://www.dmtf.org/sites/default/files/standards/documents/DSP0221_3.0.0.pdf, 13 Décembre 2012.

[16] Wikiwand. Policy-based management. L'url: https://www.wikiwand.com/en/Policy-based_management

[17] Rana, A. I., & Foghlú, M. Ó. (2009, December). New role of policy-based management in home area networks-concepts, constraints and challenges. In *New Technologies, Mobility and Security (NTMS), 2009 3rd International Conference on* (pp. 1-6). IEEE.

[18] Scherling, M., Huynh, A. N., Carlson, M., Westerinen, A., Quinn, B., Herzog, S., ... & Schnizlein, J. (2001). Terminology for Policy-Based Management. *Terminology*.

[19] CentOS. Consulté le: 15/05/2017, sur: https://en.wikipedia.org/wiki/CentOS.

[20] The Open Group. Consulté le: 20/05/2017, sur: http://www.opengroup.org/.

[21] Ranc, D. (2004). Gestion de réseaux et services. Consulté le: 15/05/2017, sur: http://www-lor.int-evry.fr/~agirs/gestion-reseaux-v2.03.pdf

[22] SimpleWbem. Consulté le: 15/05/2017, sur: http://www.simplewbem.org.

[23]cimcli-command line WBEM Client, Consulté le: 15/05/2017, sur: http://cvs.opengroup.org/cgibin/viewcvs.cgi/pegasus/src/Clients/cimcli/doc/Attic/cimcli.html?hidecvsroot=1&search=None&hideattic=1&sortby=file&logsort=date&rev=1.1&diff_forma=h

[24] Ward, D. Gouault, C. Dichtel, N. ip-xfrm: transform configuration. Consulté le: 26/06/2017, sur: https://www.systutorials.com/docs/linux/man/8-ip-xfrm/.

[25] VPN (Virtual Private Network) : définition, traduction et acteurs Fiche pratique Consulté le: 15/04/2017, sur: http://www.journaldunet.com/solutions/pratique/dictionnaire-du-webmastering/reseau/19567/vpn-virtual-private-network-definition-traduction-et-acteurs.html