

# Conception et implémentation d'un module de sécurité pour les réseaux ATM

Olivier PAUL  
Rapport de stage de DEA

Etude effectuée dans le département  
Réseaux et systèmes multimédias  
de Telecom Bretagne  
sous la direction de Mr. Pierre ROLIN

Septembre 1996



## Remerciements

Je tiens tout d'abord à remercier Monsieur Pierre ROLIN pour son accueil ainsi que pour ses conseils qui m'ont toujours été précieux.

Je remercie également Madame Maryline LAURENT pour l'aide et les conseils qu'elle m'a apporté tout au long de mon stage.

Je remercie enfin tout le personnel de l'équipe RSM pour son accueil, et en particulier Monsieur Hossam AFIFI pour l'aide qu'il a bien voulu m'accorder lorsque j'ai rencontré des difficultés techniques.



# Table des matières

<b>1</b>	<b>Introduction</b>	<b>8</b>
<b>2</b>	<b>ATM Notions de base</b>	<b>10</b>
2.1	Introduction . . . . .	10
2.2	Structure générale des couches physiques . . . . .	13
2.2.1	Sous-couche PM . . . . .	13
2.2.2	Sous-couche TC . . . . .	13
2.3	Structure générale de la couche ATM . . . . .	13
2.4	Structure générale des couches AAL . . . . .	15
2.4.1	l'AAL 1 . . . . .	15
2.4.2	l'AAL 2 . . . . .	15
2.4.3	l'AAL 3/4 . . . . .	16
2.4.4	l'AAL 5 . . . . .	16
2.5	Les différents types de flux . . . . .	17
2.5.1	Le flux usager . . . . .	17
2.5.2	La signalisation . . . . .	17
2.5.3	Les flux de gestion . . . . .	20
<b>3</b>	<b>Etat de l'art</b>	<b>22</b>
3.1	Les problèmes de sécurité des réseaux ATM . . . . .	22
3.1.1	Définitions . . . . .	22
3.1.2	Les risques encourus par l'utilisation des flux ATM . . . . .	23
3.2	Les solutions proposées . . . . .	26
3.2.1	Service d'authentification . . . . .	26
3.2.2	Service de confidentialité . . . . .	28
<b>4</b>	<b>Solution choisie</b>	<b>30</b>
4.1	Introduction . . . . .	30
4.2	Spécifications du module de sécurité . . . . .	31
4.3	Modèle de sécurité proposé . . . . .	33
4.4	Intégration de la sécurité dans le plan de contrôle . . . . .	33
4.4.1	Les modules . . . . .	35

4.4.2	Les bases de données . . . . .	36
4.4.3	L'IE de sécurité . . . . .	37
4.4.4	Fonctionnement . . . . .	41
4.5	Intégration de la sécurité dans le plan usager . . . . .	44
4.5.1	La norme IEEE 802.10 . . . . .	44
4.5.2	Format des trames IEEE 802.10 . . . . .	45
4.5.3	Application de la norme IEEE 802.10 au modèle ATM . . . . .	48
<b>5</b>	<b>Mise en œuvre</b>	<b>50</b>
5.1	Présentation du driver SUN . . . . .	50
5.1.1	Présentation des STREAMS . . . . .	50
5.1.2	Fonctionnement de TCP/IP . . . . .	51
5.1.3	La signalisation . . . . .	52
5.2	Implémentation . . . . .	52
5.2.1	Modifications réalisées dans TCP sur ATM . . . . .	53
5.2.2	Modifications touchant à la sécurité . . . . .	54
5.3	Utilisation de La politique de sécurité . . . . .	56
<b>6</b>	<b>Conclusion</b>	<b>58</b>
<b>7</b>	<b>Annexe</b>	<b>62</b>
7.1	Présentation générale du driver SUN . . . . .	62
7.1.1	Architecture des drivers dans Solaris . . . . .	62
7.1.2	Description du package ATM . . . . .	65
7.1.3	Outils de test . . . . .	67
7.2	Structure et Implémentation . . . . .	67
7.2.1	Fonctionnement de la signalisation . . . . .	67
7.2.2	Structure d'IP . . . . .	71
7.2.3	Structure de TCP . . . . .	72
7.2.4	Fonctionnement des TLIs . . . . .	75
7.3	Modification de la signalisation . . . . .	76
7.3.1	libcrypton.a et libcryptok.a . . . . .	78
7.3.2	libatm.a et libatmk.a . . . . .	79
7.3.3	Dans les fichiers d'en-tête . . . . .	82
7.3.4	Dans les fichiers <i>qcc_k.c</i> , <i>qcc_u.c</i> et <i>qcc_user.c</i> . . . . .	83
7.3.5	Dans Q93B . . . . .	83
7.3.6	Dans AAR . . . . .	83
7.3.7	Dans tstqcc . . . . .	84
7.4	Utilisation des services de sécurité . . . . .	85
7.4.1	Introduction . . . . .	85
7.4.2	Schéma général . . . . .	85
7.4.3	Modification au niveau utilisateur . . . . .	87
7.4.4	Le module ONIP . . . . .	88
7.4.5	Modification de TCP . . . . .	90

7.4.6	Modification d'IP . . . . .	91
7.4.7	Les programmes annexes à ONIP : laneplumb, laneun- plumb, secplumb et intset . . . . .	92
7.4.8	Utilisation de la politique de sécurité modifiée . . . . .	93



# Chapitre 1

## Introduction

La sécurité dans les réseaux n'a pas toujours été considérée comme primordiale. Cependant, au cours des années plusieurs facteurs ont conduit à sa prise en compte. La modification du profil des utilisateurs a entraîné l'apparition de nouveaux types de trafics, plus sensibles (transactions bancaires, casiers judiciaires, dossiers médicaux ...). Ces nouveaux trafics peuvent être sujets à plusieurs types d'attaques :

- L'écoute passive de données circulant sur le réseau.
- Modifications, suppression ou insertion de données.
- Saturation du réseau.
- etc ...

Pour répondre à ces problèmes des services de sécurité tels que le contrôle d'accès, l'authentification, l'intégrité, la confidentialité ont été mis en œuvre.

Cependant, du fait de l'apparition progressive de la demande concernant ces services, aucun d'entre-eux n'a jamais été pris en compte dès la conception des protocoles. Les services de sécurité étant rajouté "après-coup". Ceci a eu pour conséquences un certain nombre de défaillances des protocoles actuels en matière de sécurité. (Voir par exemple [CB94] en ce qui concerne les problèmes au niveau des protocoles TCP/IP).

La technologie ATM est considérée comme celle qui demain assurera le transport des données de bout en bout car elle permet de transporter des données de tout type (voix, données, images). Cependant, cette technologie, toujours en cours de normalisation n'intègre pour l'instant aucun de ces services de sécurité. A l'instar de ce qui a été fait dans IPV6, il est donc devenu primordial d'intégrer au cours de cette normalisation les impératifs de sécurité en introduisant les services à fournir dans l'ATM.

Deux organisations internationales, l'ATM Forum (Groupe AF-Security) et l'ITU (groupe 11-Q29) en sont chargés. La première devrait rendre ses spécifications en février 1997.

Dans ce rapport nous étudierons le fonctionnement général d'ATM ainsi que les problèmes qui lui sont liés, l'état de l'art concernant les risques et les solutions en la matière, la solution proposée par Maryline Laurent ainsi que l'implémentation correspondante.

## Chapitre 2

# ATM Notions de base

Dans ce chapitre nous décrirons rapidement le fonctionnement d'ATM [Ro195] fournit des informations plus détaillées.

### 2.1 Introduction

Au début des années 80, les équipes du CNET de Lannion mirent au point les principes d'ATM. Ce protocole supporte non seulement une grande diversité de trafics (voix, image, données ...) mais il s'adapte également à un grand nombre de supports physiques. Afin de fournir cette diversité le protocole ATM utilise des couches de convergence vis à vis du support physique (couches physiques) et vis à vis des applications (couches AAL).

Le protocole ATM est un protocole orienté connexion ; Comme dans tout protocole orienté connexion, la communication ATM se passe en trois temps : établissement de la connexion, transfert des données et fermeture de la connexion. La phase d'établissement de la connexion permet au réseau de réserver les ressources nécessaires si elles sont disponibles. Si les ressources nécessaires sont insuffisantes, la connexion est refusée au terminal appelant. Ce mode de fonctionnement orienté connexion permet au réseau de garantir la qualité requise par le client.

L'unité de donnée utilisée par le protocole ATM est la cellule. La cellule est un paquet dont la taille est de 53 octets. Elle est constituée de deux parties : une partie donnée de 48 octets et une partie en-tête de 5 octets. L'unité de donnée a les caractéristiques suivantes :

- La longueur du champ d'information est relativement réduite :  
Afin de réduire les mémoires tampons internes des noeuds de commutation et limiter les délais subis par les files d'attente dans ces tampons, la longueur du champ d'information reste relativement réduite.

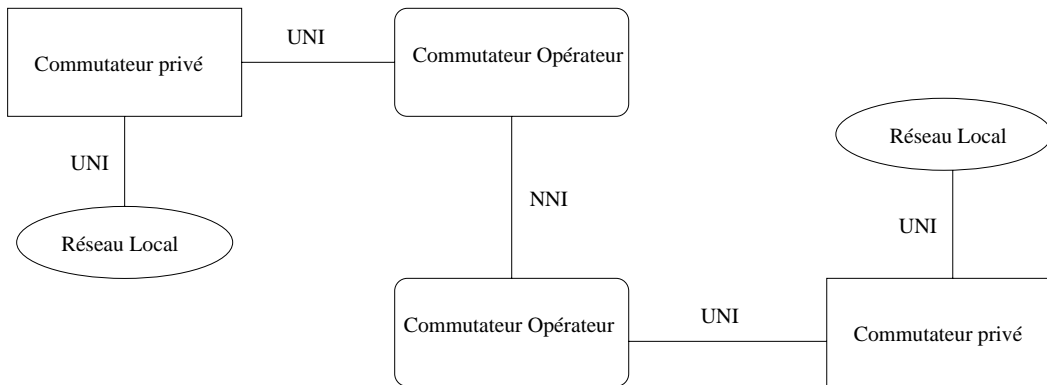


FIG. 2.1 - *Les interfaces réseau*

- Fonctionnalité réduite de l'en-tête :  
 Pour garantir un traitement rapide sur le réseau, l'en-tête ATM a une fonction très limitée. Elle permet d'identifier la connexion virtuelle par un identificateur sélectionné lors de l'établissement de l'appel et de garantir un routage approprié de chaque cellule sur le réseau. En outre, elle permet le multiplexage facile des différentes connexions virtuelles sur une seule liaison. Enfin l'en-tête permet de détecter et parfois de corriger les erreurs sur l'en-tête au moyen d'un champ HEC. Ce champ n'a été prévu ni pour la détection d'erreur sur le champ de données, ni pour la détection de perte de cellules car les supports sont supposés de bonne qualité.

ATM est destiné à interconnecter aussi bien des abonnés terminaux que des réseaux publics ou privés. La taille du réseau tant en nombre de points d'accès que géographique ne saurait être limitée. L'architecture du réseau est maillée. Ces interconnexions sont permises par trois types d'interfaces :

- NNI :  
 Network to Network interface. Ce sont les connexions internes au réseau entre deux composants de l'exploitant ou entre les réseaux de deux exploitants.
- UNI :  
 User to Network Interface. Ce sont les interfaces entre l'abonné au réseau public et l'opérateur.
- B-ICI :  
 Celle-ci n'est pas encore clairement définie. Elle se situe entre opérateurs.

La figure 2.1 montre les différentes interfaces réseau. La figure 2.2 présente les différents plans du modèle de référence du protocole ATM.

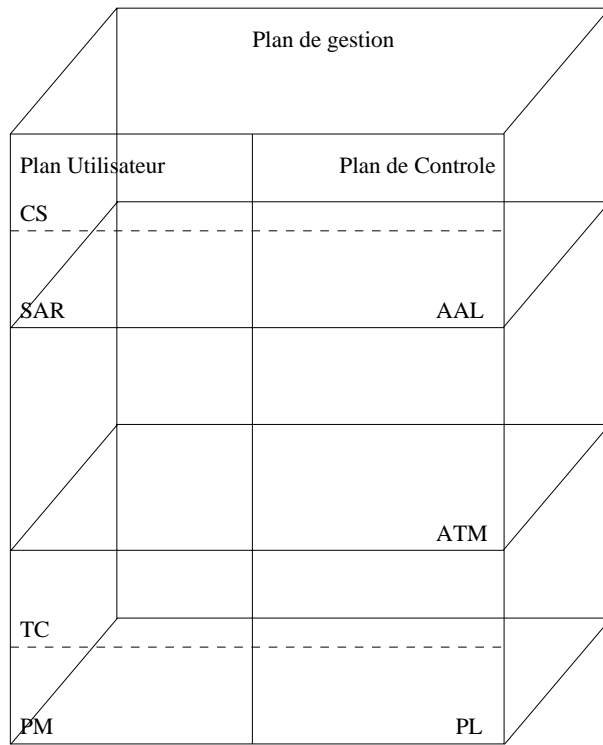


FIG. 2.2 - *Modèle de référence ATM*

GFC/VPI	VPI		
VPI	VCI		
VCI	VCI		
VCI	PT	RES	CLP
HEC			

FIG. 2.3 - *En tête des cellules ATM*

## 2.2 Structure générale des couches physiques

On peut séparer les couches physiques en deux sous couches :

### 2.2.1 Sous-couche PM

La couche Physical Medium ne prend en charge que des fonctions spécifiques du support. Elle permet la transmission de bits : codage, alignement ...

### 2.2.2 Sous-couche TC

La couche Transmission convergence convertit le flux des cellules ATM en bits à transporter sur un support physique. La sous-couche TC se charge aussi de générer et de récupérer les trames de transmission. De même, elle génère et vérifie l'information de contrôle d'erreur de l'en-tête et fait l'adaptation à la trame de transmission.

## 2.3 Structure générale de la couche ATM

La couche ATM réalise les fonctions suivantes :

- Multiplexage et démultiplexage des cellules des couches AAL et des couches physiques. Cette fonction consiste à diriger les cellules reçues selon des règles de priorité.
- Gestion de l'en-tête des cellules. Ceci comprend la génération et l'extraction de la partie correspondant à la couche ATM, c'est à dire les 4 premiers octets de l'en-tête. La figure 2.3 montre les différents champs de l'en-tête d'une cellule ATM.

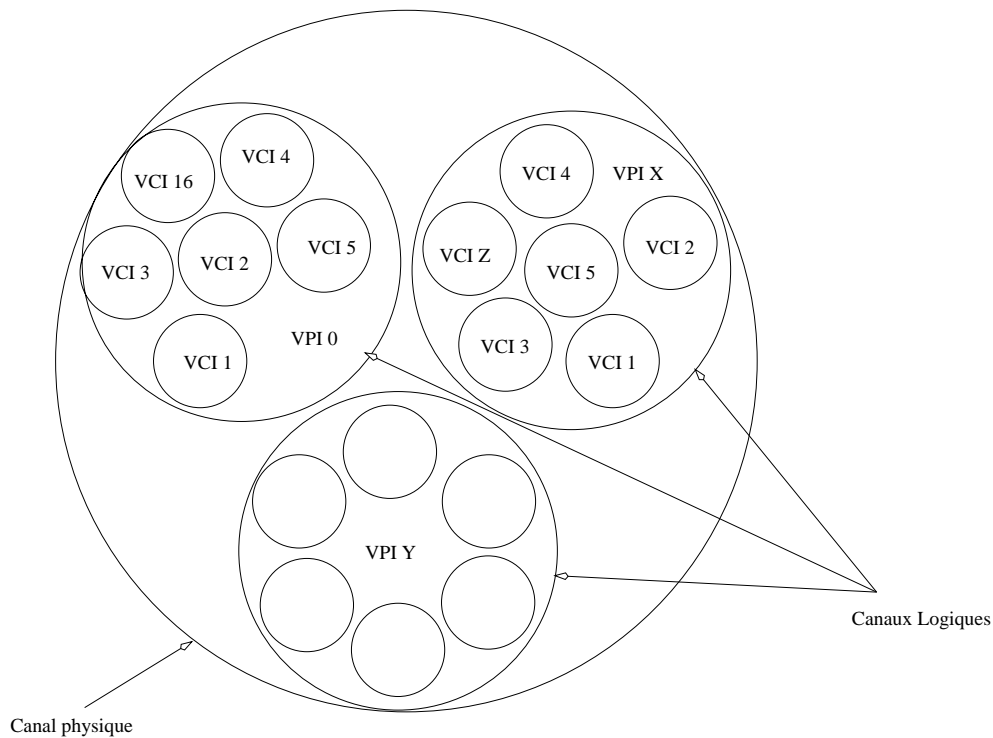


FIG. 2.4 - *Multiplexage de flux ATM*

- Commutation ou relayage. Cette fonction est réalisée grâce au couple VCI/VPI. La cellule est soit dirigée vers une couche AAL locale, soit relayée vers un autre commutateur dans le cas où elle n'a pas atteint sa destination finale.
- Fourniture à l'utilisateur d'une voie virtuelle ou d'un faisceau virtuel une classe de QoS parmi un certain nombre de classes prises en charge par le réseau.
- Mise en place de mécanismes de contrôle de flux afin de prévenir des situations de congestion.
- Fourniture de fonctions d'administration.

Le routage s'effectue dans ATM au moyen d'un couple de valeurs VPI/VCI contenu dans l'en-tête des cellules. Au niveau de chaque commutateur, ces valeurs peuvent être associées à une connexion. On peut donc distinguer deux types de connexions ; Les VPC (ou Virtual Path Connection) qui relient un ou plusieurs liens de VP et les VCC (ou Virtual Channel Connection) qui relient

un ou plusieurs liens de VC. Il est possible de considérer les VPC comme des agrégats de VCC.

Au passage dans un commutateur, les valeurs de ces champs seront traduites à partir d'une table propre au commutateur et la cellule sera routée.

Afin d'accélérer la traduction, certains commutateurs appelés brasseurs ne considèrent que le champ VPI quand celui-ci est utilisé.

La figure 2.4 représente un exemple des relations pouvant exister entre VPC et VCC.

## 2.4 Structure générale des couches AAL

Cette couche est destinée à améliorer la qualité du service fournie par la couche ATM.

### 2.4.1 l'AAL 1

Les objectifs de l'AAL1 sont les suivants :

- Transférer des unités de données issues de sources à débit constant et qui seront délivrées au même débit.
- Transférer des informations de structuration et de synchronisation entre la source et la destination.
- Indiquer des pertes éventuelles.

#### sous-couche SAR

Elle assure les propriétés de séquençement et de détection d'erreurs.

#### sous-couche CS

Elle assure la restitution au récepteur d'un signal d'horloge produit par l'émetteur. Les fonctions assurées par cette couche sont fortement dépendantes du type de service traité.

### 2.4.2 l'AAL 2

Les objectifs de l'AAL2 sont les suivants :

- Transférer des informations à débit variable.
- Transférer des informations de synchronisation.
- Indiquer les pertes ou les erreurs sans reprise.

### 2.4.3 l'AAL 3/4

Les objectifs de l'AAL3/4 sont les suivants :

- Fragmenter et réassembler les paquets émis par différents protocoles.
- Véhiculer des espaces d'adresses totalement différents.
- Offrir un service point à point et un service multipoint.

#### sous-couche SAR

Elle assure les fonctions de segmentation et de réassemblage permettant de transférer plusieurs SAR-SDU de longueur variables de façon simultanée sur une même connexion ATM entre deux entités AAL. Cette sous-couche est commune à tous les espaces d'adressage.

#### sous-couche CPCS

Cette sous-couche identifie l'espace d'adressage utilisé.

#### sous-couche SPCS

Elle est propre à chaque espace d'adressage. Elle peut ne pas exister.

### 2.4.4 l'AAL 5

L'AAL5 est une adaptation de l'AAL3/4 pour les besoins des fabricants de matériel, elle a donc exactement les mêmes objectifs.

#### sous-couche SAR

Elle assure les fonctions de segmentation et de réassemblage ainsi qu'un contrôle d'intégrité.

#### sous-couche CPCS

Elle assure un service à datagramme.

#### sous-couche SSCOP

Elle assure

- Délivrance des données.
- Transmission de données de taille variable.
- Garantie de l'intégrité du séquençement des paquets.

- Retransmission des paquets erronés.
- Contrôle de flux.
- Remontées d'erreurs.
- Messages de maintien de connexion.
- Contrôle des communications.
- Rapport d'état des voisins.

#### **sous-couche SSCF**

Cette sous-couche assure la spécification de primitives conformes au modèle OSI offertes aux utilisateurs du service.

## **2.5 Les différents types de flux**

[Rol95] distingue trois types de flux :

### **2.5.1 Le flux usager**

Ce flux est constitué de cellules comprenant :

- Une en-tête indiquant entre autres le VC de transit de la cellule (VPI/VCI). Toutes les valeurs VCI/VPI exceptées celles utilisées par les autres types de flux sont possibles.
- Une charge utile, i.e. des informations uniquement prises en compte par les couches hautes des entités d'extrémités. L'en-tête permet d'assurer le bon acheminement des cellules au travers du réseau intermédiaire. Le réseau l'analyse tout au long du parcours de la cellule. Par contre pour la charge utile, le réseau public est complètement transparent.

Par la suite, ce flux sera appelé flux information.

### **2.5.2 La signalisation**

Il existe trois catégories de flux de signalisation: la méta signalisation, la signalisation utile à la diffusion et la signalisation point à point. Cependant, tandis que la signalisation point à point doit impérativement être prise en compte par les équipements de communication, les deux autres sont optionnelles et peuvent même être supprimées au niveau d'un équipement qui ne les traite pas. La figure 2.4 montre une représentation des flux ATM par canaux logiques.

### **La méta-signalisation**

Elle permet principalement d'établir, libérer, contrôler et vérifier l'état des canaux utilisés pour le transport de la signalisation point à point et point à multipoints. Pour les connexions VCC permanentes, cette méta-signalisation est inexistante. Les cellules correspondantes sont caractérisées par un numéro de VCI = 1. Le VPI associé vaut 0 par défaut. Cette valeur par défaut de VPI est réservée pour la communication entre une entité d'extrémité et le commutateur local. N'importe quelle valeur de VPI peut être choisie pour les autres types de communication.

### **La signalisation utile à la diffusion**

Elle permet de diffuser des informations de signalisation lorsque le réseau ATM est utilisé en mode diffusion. Ce type de signalisation n'est pas utilisé sur les connexions VCC permanentes. Les cellules correspondantes sont caractérisées par un numéro VCI = 2. Le VPI associé vaut 0 par défaut. Cette valeur par défaut de VPI est réservée pour le dialogue entre une entité d'extrémité et le commutateur local, n'importe quelle valeur de VPI peut être choisie pour les autres types de communication.

### **La signalisation point à point**

Elle permet d'établir, libérer, gérer et superviser la connexion. Ces informations de signalisation sont de deux sortes:

- la signalisation utile à l'opérateur public pour établir correctement la connexion (routage, QoS...) au travers des commutateurs du réseau public. On l'appellera par la suite: signalisation réseau (public) (En fait cette notation englobe les deux autres flux de signalisation). Les cellules correspondantes sont caractérisées par leur VCI = 5. Le VPI associé vaut 0 par défaut. Cette valeur par défaut de VPI est réservée pour la communication entre une entité d'extrémité et le commutateur local. N'importe quelle valeur de VPI peut être choisie pour les autres types de communication.
- la signalisation uniquement prise en compte au niveau des entités d'extrémités de la connexion. Elle sera appelée: signalisation de bout en bout. Les cellules sont portées sur les mêmes VCI et VPI que les informations utilisateur. Les spécifications de cette signalisation ne sont pas encore élaborées.

Comme le montre la figure 2.5 le plan de signalisation utilise les services de communication de l'AAL5 pour transmettre ses données protocolaires. Celle-ci assure la segmentation et le réassemblage des messages car ceux-ci peuvent excéder la taille d'une cellule. Cependant du fait que l'AAL5 ne garantit pas

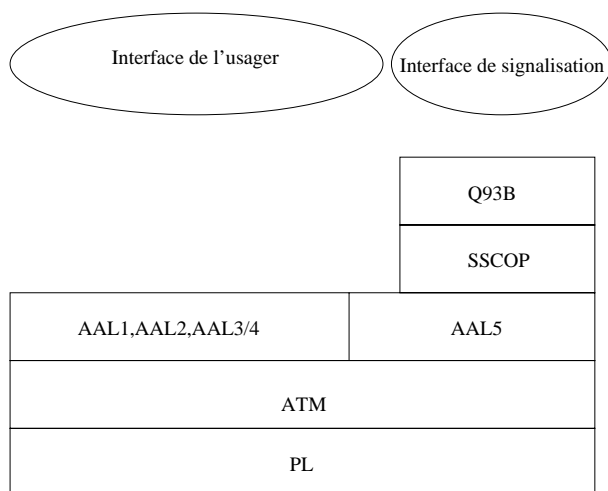


FIG. 2.5 - Place de la signalisation dans la pile protocolaire d'ATM

l'absence de perte de messages, un protocole sur connexion, SSCOP est utilisé. Nous avons décrit les services qu'il rend dans la partie précédente.

L'ATM Forum a proposé une spécification des services à l'interface UNI. Cette spécification est décrite dans [For94] et couvre la signalisation, celle-ci étant appelée Q93B.

Q93B rend les services suivants :

- Demande d'établissement de connexion.
- Gestion des paramètres de connexion.
- Affectation des valeurs de VCI/VPI à la connexion établie.
- Détection d'erreurs aux différents niveaux du protocole mis en œuvre.
- Mécanisme d'enregistrement de l'adresse abonnée.
- Négociation des paramètres de bout en bout.

Afin de rendre ces services, Q93B utilise un protocole d'échange de messages. Chaque message est typé et est constitué d'un ensemble bien définis d'éléments d'information (IE) qui définissent les paramètres relatifs à celui-ci. Ces IEs peuvent être utilisés dans plusieurs types de messages. Parmi les différents types de messages nous détaillerons les messages d'établissement et de libération des connexions bipoints :

- **SETUP** :  
Ce message est émis par l'appelant vers le réseau et par le réseau vers l'appelé afin d'établir la connexion.

- CALL PROCEEDING :  
Ce message est émis par l'appelé vers le réseau et par le réseau vers l'appelant afin de lui indiquer que la demande de connexion est en cours.
- CONNECT :  
Ce message est émis par l'appelé vers le réseau et par le réseau vers l'appelant afin de lui indiquer que la connexion est établie.
- CONNECT\_ACK :  
Ce message est émis par le réseau vers l'appelé afin de lui confirmer l'appel.
- RELEASE :  
Ce message est émis vers ou par le réseau pour demander ou indiquer une coupure de connexion.
- RELEASE COMPLETE :  
Ce message est émis vers le réseau pour indiquer que le canal virtuel a été libéré.

### 2.5.3 Les flux de gestion

Dans ces flux de gestion, on distingue le flux ILMI (interface provisoire de gestion locale) et le flux OAM. Le flux ILMI caractérisé par défaut par le VC : VPI = 0 et VCI = 16, permet à une entité d'échanger des informations de la MIB avec une autre entité.

Le flux OAM comprend 5 flux hiérarchiques F1, F2, F3, F4 et F5. En se référant à la recommandation UIT-T I.610, ils sont chargés de :

- contrôler la qualité de fonctionnement,
- détecter les défauts et les dérangements,
- protéger les systèmes,
- informer des dérangements,
- localiser les dérangements.

Les flux F1, F2 et F3 sont utilisés par la couche physique aux niveaux des sections de régénération (F1), des sections numériques (F2) et des trajets de transmission (F3). Leur implémentation dépend du format de système de transmission utilisé : systèmes en hiérarchie numérique synchrone (SDH), en hiérarchie numérique plésiochrone (PDH) ou par cellules. Comme ces flux sont situés au niveau de la couche physique ils sont totalement dépendants des systèmes de transmission choisis.

Les flux F4 et F5 sont utilisés par la couche ATM pour gérer les VPs (F4) et les VCs (F5). Ils renseignent l'administrateur sur les problèmes de connexion (alarmes) et peuvent vérifier l'état des connexions (méthode du rebouclage). Ils n'interviennent qu'une fois que la connexion a été établie.



# Chapitre 3

## Etat de l'art

Dans ce chapitre, nous présentons les travaux qui ont été effectués jusqu'à présent sur la sécurité dans ATM. Pour cela nous nous sommes inspirés pour ce chapitre ainsi que le suivant des articles [ML95], [Lau96c], [Lau96b] et [Lau96a].

### 3.1 Les problèmes de sécurité des réseaux ATM

Dans cette section, nous mettons en évidence les lacunes en matière de sécurité dans les réseaux ATM, c'est à dire, nous analysons l'impact qu'un intrus peut avoir sur les communications ATM ou sur le réseau ATM en agissant sur le flux. En préalable nous proposons quelques définitions sur les menaces issues de [ISO90].

#### 3.1.1 Définitions

##### Menaces classiques

- Atteinte à l'authentification de l'origine des données ou à l'authentification de l'entité homologue:  
Elle consiste pour une entité à se faire passer pour une autre soit au cours d'une procédure d'authentification spécifique soit lors d'une procédure d'authentification basée sur les données échangées.
- Atteinte à la confidentialité des données :  
Elle consiste pour une entité à tenter d'avoir accès à des données dont l'accès ne lui est pas autorisé.
- Atteinte à l'intégrité des données :  
Elle consiste pour une entité à chercher à modifier, supprimer, insérer ou rejouer des données de manière non autorisée.

- Atteinte à la disponibilité :  
Elle consiste pour une entité à mettre en œuvre des actions visant à empêcher que l'accès à un service offert par le système à une entité autorisée ne soit pas toujours possible.
- Atteinte au contrôle d'accès :  
Elle consiste pour une entité à essayer d'utiliser de manière non autorisée des ressources.
- Atteinte au secret du flux :  
Elle consiste pour une entité à tenter d'obtenir des informations de l'observation des flux de données.

### **Les canaux cachés**

Un canal caché est un chemin de communication non prévu et/ou non autorisé qui peut être utilisé pour transférer de l'information d'une manière qui viole la politique de sécurité du système d'information, c.à.d. un canal de données entre deux entités du réseau illégalement utilisé dans le but de transmettre des informations de manière invisible.

### **3.1.2 Les risques encourus par l'utilisation des flux ATM**

Afin de mieux mesurer le niveau de sensibilité de chacun des flux ATM, nous nous intéressons aux conséquences possibles que les menaces décrites dans la section précédente appliquées au niveau des cellules/messages impliquent sur le réseau.

#### **Flux utilisateur**

Les menaces envers le flux utilisateur sont de trois sortes :

- Menaces portant sur la confidentialité ou la confidentialité du flux de données :  
Par écoute passive du réseau, un intrus peut récupérer l'ensemble des cellules appartenant à une connexion en isolant celles possédant les mêmes couples VPI/VCI et reconstituer la totalité du message transféré. Cette technique permet également d'obtenir toutes les informations désirées sur le flux de données et constitue donc une menace envers la confidentialité du flux.
- Menaces portant sur l'intégrité :  
Un intrus peut également intervenir sur les cellules au cours de leur transfert en insérant, modifiant, supprimant des informations dans les cellules transmises ou en jouant des cellules déjà transmises. En général, ces cellules sont rejetées par le récepteur après le passage du test d'intégrité,

mais parfois, si ce test est passé avec succès, il peut arriver qu'elles soient traitées ce qui met en cause l'intégrité de tout le flux.

- Canaux cachés :  
L'intrus peut utiliser certains champs de la cellule pour construire des canaux cachés. Nous présentons ici trois canaux cachés, chacun correspondant à un champ particulier. Ces champs sont les suivants :
  - les bits de bourrage :  
Ils peuvent servir à transmettre de l'information en clair ou chiffrée.
  - les champs VCI/VPI :  
Certains commutateurs limitent l'espace VCI/VPI utile et ne traitent que les cellules de VCI/VPI y appartenant. Il est possible de réaliser un canal caché en choisissant deux adresses VCI/VPI hors de cet espace. Le codage binaire consiste à envoyer sur l'une ou l'autre de ces adresses, des cellules. Ces cellules sont rejetées au niveau du commutateur mais un intrus écoutant de manière passive les messages entre l'émetteur et le commutateur peut décoder le message secret.
  - le champ HEC :  
Il est possible de créer un canal caché en créant artificiellement des erreurs dans les en-têtes des cellules envoyées. Afin de ne pas provoquer des rejets de cellules pouvant attirer l'attention on n'utilise que des erreurs corrigibles. Un intrus placé entre l'émetteur et son commutateur local peut alors utiliser ce code pour décoder le message secret.

### **Flux de signalisation**

Les menaces portant sur le flux de signalisation varient suivant le type de message échangé. Ces menaces sont de trois sortes :

- Menace portant sur la confidentialité des données et du flux de données :  
En écoutant les messages SETUP dans le cas des communications bipoints ou ADD\_PARTY dans le cas des communications point à multipoints un intrus peut déduire les adresses des deux parties désirant établir une connexion ainsi des informations sur le trafic échangé et les applications utilisées. En écoutant les messages CONNECT et CALL\_PROCEEDING un intrus peut connaître les VCI/VPI correspondant à cette connexion et par la suite utiliser ce couple pour isoler les données relatives à la connexion dans le flot utilisateur. L'écoute de ces messages met en jeu à la fois la confidentialité de la signalisation, la confidentialité du flux de la signalisation mais également la confidentialité du flux usager.
- Menace portant sur l'intégrité :  
En modifiant le contenu des messages de signalisation ou de leurs en-têtes,

un intrus peut provoquer le rejet du message par le récepteur, une perte du message en cours de transfert, la perturbation d'une autre connexion, etc...

- Menace portant sur la disponibilité :  
On peut réaliser celle-ci de plusieurs manières :
  - En saturant une entité du réseau (commutateur, station de travail) par l'envoi de messages de signalisation multiples tels que SETUP.
  - En utilisant des messages RELEASE ou RELEASE COMPLETE afin de provoquer des coupures de connexions.
- Canaux cachés :  
De nombreux canaux cachés sont réalisables au moyen de la signalisation. On peut exemple :
  - Utiliser le contenu des messages.
  - Jouer sur la fréquence d'émission des messages de signalisation relativement au nombre de cellules utilisateur émises.

### **Flux de gestion**

Les menaces portant sur le flux de gestion sont de quatre sortes :

- Menace portant sur la confidentialité des données et du flux de données :  
En écoutant les messages du flux de gestion en transit sur le réseau, un intrus peut déduire des informations sur l'état du réseau mais aussi sur le nombre de cellules échangées. Ceci remet en compte la confidentialité du flux, du flux utilisateur.
- Menace portant sur l'intégrité :  
En modifiant le contenu des messages de gestion ou de leurs en-têtes, un intrus peut provoquer un dysfonctionnement du réseau.
- Menace portant sur la disponibilité :  
A la suite d'une attaque portant sur l'intégrité, un intrus peut provoquer des atteintes à la disponibilité. En effet celle-ci peut conduire à la non détection des erreurs sur la ligne, à la coupure de la ligne alors que celle-ci est tout à fait opérationnelle ou à une mauvaise localisation du problème de ligne.
- Canaux cachés :  
Ceux-ci sont réalisables en jouant sur des paramètres tels que la fréquence d'émission de cellules de gestion de qualité (Flux F1 décrit au chapitre précédent) relativement au nombre de cellules utilisateur émises.

## 3.2 Les solutions proposées

A ce jour, ni le groupe AF-Security de l'ATM Forum ni le groupe 11/29 de l'ITU n'ont émis de recommandations ou de spécifications officielles en matière de sécurité. Cependant l'ATM Forum qui mène principalement des travaux sur les services d'authentification et de confidentialité depuis un an prévoit l'aboutissement de ceux-ci en février 1997 et a émis des *drafts* concernant ces sujets en juin 96. A coté de ces organisations d'autres chercheurs s'intéressent à la sécurité des réseaux ATM. Nous présentons ici les idées émises par ces groupes de recherche et nous structurons notre analyse autour des services d'authentification et de confidentialité qui même s'ils ne sont pas les seuls en jeu donnent un bon aperçu des techniques utilisées.

### 3.2.1 Service d'authentification

En ce qui concerne le service d'authentification, l'ATM Forum propose trois approches en matière d'implémentation. Ces approches varient suivant que l'information d'authentification est envoyée sur le flux de gestion, sur le flux de signalisation ou dans un canal auxiliaire. Les solutions proposées ci-dessous permettent non seulement l'authentification mais également l'échange de paramètres de sécurité pouvant être utilisés par la suite (comme par exemple les clefs).

#### Authentification par le flux de signalisation

Plusieurs contributions de l'ATM Forum ainsi que Gong ([RD95]) proposent d'introduire cette information d'authentification dans le flux de signalisation à la demande d'ouverture de connexion dans un élément d'information du SETUP ou du CONNECT. Cette approche que nous appellerons par la suite solution 1 est décrite dans la figure 3.1. Cependant cette approche a de nombreux inconvénients :

- L'introduction de nouveaux éléments d'information de sécurité dans les messages de signalisation impliquent la modification des spécifications UNI, ce qui rend les constructeurs au sein de l'ATM Forum assez réticents car cela implique la modification et la révision de l'ensemble des équipements ATM - commutateurs, adaptateurs - actuellement sur le marché.
- D'autre part pour que cette solution soit applicable, il faut que les opérateurs consentent à faire transiter de bout en bout via la signalisation qui, notons-le pourrait être gratuite, des informations de sécurité de contenu inconnu pour eux qui pourraient très bien servir à faire passer des données utilisateurs gratuitement.
- La phase de négociation des paramètres de sécurité et d'authentification doit se dérouler en au plus deux échanges (messages SETUP et

CONNECT), ce qui interdit toute utilisation de protocole d'authentification forte (qui nécessite trois échanges) et requiert l'utilisation d'estampilles.

- Comme, pour une connexion donnée, cet échange d'informations de sécurité est réduit à la seule phase d'ouverture de connexion, cette technique ne peut pas être réutilisée pour effectuer une réauthentification et surtout une renégociation de paramètres de sécurité au cours de la communication.

### Autres solutions

Pour répondre au problème évoqué ci-dessus, l'ATM Forum a étudié deux approches. La première approche, présentée dans la figure 3.2 et que nous appellerons ensuite solution 2 consiste à introduire tous les paramètres de sécurité dans le flux de gestion F5 sur le même canal que la connexion à sécuriser. La seconde approche présentée dans la figure 3.3 et que nous appellerons ensuite solution 3 utilise une connexion parallèle mise en place au moment du SETUP. Cette seconde connexion permet d'échanger des paramètres de sécurité relatifs à la première connexion.

Ces deux approches ont par rapport à la solution proposée par Gong l'avantage de ne pas restreindre le nombre d'échanges de messages de sécurité et de permettre l'échange de messages de sécurité à la demande rendant ainsi possible la renégociation de paramètres de sécurité tels les clés au cours d'une connexion. En particulier pour la solution 2, du fait que les messages de sécurité transitent par le même canal que le flux utilisateur à protéger, cette solution assure la synchronisation entre les données utilisateur transmises et les informations de sécurité s'y rapportant, ce qui s'avère très utile en cas de resynchronisation du déchiffrement sur la station réceptrice ou en cas de changement de clés. Par contre on peut remarquer que cette propriété propre à la solution 2 rend cette solution inutilisable pour les connexions unidirectionnelles (point à multipoints) ou temps-réel.

Tout comme pour la solution 1, la solution 2 nécessite de modifier les spécifications UNI en définissant des cellules de gestion supplémentaires spécifiques à l'administration de la sécurité.

La solution 3 par contre ne requiert aucune modification car l'échange d'informations de sécurité se fait de façon transparente pour le réseau sur une connexion dédiée (canal auxiliaire). Cependant, cette solution nécessite deux ouvertures de connexions au lieu d'une seule, ce qui implique un accroissement du coût de la communication et d'autre part elle requiert que les deux parties qui veulent ouvrir une connexion sécurisée réussissent à gérer en parallèle deux connexions qui sont très dépendantes l'une de l'autre.

L'un des gros inconvénients des solutions 2 et 3 est de ne pas assurer la protection du flux de signalisation (par une authentification sur les adresses ATM par exemple) qui est comme nous l'avons vu dans la section précédente

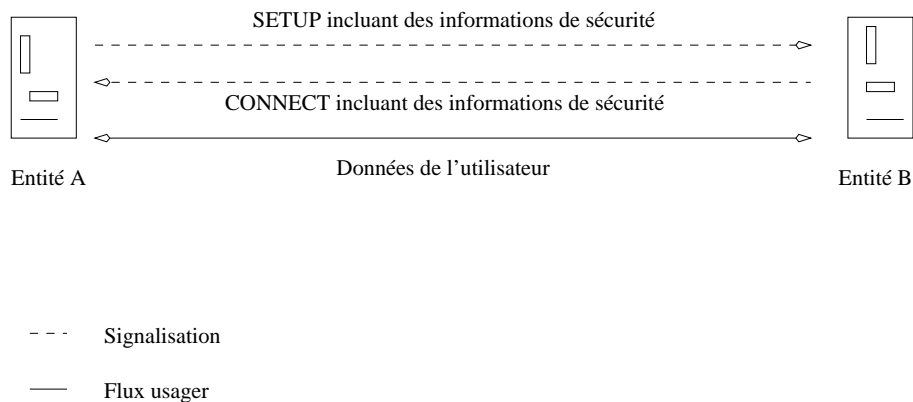


FIG. 3.1 - Protocole utilisé pour la solution 1

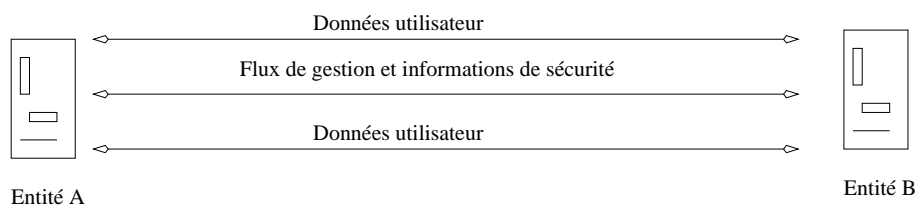


FIG. 3.2 - Protocole utilisé pour la solution 2

particulièrement sensible aux attaques en confidentialité, disponibilité/intégrité.

Comme on peut le voir la troisième solution est celle la plus propice à l'utilisation de proxys. Ceci peut également être envisagé dans les autres cas mais est moins utile.

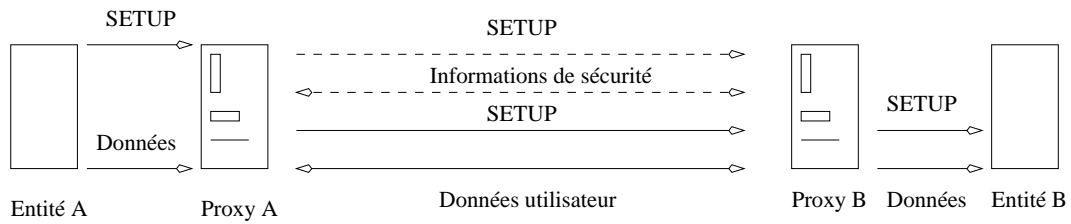
### 3.2.2 Service de confidentialité

En ce qui concerne le service de confidentialité, l'ATM Forum ainsi que Chuang ([Chu96]) proposent le placement du chiffrement au niveau de la couche ATM afin de permettre un chiffrement cellule par cellule.

Les principaux problèmes soulevés par le chiffrement concernent le choix de l'algorithme ainsi que la synchronisation entre la source et l'émetteur. Celle-ci a un double objectif

- recalculer régulièrement le déchiffrement des données afin de limiter les pertes de données utilisateur en cas d'erreurs.
- changer de clef en plein chiffrement de données.

Afin de répondre au premier problème, Gong ([RD95]) propose de placer le chiffrement au niveau de l'AAL entre la couche SAR et CS (le problème ne se



- - - - connexion 1  
 ——— connexion 2

FIG. 3.3 - Protocole utilisé pour la solution 3

posant pas car les données à chiffrer sont de grande taille). Cependant d'une part la distinction entre les sous-couches SAR et CS étant loin d'être évidente, il n'est pas sûr que l'implémentation de cette solution soit facile, d'autre part cette solution est moins sûre car elle laisse les en-têtes des trames AAL non chiffrées.

# Chapitre 4

## Solution choisie

### 4.1 Introduction

Le modèle de sécurité exposé ci-dessous a été guidé par l'idée de réaliser un système qui permettrait de sécuriser un RLE dans un environnement tout ATM de la même façon que les RLEs sont aujourd'hui protégés sur Internet par des *firewalls*.

Comme la technique des firewalls n'est pas directement applicable à l'ATM à cause du principe même de l'ATM qui dissocie le traitement de la signalisation de celui des données utilisateur, nous avons défini un module de sécurité qui serait intégré dans un commutateur et qui réaliserait les fonctions de sécurité usuellement confiées au *firewall*, à savoir :

- le contrôle d'accès,
- l'audit,
- l'authentification.

En plus de ces services de sécurité, ce module peut offrir la possibilité d'effectuer les services de confidentialité, intégrité/réauthentification et bourrage, prévention contre le rejeu de données, ce qui permet de sécuriser les échanges sur le réseau public.

Si on devait comparer ce modèle avec les modèles présentés dans l'état de l'art du chapitre précédent, il serait à rapprocher de la première solution pour ce qui concerne l'échange de paramètres de sécurité car de la même façon, toutes les informations de sécurité transitent via les messages de signalisation. Il présente donc les mêmes inconvénients que cette solution. Cependant il est à noter que notre modèle permet de traiter un plus grand nombre de services de sécurité que les modèles qui ont été proposés jusqu'à présent. En ce qui concerne le service de confidentialité des échanges, notre modèle prévoit de chiffrer les données juste

au dessus de la couche AAL contrairement aux modèles présentés à la section 3.2.2 qui chiffrent dans la couche AAL ou dans la couche ATM. D'autre part, par rapport au modèle plaçant le chiffrement au niveau de la couche ATM, notre modèle ne présente pas le problème de la manipulation des clefs car le chiffrement s'effectue sur des trames AAL. Par contre notre modèle n'assure pas la confidentialité des entêtes ajoutés par la couche AAL et le chiffrement proposé dans la couche ATM apparait plus performant du fait que pour ce modèle il est effectué par un équipement matériel spécialisé. Par rapport au modèle de Gong, notre modèle apparait indépendant de la couche AAL utilisée et plus facile à implémenter du fait que Gong place son chiffrement entre les couches SAR et CS qu'il est bien souvent difficile à distinguer dans les logiciels. Cependant il ne répond pas à certains points tels que le protocole d'échange de clés.

Pour réaliser un tel module, nous nous sommes inspirés du modèle IEEE 802.10 qui consiste à intercaler entre les couches MAC et LLC une couche Secure Data Exchange (SDE). Cette couche est chargée de sécuriser les échanges de données sur un réseau en offrant les services d'authentification, contrôle d'accès, confidentialité et intégrité.

Nous nous sommes également inspirés de l'architecture utilisée par Ioannidis et Blaze ([IB93]) pour implémenter le protocole de sécurité swiPe.

## 4.2 Spécifications du module de sécurité

Le module de sécurité est intégré dans le commutateur ATM privé d'un RLE. Si plusieurs commutateurs privés sont utilisés au sein d'un RLE, on le place sur le(s) commutateur(s) interfacé(s) au réseau public. Ce module de sécurité a pour rôle de sécuriser les échanges effectués par les utilisateurs du RLE au travers du réseau public. Le module offre plusieurs services de sécurité :

- ceux habituellement rendus par le *firewall*:
  - le contrôle d'accès : à l'ouverture de connexion, le module de sécurité peut interdire ou autoriser un utilisateur d'un RLE distant de communiquer avec un utilisateur local ou réciproquement ou bien il peut s'opposer à ouvrir des connexions suivant le type d'application demandé. En général, ce service est couplé au service d'authentification.
  - authentification/intégrité d'une partie du flux de signalisation : le module de sécurité peut garantir que l'utilisateur à l'extrémité de la connexion est bien celui qu'on croit. Ceci peut être réalisé par des protocoles dédiés et l'usage des procédés cryptographiques : algorithmes à clefs symétriques (DES) / asymétriques (RSA) ou autres telles que les méthodes zero knowledge.

- l'audit : le module de sécurité peut conserver en mémoire la trace de tous ceux qui ont tenté de se connecter au RLE. Ainsi, à partir de cet audit, il est possible de déceler les tentatives d'intrusion sur le domaine du RLE; il est également possible de retrouver l'utilisateur qui a réussi à déjouer la politique de sécurité du RLE.
- ceux offerts pour assurer la protection des données échangées :
  - la confidentialité : le module de sécurité placé sur le commutateur du RLE émetteur se charge de chiffrer les messages émis par une station de travail de son RLE. C'est donc le message chiffré qui est transmis sur le réseau public. La connexion est donc sécurisée. Sur le RLE récepteur, le module de sécurité doit déchiffrer le message et transmettre le message en clair au bon destinataire.
  - le bourrage : Afin de rendre difficile l'analyse de flux, le module de sécurité peut également effectuer du bourrage sur le réseau public. Cela consiste à émettre des cellules inutiles dans le flux de cellules utiles afin de faire croire à un éventuel attaquant (placé en écoute sur le réseau public) que les deux LANs échangent des données et ainsi de le tromper quant au rythme /à la quantité/ d'informations échangées entre les deux LANs.
  - l'intégrité : le module de sécurité peut assurer au destinataire des messages que les messages échangés n'ont subi aucune modification au cours de leur transfert sur le réseau public en intégrant dans le flux de données des contrôles d'intégrité.
  - l'authentification : en général, le service d'intégrité est réalisé à l'aide de procédés cryptographiques qui assurent en même temps l'authentification de l'émetteur des données utilisateur. Les risques d'un éventuel déguisement sont ainsi écartés.
  - la prévention contre le rejeu de données : ce service peut être combiné au service d'intégrité en y intégrant un numéro de séquence ou une estampille temporelle.

En plus de ces services, il est possible que le module de sécurité assure la confidentialité d'une partie du flux de signalisation.

La politique de sécurité du site dépend de nombreux paramètres tels l'utilisateur local, le destinataire à joindre, etc, ce qui permet de répondre à un très large éventail d'exigences des utilisateurs quant à la façon dont il souhaite protéger leurs communications. La politique de sécurité du site est gérée par l'officier de sécurité qui prend en compte les exigences en matière de sécurité de chacun des utilisateurs tout en se conformant à une politique de sécurité minimale qu'il s'est fixée pour assurer, indépendamment des utilisateurs, un niveau minimal de sécurité du site.

### 4.3 Modèle de sécurité proposé

Comme nous l'avons vu précédemment, les réseaux ATM ont la particularité de traiter distinctement la signalisation (ouverture / libération de connexion, contrôle, etc) et l'envoi de données utilisateur. Ces deux flux transitent sur le réseau sur deux canaux virtuels différents et au niveau des équipements du réseau, ils sont pris en charge par deux piles de traitement différentes appelées plan de contrôle et plan utilisateur.

Le plan de contrôle a pour rôle, entre autres, d'autoriser ou de refuser d'établir certaines connexions. D'ordinaire, ce choix dépend du contrat de trafic négocié auprès de l'opérateur et de la capacité du terminal local à pouvoir satisfaire les exigences de l'utilisateur qui sont précisées dans la demande d'ouverture de connexion : qualité de service, applications, etc; mais il devra aussi dépendre de choix faits en matière de sécurité. Il est tout à fait envisageable de placer dans le plan de contrôle, un module de sécurité chargé des services de sécurité habituellement rendus par le firewall : contrôle d'accès, authentification de l'utilisateur origine de l'ouverture de connexion, audit. En plus de ces services, il peut également rendre confidentielle une partie du flux de signalisation en le chiffrant. Ces services font appel à un service de gestion de clés et de politiques de sécurité similaire à celui du modèle de sécurité IEEE 802.10.

Le plan utilisateur s'occupe de l'envoi/réception de données utilisateur. Il est donc en mesure de protéger l'échange de données par les services de confidentialité, bourrage, intégrité/réauthentification, prévention contre le rejeu de données. De plus, l'ATM fonctionne en mode connecté. Afin d'empêcher qu'un utilisateur distant déjà connecté pour un service donné n'ait accès à d'autres services auxquels il n'a pas droit, il est nécessaire de mettre en place un service de contrôle d'accès dans le plan utilisateur.

La figure 4.1 présente le modèle de sécurité ATM proposé.

### 4.4 Intégration de la sécurité dans le plan de contrôle

La solution choisie est inspirée du protocole de sécurité swIPE de Ioannidis et Blaze visant à sécuriser le protocole IP.

Le plan de contrôle du module de sécurité réalise les services de sécurité suivants :

- contrôle d'accès,
- authentification et intégrité d'une partie du flux de signalisation,
- confidentialité d'une partie du flux de signalisation,
- prévention contre le rejeu,

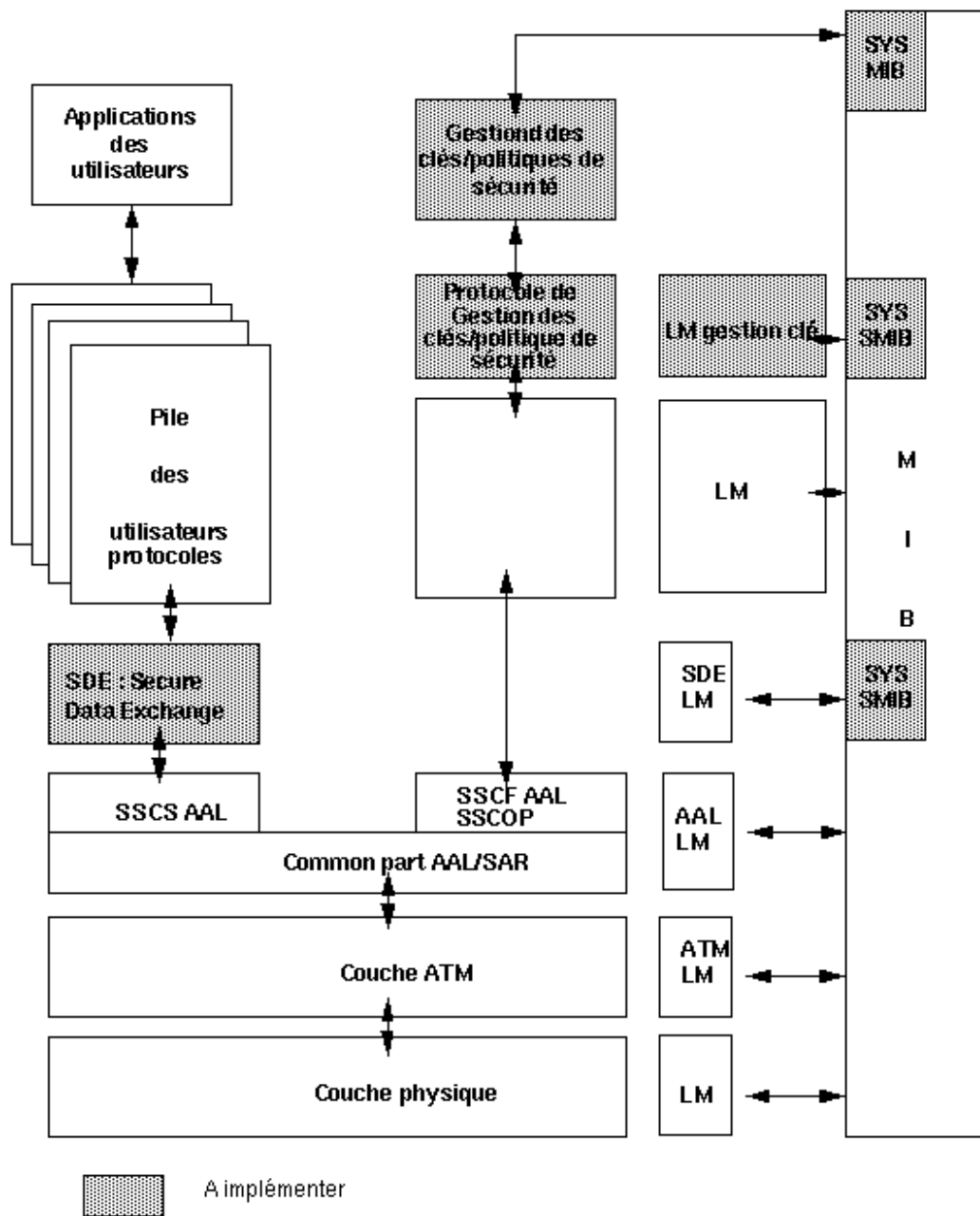


FIG. 4.1 - Le modèle de sécurité proposé

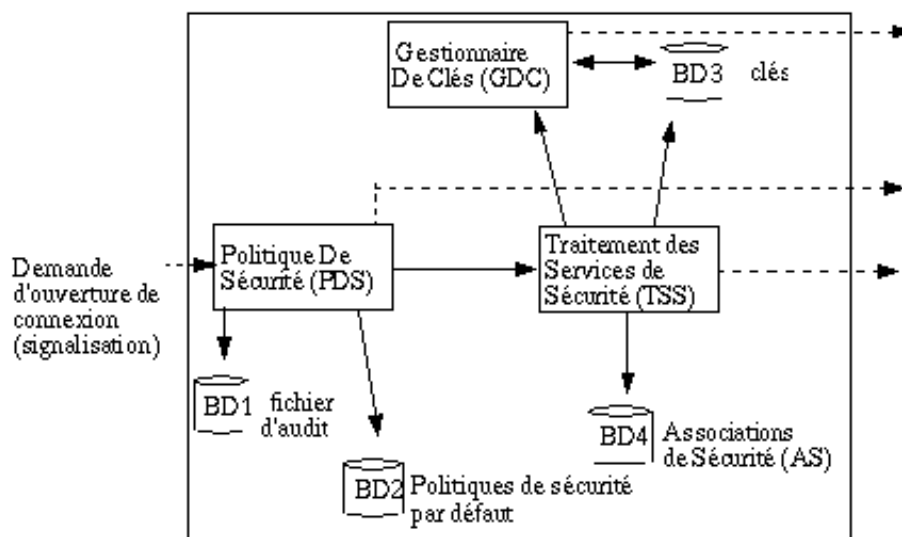


FIG. 4.2 - Composants de sécurité du plan contrôle

- audit.

Il met également en place les services de sécurité qui seront appliqués ultérieurement par le plan utilisateur (confidentialité, intégrité/réauthentification, bourrage). Cependant, pour que ces services soient exploitables, il faut en plus que le module de sécurité s'occupe de la gestion des clés, de la gestion de la politique de sécurité et des associations de sécurité.

L'architecture de la solution obtenue est composée de trois modules et de quatre bases de données. Elle est présentée par la figure 4.2.

#### 4.4.1 Les modules

Les trois modules sont :

- Le PDS ou Politique De Sécurité :

Il implémente la politique de sécurité du RLE spécifiée dans BD2. C'est à dire, à chaque fois qu'il émet un message de signalisation, en particulier les messages SETUP, RELEASE, CONNECT, il consulte BD2 et décide de l'exacte nature des opérations de de sécurité que le module de traitement TSS doit effectuer. De même quand il reçoit un message de signalisation, il consulte BD2 et détermine les services de sécurité que le TSS doit mettre en place. Ce module peut également procéder à de l'audit en enregistrant dans BD1 toutes les ouvertures de connexions avec un certain nombre de

précisions : les identités des utilisateurs en communication, l'application demandée, etc. D'autres informations telles que les coupures de connexions pourraient également être enregistrées.

- Le TSS ou Traitement des Services de Sécurité :  
A chaque fois qu'un message de signalisation est à émettre, il exécute les services de sécurité que lui a spécifiés le PDS pour une connexion donnée. Pour cela, il construit à partir des paramètres de sécurité des deux bases de données : BD3 pour les clés et BD4 pour les associations de sécurité une IE de sécurité. A chaque fois qu'il reçoit un message de signalisation, le TSS l'analyse et récupère l'IE de sécurité. Si les informations de sécurité ne sont pas correctes, le TSS libère la connexion. Si le message reçu est un SETUP, il retourne un CONNECT précisant les services de sécurité qu'il souhaiterait mettre en place sur cette connexion. Si le message reçu est en tout point correct, le TSS met à jour la base SDE qui enregistre ces services de sécurité. Le TSS est également chargé de mettre à jour BD4.
- Le GDC ou Gestionnaire De Clefs :  
Il gère la base de clefs BD3 utilisées par le module TSS pour authentifier les modules de sécurité (ou les utilisateurs), chiffrer ou vérifier l'intégrité des messages de signalisation et également par le SDE pour effectuer du chiffrement, vérifier l'intégrité/authentification des données. Il communique avec d'autres modules GDC localisés sur d'autres RLE pour prendre connaissance des clefs propres au site distant et établir des clefs de session.

#### 4.4.2 Les bases de données

Les quatre bases de données qui sont stockées dans la MIB sécurisée (SMIB) sont :

- BD1 ou le fichier d'audit.
- BD2 :  
Elle contient la politique de sécurité que le module TSS doit appliquer. Elle se résume à des associations du type : (services de sécurité, et optionnellement identificateur d'un utilisateur local, identificateur d'un module de sécurité distant, identificateur d'un utilisateur distant, application visée, etc). Seul l'officier de sécurité a le droit d'initialiser/modifier BD2.
- BD3 :  
BD3 contient les clés en cours d'utilisation : clés publiques/privés du module de sécurité, clés publiques des autres modules de sécurité, clés de session, la clé publique de l'autorité de certification qui lui a fourni un certificat d'authenticité de sa clé publique, etc. Pour des raisons évidentes de sécurité, BD3 ne peut être accédée en écriture/lecture que par le module GDC et qu'en lecture par le module TSS.

- BD4 :  
BD4 contient les Associations de Sécurité, c.à.d. les paramètres de sécurité relatifs aux utilisateurs locaux, nécessaires au chiffrement, à la vérification de l'authentification/intégrité et au contrôle d'accès. BD2 et BD4 diffèrent en ce que BD2 dicte quels sont les services de sécurité à mettre en place et BD4 précise comment faire: quels algorithmes utiliser, quels sont les utilisateurs autorisés à pénétrer dans le RLE et pour quelles applications leur accès est-il autorisé, etc.

### 4.4.3 L'IE de sécurité

Les modules de sécurité ont en général leur propre politique de sécurité. Dès l'ouverture de connexion, ils doivent donc se mettre d'accord sur les services de sécurité à mettre en oeuvre et sur les méthodes à utiliser. Pour cela, ils doivent inclure dans les messages de signalisation d'établissement d'une connexion SETUP et CONNECT, un élément d'information de sécurité décrit par la figure 4.3.

Une fois cette phase de négociation réalisée, les deux modules sont alors en mesure de s'échanger :

- des messages de signalisation de façon sécurisée ou pas. Tous les messages de signalisation RELEASE, STATUS, etc peuvent être authentifiés et en partie protégés en confidentialité, en intégrité grâce à l'élément d'information de sécurité qui peut être ajouté en plus de leurs éléments d'information ordinaires (called party number, calling party number, etc).
- des données utilisateur de façon sécurisée ou pas. Ces données peuvent être authentifiées, protégées en intégrité et en confidentialité et également protégées de toute analyse de flux grâce aux techniques de bourrage.

Détaillons cet élément d'information de sécurité dans l'ordre d'apparition des champs :

- Les trois premiers champs :  
Les trois premiers champs sont les trois champs classiques des éléments d'information. Le champ Security Identifier (1 octet) permet d'identifier l'élément de sécurité. Le champ IE instruction field (1 octets) sert à assurer la compatibilité de cet IE avec les autres. Enfin Le champ length (2 octets) donne la longueur de l'IE de sécurité.
- Le champ SAID (Security Association Identifier) :  
Les quatre premier champs sont toujours en clair, mais le reste du message peut être chiffré pour assurer la confidentialité de l'IE de sécurité. Le champ SAID alors renseigne sur le contexte de sécurité à utiliser entre les deux modules de sécurité. Typiquement ce contexte doit déjà avoir été négocié entre les deux modules de sécurité. Il peut préciser l'algorithme utilisé pour chiffrer le reste de l'IE de sécurité, la clé, mais aussi tous les

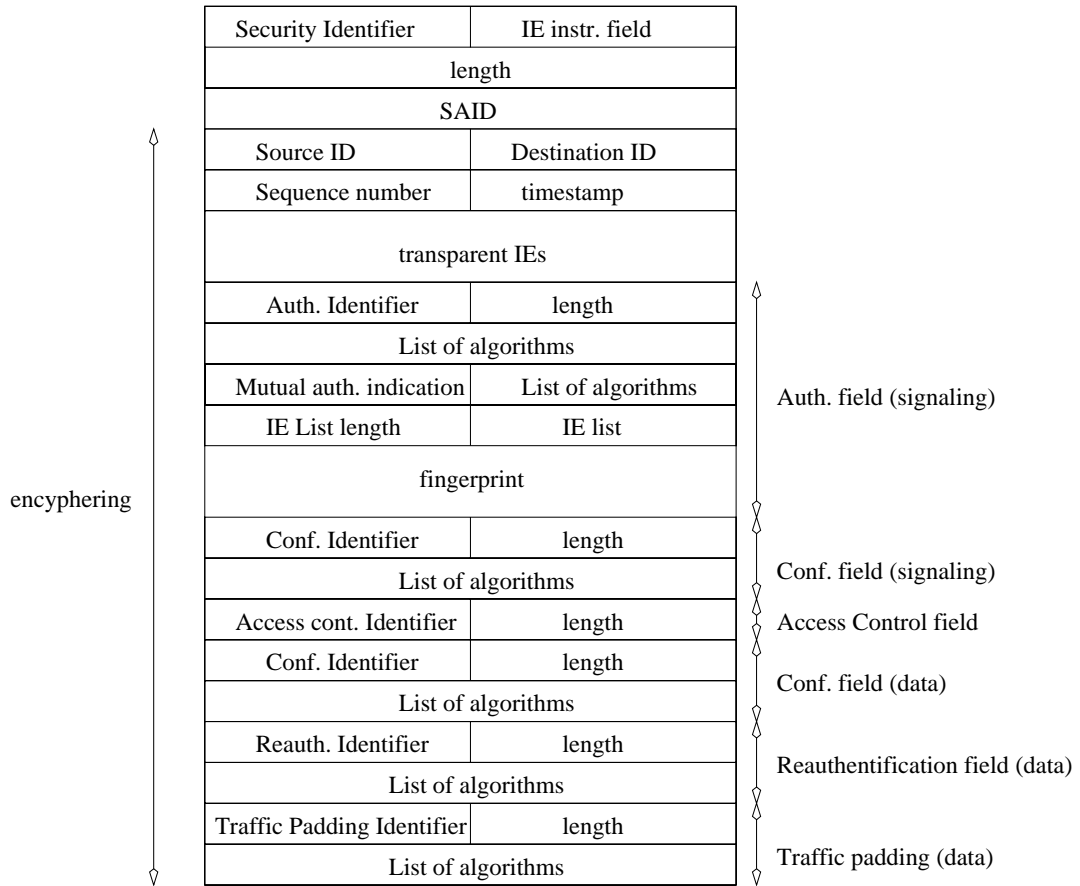


FIG. 4.3 - L'IE de sécurité

paramètres de sécurité tels : les algorithmes de chiffrement pour réaliser l'authentification, l'intégrité, la confidentialité, etc.

- Les champs Source Id et Destination Id :  
Ils contiennent des informations qui permettent d'identifier précisément l'émetteur/récepteur des messages. Il peut par exemple s'agir du nom des utilisateurs. Mais aussi il pourrait s'agir de l'adresse de leur terminal.
- Les champs Sequence number et Timestamp :  
Ces deux champs permettent de déjouer l'attaque qui consiste à rejouer un IE de sécurité. Le Timestamp indique la date et l'heure d'émission du message (relativement au méridien de Greenwich) et le Sequence Number, est un numéro de séquence qui permet de distinguer deux messages émis à la même heure.
- Le champ IEs transparent for the public network :  
Le message de signalisation comporte plusieurs types d'IEs. Si du chiffrement est effectué sur l'IE de sécurité, quelques IEs peuvent être déplacés dans ce champ et chiffrés, ce qui permet de garantir la confidentialité de ces IEs. Si de plus, l'IE de sécurité contient un élément d'authentification, ces IEs sont alors protégés en intégrité.
- Le champ Authentication/Integrity Element (signalling flow):  
Il donne des paramètres de sécurité relatifs à la façon de réaliser le service d'authentification et le service d'intégrité portant sur le message de signalisation. Voici les différents champs de cet élément :
  - Authentication/integrity identifier, (1 octet) un identificateur précisant qu'il s'agit du service d'authentification/intégrité du flux de signalisation.
  - Length, (2 octets) la longueur de cet élément.
  - List of algorithms: informations précisant dans l'ordre de préférence décroissant les algorithmes à utiliser pour effectuer l'authentification/intégrité, le premier spécifié étant celui utilisé pour la signature digitale. Chaque algorithme est représenté par un code de 1 octet. Parmi ces algorithmes, il serait possible d'utiliser le MD5 + RSA, le keyed MD5. Laisser trois choix d'algorithmes possibles semble être suffisant.
  - Mutual authentication indication: information précisant si une authentification mutuelle est exigée et si c'est le cas, le champ suivant List of algorithms donne dans l'ordre de préférence décroissant la liste des algorithmes de chiffrement que le module de sécurité récepteur doit utiliser pour réaliser son authentification.

- IEs list length et IEs list précisent le nombre d'IEs constituant la liste d'IEs et la liste des identificateurs d'IEs sur lesquels la signature digitale doit porter. Contrairement au champ IEs transparent for the public network, seul l'identificateur d'IE est placé dans le champ IEs list. L'IE est lui placé en dehors de l'IE de sécurité. La signature digitale calculée porte sur les IEs précisés dans ce champ dans l'ordre donné par la liste. Typiquement, les IEs apparaissant dans le champ IEs list sont des IEs dont on veut assurer l'intégrité, contrairement aux IEs inclus dans l'IEs transparent for the public network qui sont des IEs dont l'intégrité et la confidentialité sont assurés. A noter que les IEs du champ IEs list sont des IEs qui ne doivent pas être modifiés par le réseau public. Typiquement ces IEs sont les IEs des numéros appelant et appelé.
- Digital Signature : La signature digitale porte sur l'IE de sécurité et les IEs spécifiés dans le champ IEs list. L'algorithme utilisé est le premier spécifié dans le champ list of algorithms.
- Le champ Confidentiality Element (signalling flow):  
Il donne les paramètres de sécurité relatifs à la façon de réaliser le chiffrement de la confidentialité des messages de signalisation. Typiquement cet élément est placé dans l'IE de sécurité pour renégocier l'algorithme de chiffrement, pour initialiser un contexte de sécurité de numéro SAID relatif à la confidentialité du flux de signalisation. Voici les différents champs de cet élément :
  - les champ Confidentiality ID, Length : idem l'élément d'authentification/intégrité.
  - List of encryption algorithms to be used donne la liste dans l'ordre de préférence décroissant des algorithmes à utiliser pour effectuer le chiffrement de l'IE de sécurité, classiquement le DES. Un choix de trois algorithmes semble suffisant.
- Le champ Access control Element :  
Sa présence dans l'élément de sécurité indique que le module de sécurité homologue doit intégrer dans son message de signalisation les éléments d'information nécessaires pour caractériser l'application visée : AAL parameters, Broadband high layer information, Broadband low layer information, QoS parameter. Voici les différents champs de cet élément :
  - les champs Access control ID, Length : idem l'élément d'authentification/intégrité.
- Le champ Confidentiality Element (data flow):  
Il donne des paramètres de sécurité relatifs à la façon de réaliser le service

de confidentialité portant sur les données utilisateur. Voici les différents champs de cet élément :

- les champs Confidentiality ID, Length : idem l'élément d'authentification/intégrité.
  - List of encryption algorithms to be used donne la liste dans l'ordre de préférence décroissant des algorithmes de chiffrement à utiliser pour effectuer le chiffrement des données utilisateur (e.g. DES). Un choix de trois algorithmes semble suffisant.
- Le champ Integrity/Reauthentication Element (data flow):  
Il donne des paramètres de sécurité relatifs à la façon de réaliser le service de confidentialité portant sur les données utilisateur. Voici les différents champs de cet élément :
- les champs Integrity/reauthentication ID, Length : idem l'élément d'authentification/intégrité.
  - List of algorithms to be used donne la liste dans l'ordre de préférence décroissant des algorithmes à utiliser pour effectuer le chiffrement des données utilisateur (e.g. DES). Un choix de trois algorithmes semble suffisant.
- Le champ Traffic padding Element (data flow):  
Il donne des paramètres de sécurité relatifs à la façon de réaliser le service de bourrage portant sur les données utilisateur. Voici les différents champs de cet élément :
- les champs Traffic padding ID, Length : idem l'élément d'authentification/intégrité.
  - List of algorithms to be used donne la liste dans l'ordre de préférence décroissant des algorithmes à utiliser pour réaliser le bourrage. Un choix de trois algorithmes semble suffisant.

#### 4.4.4 Fonctionnement

Dans cette section, nous étudions les différents échanges entre modules de sécurité et la façon dont les messages de sécurité sont traités par les modules à l'ouverture de connexion.

Dans les explications qui suivent, nous supposons que les deux RLEs qui veulent communiquer disposent d'un module de sécurité. Rien n'empêche un RLE pourvu d'un module de sécurité et un RLE sans module de sécurité de communiquer. Il suffit pour cela que le module de sécurité n'exige aucun service de sécurité du RLE distant. S'il exige un minimum de sécurité, la connexion ne pourra pas se faire.

Supposons que les bases de données BD2 et BD4 soient correctement initialisées dans le module de sécurité du RLE émetteur et qu'un utilisateur A de ce RLE veuille ouvrir une connexion avec un utilisateur B d'un autre RLE.

– Du côté du RLE émetteur :

L'utilisateur A envoie à son commutateur local sa demande d'ouverture de connexion SETUP dans lequel sont précisés : son adresse, l'adresse du destinataire, l'identité de l'utilisateur origine A, l'identité de l'utilisateur destinataire B et optionnellement un authentificateur qui peut servir au module de sécurité local pour s'assurer de l'identité de l'utilisateur émetteur de la demande.

Le PDS du module de sécurité peut enregistrer dans son fichier d'audit BD1 la demande avec la liste d'informations (numéro de l'appelant, numéro de l'appelé, identificateur de l'utilisateur émetteur, identificateur de l'utilisateur destinataire, date et heure (horodateur), numéro de séquence, informations BHLI et BLLI, paramètres AAL). Le PDS consulte BD2. Suivant l'identité des utilisateurs émetteur et récepteur, leur localisation, etc, il détermine les services de sécurité à effectuer pour la connexion et il transmet ces informations à TSS.

Le TSS traite les différents services de sécurité demandés. S'il s'agit du contrôle d'accès, il consulte BD4 et en déduit si la connexion entre l'utilisateur local et l'utilisateur distant situé sur un certain RLE (numéro appelé) dans une certaine zone géographique est autorisée. Si cette connexion est interdite, il stoppe tout traitement et provoque l'envoi d'un message RELEASE à l'utilisateur émetteur en précisant la raison : "connexion interdite" par exemple. Sinon, il peut construire un élément de sécurité de contrôle d'accès qu'il insèrera dans l'IE de sécurité du message SETUP. Si les services de sécurité comprennent l'authentification/intégrité de la signalisation, la confidentialité de la signalisation, l'intégrité et/ou la confidentialité des données, il doit vérifier que BD3 contient les clés nécessaires à la connexion et que ces clés sont suffisamment récentes (durée de validité). Si BD3 ne contient pas les clés appropriées, le TSS avertit le module GDC qu'il faut qu'il récupère les clés de tel RLE et qu'il établisse des clés de session. Le GDC ouvre une connexion avec le GDC distant, prend connaissance des clés de l'autre RLE et établit des clés de session. Une fois que BD3 détient les bonnes clés, le TSS est en mesure de traiter les services de sécurité suivants :

– l'intégrité/authentification des données, confidentialité, bourrage : le TSS doit avertir son homologue que ces services là sont exigés pour l'échange de données utilisateur, en ajoutant des éléments de sécurité au SETUP. Il doit aussi conserver en mémoire (dans BD4 dans le champ associations des éléments de sécurité : confidentialité des données, authentification/intégrité des données, bourrage) que la VCC

utilisateur (identifiée par son numéro de référence) devra rendre ces services de sécurité. Ces services seront par la suite rendus par la couche SDE. Le TSS communique ces informations à la couche SDE par l'intermédiaire de la SMIB.

- l'authentification/intégrité de la signalisation : le TSS ajoute dans le message SET UP initial un élément d'authentification des éléments de sécurité correspondant aux services de sécurité.
- prévention contre le rejeu.

Le TSS envoie alors le message de SETUP enrichi par des éléments de sécurité au RLE destinataire.

- Du côté du RLE récepteur : Le PDS reçoit le SET UP en provenance du RLE émetteur. Il peut enregistrer cette demande d'ouverture de connexion dans son fichier BD1 d'audit. Après avoir examiné le message SETUP et consulté BD2, il transmet à TSS les services de sécurité à mettre en oeuvre : les services de sécurité attendus par le module de sécurité émetteur mais aussi les services de sécurité exigés par la politique de sécurité locale.

Si le contrôle d'accès fait partie de la politique de sécurité locale, le TSS consulte ses Associations de sécurité BD4 et refuse (envoi d'un message RELEASE au RLE émetteur) ou accepte la connexion. Si les services d'authentification, intégrité, confidentialité sont exigés, le TSS consulte sa base BD3 et, si elle n'est pas complète, exige de GDC qu'il obtienne les clés manquantes. Il vérifie alors grâce à BD3, l'authentification et conserve en mémoire que sur la connexion VCC utilisateur (numéros VPI/VCI spécifiés dans le SET UP) traitée par le plan utilisateur, les services de confidentialités, intégrité sont à mettre en place.

Une fois ces éventuels services d'authentification/contrôle d'accès effectués, le module de sécurité fait parvenir à l'utilisateur B la demande d'ouverture de connexion SET UP en prenant le soin de supprimer au préalable les éléments d'information de sécurité. L'utilisateur est en droit d'accepter ou de refuser la connexion.

S'il l'accepte, il retourne un message CONNECT au module de sécurité local, lequel le transmet au RLE qui, à son tour le fait suivre à l'utilisateur A.

Une fois que ce message est reçu par l'utilisateur A, un canal virtuel VCC spécialement réservé à l'échange de données entre les utilisateurs A et B est ouvert. Le plan de contrôle des modules peuvent alors communiquer au plan utilisateur et en particulier à la couche SDE l'ensemble des services de sécurité qu'ils doivent appliquer, ainsi que les clés à utiliser. Cette communication inter-plans s'effectue grâce à la SMIB.

## 4.5 Intégration de la sécurité dans le plan usager

Dans le plan utilisateur, il suffit d'intercaler une couche SDE similaire à la couche SDE de la norme 802.10 pour sécuriser l'échange de données utilisateur.

Comme expliqué ci-dessus, il faut qu'un module de sécurité pourvu d'une couche SDE soit en mesure de communiquer avec n'importe quel système non sécurisé. Il faut donc que cette couche soit optionnelle, c'est à dire qu'elle n'interdise pas la communication entre systèmes sécurisés et systèmes non sécurisés. C'est en fait à l'ouverture de connexion, que les sites RLEs doivent préciser s'ils disposent d'un module de sécurité. Si l'un d'entre eux n'en possède pas, une connexion ne peut s'établir que si les deux sites autorisent l'échange de données non sécurisé.

La couche SDE doit être capable d'effectuer les services de sécurité suivants :

- confidentialité,
- bourrage,
- intégrité/authentification,
- contrôle d'accès.

Parmi ces services de sécurité, seuls quelques uns sont appliqués. La sélection des services de sécurité se fait par le plan contrôle à l'ouverture de connexion. Le plan contrôle communique ensuite le choix fait en matière de sécurité au plan utilisateur, ou plus exactement au SDE par le biais de la SMIB. Le plan contrôle écrit dans la SMIB l'association (ATM SAP ou AAL SAP suivant le modèle de sécurité utilisé - services de sécurité avec quelques précisions telles que : l'algorithme de chiffrement utilisé, la clé, etc). Une fois la connexion VCC utilisateur ouverte, la couche SDE doit lire dans la SMIB les informations de sécurité relatives à sa connexion.

### 4.5.1 La norme IEEE 802.10

Dans cette section, nous présentons uniquement la partie de la norme IEEE 802.10 qui nous intéresse. Les parties gestion des clés et administration de la sécurité ne sont que très peu abordées.

La norme IEEE 802.10 permet d'introduire dans le modèle IEEE 802 différents services de sécurité - confidentialité des données, authentification, contrôle d'accès et intégrité des données - qui ont pour finalité de garantir des échanges de données sur le réseau sécurisés. La couche qui permet d'implanter ces services s'appelle couche SDE ou service d'échange de données en confiance et est placée entre les couches LLC et MAC du modèle OSI.

L'interface fournie au service LLC par SDE est identique à celle fournie par la couche MAC, c'est-à-dire qu'il s'agit d'un service à datagramme. LLC soumet

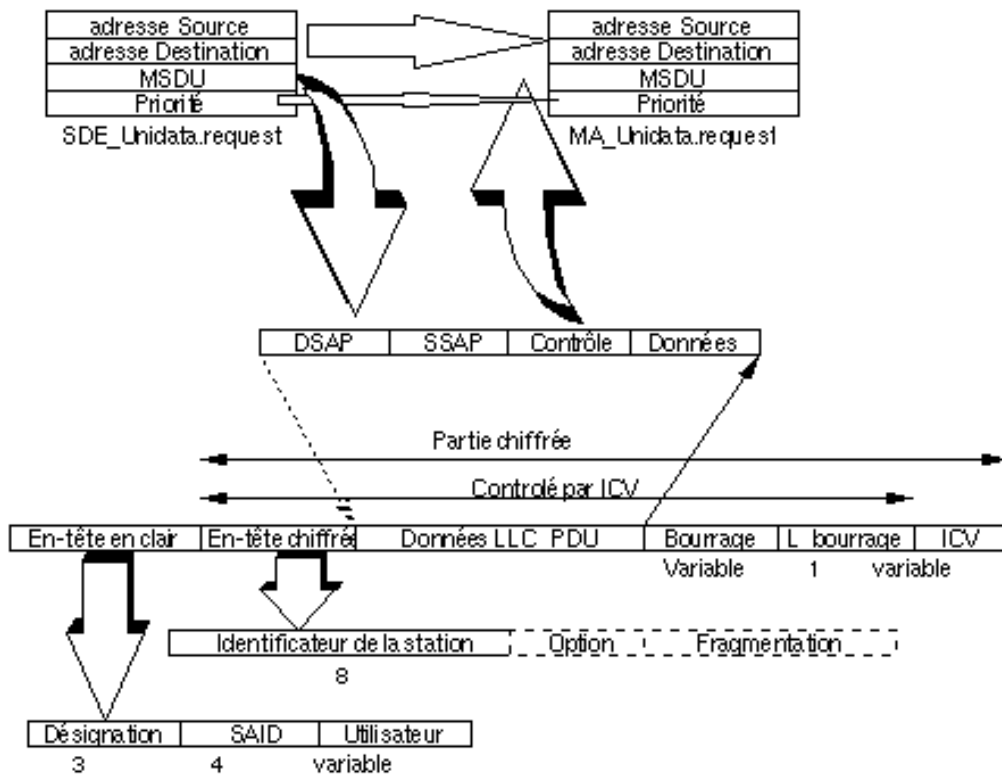


FIG. 4.4 - Construction des SDE-PDU dans IEEE 802.10

et reçoit des paquets indépendants les uns des autres. Cela permet de conserver une complète compatibilité avec les protocoles supérieurs déjà développés. Néanmoins comme il faut définir les paramètres de sécurité, un contexte de connexion est défini pour les échanges entre groupe de partenaires. On parlera d'association de chiffrement ou de sécurité.

Ces associations contiennent l'ensemble des paramètres de sécurité à utiliser pour chaque connexion (ex: les clés de chiffrement, services de sécurité, options, etc). Ces informations sont stockées dans une MIB locale appelée SMIB ou Security MIB. Elles sont prédéfinies et sélectionnées par les modules de gestion de clés et administration de la sécurité. Le premier est chargé de d'échanger et de maintenir à jour les clés. Le second permet de gérer - lire, mettre à jour, détruire - les informations de la SMIB.

#### 4.5.2 Format des trames IEEE 802.10

Les primitives fournies par la couche SDE sont identiques à celles définies pour la couche MAC. Comme le service offert est à datagramme, deux primitives seulement sont définies :

- SDE-UNIDATA request (adr. source, adr. destination, MA-SDU, priorité) pour émettre un bloc de données,
- SDE-UNITDATA indication (adr. source, adr. destination, MA-SDU, priorité) pour signaler l'arrivée d'une unité de donnée.

Les informations sont fournies en clair et seul le champ MA-SDU fera l'objet d'un chiffrement. La requête traverse la couche SDE, les paramètres adresse source, adresse destination et priorité ne sont pas affectés, tandis que le champ MA-SDU fait l'objet d'un traitement spécifique de la sécurité qui est précisé dans la SMIB. Le champ MA-SDU contient dans ses deux premiers octets les champs DSAP et SSAP. Le champ MA-SDU est intégré dans les données transmises. La couche SDE lui ajoute les enveloppes suivantes : en-tête en clair, en-tête protégé, champ bourrage et contrôle d'intégrité. La figure 4.4 présente cette construction.

- En-tête en clair :

Sa valeur est déterminée pendant la construction de l'association et reste constante pendant toute la durée de vie de celle-ci. Ce champ est optionnel. S'il n'est pas utilisé toutes les stations du réseau doivent disposer de la couche SDE. L'utilisation de ce champ permet de faire coexister des services non sécurisés et SDE sur le même réseau physique. Les trois premiers octets de ce champ sont appelés "designant SDE". Ils sont identiques à l'enveloppe LLC d'une trame non numérotée. Ils contiennent dans les deux premiers octets les LSAP source et destination définis dans le réseau comme étant le point d'entrée du module SDE. Le troisième octet contient le code UI, trame information non numérotée de la norme IEEE 802.2. Ainsi l'en-tête de cette trame est totalement conforme à la norme IEEE 802.2. En affectant des SAP spécifiques pour la couche SDE, on autorise le réseau à véhiculer des trames normales (non soumises à travers les fonctions de sécurité) et des trames sécurisées à circuler sur le réseau. Bien sûr les numéros de SAP choisis ne doivent pas entrer en conflit avec ceux déjà définis ou utilisés. Donc ce champ sert au démultiplexage des différentes trames vers le bon protocole. Les quatre octets suivants, appelés SAID, Security Association Identification, identifient l'association sécuritaire utilisée, SAID permet à la couche SDE de retrouver dans ses tables les paramètres utiles pour le chiffrement/déchiffrement de cette association et toute autre action spécifique. Un champ optionnel, défini par l'utilisateur, suit le champ SAID. Ce champ a une taille variable définie dans la base de donnée SMIB (la taille maximale est toutefois limitée à 20 octets). Ce champ n'a pas pour but de définir l'association mais il peut

permettre d'en préciser certains éléments. Il peut servir :

- de bourrage pour aligner les données sur une frontière de mots,
- de tampon contenant l'état du mécanisme de chiffrement,
- à identifier des attributs de l'association,
- à filtrer les trames dans un pont.

Seul l'en-tête en clair n'est pas soumis au mécanisme de chiffrement. Tous les champs suivants sont chiffrés. Cet en-tête ainsi d'ailleurs que l'enveloppe MAC sont susceptibles d'être modifiés par un intrus.

- **En-tête protégé** Il contient un champ unique appelé identificateur de la station (SID Station Identification) de 8 octets. Lorsque l'option de fragmentation est mise en oeuvre des champs complémentaires sont prévus. Le SID est conçu pour ranger un identificateur de la station, à charge pour les applications de SDE de déterminer comment utiliser ce champ. Cette adresse où l'identificateur est protégé par le mécanisme de chiffrement permet d'éviter le déguisement de la part d'un intrus qui chercherait à envoyer un message en lieu et place d'une autre station. En effet cet intrus ne pourra pas modifier correctement le champ SID tant qu'il n'aura pas cassé le chiffrement utilisé. Il est ainsi possible dans SDE de vérifier la cohérence entre le SID et l'adresse individuelle de l'origine du message de manière sûre. La nécessité de la fragmentation et du réassemblage apparaît dans SDE lorsque l'on utilise un mécanisme de chiffrement qui provoque une expansion de la taille de la MA-SDU.
- **Données :**  
Ce champ contient le SDU soumis après chiffrement. Un champ de bourrage est prévu à la fin des données pour aligner la taille de la trame physique sur une longueur adaptée au principe du chiffrement utilisé. Le champ bourrage est constitué d'un champ longueur qui permet de retirer le nombre d'octets de bourrage qui ont dû être ajoutés. Ce champ est nécessaire lorsque l'algorithme de chiffrement impose des blocs de taille fixe (cas du DES par exemple). Ce champ est optionnel. S'il est utilisé, le champ longueur du bourrage est toujours présent alors que la longueur du bourrage est variable et éventuellement nulle.
- **Contrôle d'intégrité, ICV (Integrity Control Value):**  
Le choix d'un mécanisme de contrôle (CRC, ...) est dépendant de l'association et donc défini pour chaque association dans la SMIB. La longueur de ce champ est variable. Mais dans une association donnée il est fixé. Comme les autres champs, IVC est optionnel.

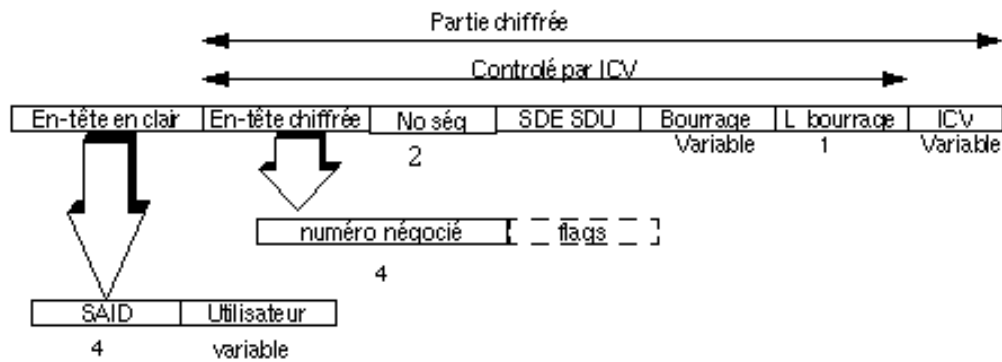


FIG. 4.5 - Construction des SDE-PDU dans notre modèle

### 4.5.3 Application de la norme IEEE 802.10 au modèle ATM

Il existe une différence capitale entre la norme IEEE 802.10 et le modèle de sécurité ATM. La norme IEEE 802.10 fonctionne dans un contexte de datagrammes tandis que le modèle ATM fonctionne en mode connecté. Du fait de cette différence, le traitement de la couche SDE a besoin d'être modifié et adapté au modèle ATM. Reprenons les différents champs introduits par la norme IEEE 802.10 ainsi que le montre la figure 4.5 :

- l'entête en clair. Il est optionnel et contient trois sous-champs : Désignation qui contient entre autres les LSAP source et de destination, SAID et un champ optionnel utilisateur. Grâce aux adresses LSAP, cet entête permet de repérer si un datagramme à traiter doit être pris en charge par une couche SDE et de l'orienter si nécessaire vers une couche SDE (adresse LSAP de l'entête). Dans le cas du modèle ATM, dès l'ouverture de connexion, les entités communicantes savent si les cellules qui seront échangées sur cette connexion font appel au service sécurisé. Pour le modèle ATM, il n'est donc pas nécessaire d'ajouter des informations équivalentes aux adresses LSAP de la norme IEEE 802.10. Par contre, le champ de type SAID peut être conservé. Il permet de récupérer les attributs de l'association de sécurité d'une connexion en interrogeant la SMIB. Cependant, ce champ est optionnel car d'autres éléments tels que l'ATM SAP ou AAL SAP peuvent également être utilisés dans ce but. Le champ optionnel utilisateur (jusqu'à 20 octets) peut également être conservé pour spécifier un certain nombre d'éléments propres à l'association sécuritaire (informations de synchronisation, vecteurs d'initialisation ...) ou pour introduire du bourrage.

- l'entête protégée (optionnelle) qui contient un sous-champ identificateur de la station (adresse). Il faudrait également introduire dans le modèle ATM un moyen d'authentification de l'origine du message tel que le SID de la norme IEEE 802.10 afin de se prémunir contre tout déguisement. Deux solutions sont envisageables :
  - les deux entités communicantes négocient à l'ouverture de connexion un numéro (e.g. 4 octets) qu'elles devront conserver pour toute la durée de la connexion et introduire dans l'entête protégée de chacun de leurs messages.
  - les deux entités communicantes signent grâce au RSA la partie chiffrée de chacun des messages. Ainsi, cette signature rendrait deux services de sécurité à la fois : intégrité et authentification.
- les données et le champ de bourrage sont conservés.
- le champ ICV est conservé.

Il est possible d'ajouter aux services de confidentialité, intégrité, authentification, contrôle d'accès prévus par la norme IEEE 802.10, le service de confidentialité du flux de données (appelé bourrage). Ce service peut être traité en ajoutant un champ optionnel à l'entête protégé. Comme un seul bit suffit pour indiquer si le champ SDE SDU contient du bourrage, il peut par exemple être intégré dans un champ flags de l'entête protégé. Dans tous les cas, ce champ doit obligatoirement figurer dans l'entête protégé car, il ne faut pas oublier que le but du service de bourrage est d'empêcher qu'un individu posté en écoute sur le réseau puisse dissocier les cellules de bourrage des cellules d'informations utilisateur. D'autre part, il est aussi nécessaire que le champ SDE SDU qui sert au bourrage ne soit jamais le même, une série de 0 par exemple. Il pourrait s'agir d'un compteur. En effet, si ce champ restait invariant, toutes les cellules de bourrage seraient identiques et le service de confidentialité du flux de données ne serait pas assuré.

# Chapitre 5

## Mise en œuvre

Dans ce chapitre nous présentons rapidement une mise en œuvre du modèle présenté au chapitre précédent. Pour des informations plus précises, le lecteur se reportera à l'annexe en chapitre 7.

### 5.1 Présentation du driver SUN

#### 5.1.1 Présentation des STREAMS

Avant de présenter le driver SUN nous présentons le concept des STREAMS qui est utilisé par celui-ci.

Le concept des STREAMS est un outil de programmation particulièrement bien adapté à la programmation des protocoles.

Un stream est une voie de communication entre l'espace utilisateur et le noyau. Il peut être composé de trois types d'éléments :

- Une tête de stream qui traduit les primitives du programme utilisateur en messages.
- Zéro, un ou plusieurs modules qui effectuent des actions sur les messages échangés.
- Un driver. Celui-ci pouvant être matériel ou logiciel.

Ces éléments communiquent entre eux par des messages. Ces messages sont formés de deux parties :

- Une partie contrôle qui détermine les actions à effectuer par les modules.
- Une partie données.

Ces messages passent d'un élément à l'autre à travers des files qui relient les éléments adjacents. A chaque file sont associées des primitives qui permettent le traitement des messages.

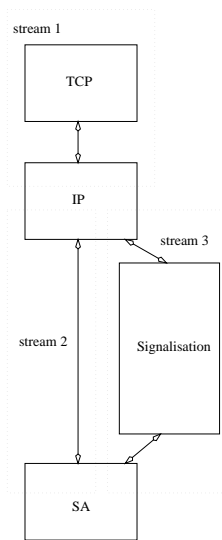


FIG. 5.1 - Structure générale du driver SUN

### 5.1.2 Fonctionnement de TCP/IP

Les implémentations des protocoles TCP et IP sont complexes. Nous ne décrirons donc que les points qui nous semblent primordiaux.

#### Le driver TCP

Le Driver TCP implémente le protocole TCP. Celui-ci est un protocole fiable fonctionnant sur connexion. Pour cela il utilise le concept des STREAMS; il reçoit des messages en provenance des TLIs, effectue les traitements relatifs au protocole puis construit un message contenant le paquet IP et le passe au driver IP.

Il existe une instance de TCP par connexion. Les paramètres de cette connexion sont gérés au moyen d'une structure de donnée associée à cette instance.

#### Le driver IP

Le Driver IP implémente le protocole IP. Celui-ci est un protocole non fiable fonctionnant en mode non connecté.

Le driver utilise également le concept des STREAMS. Cependant contrairement à TCP il joue un rôle de multiplexeur puisqu'il gère trois types de streams différents comme le montre la figure 5.1 :

- Le stream 2 entre IP et le driver SA de la carte ATM correspond au plan usager dans le modèle ATM.

- Le stream 3 entre IP, la signalisation et SA qui correspond au plan de contrôle du modèle ATM.
- Le stream 1 entre TCP et IP. Il y a en fait un stream de ce type par connexion.

Le paquet IP est dirigé vers la bonne instance du driver TCP au moyen du couple (adresses IP,numéros de ports).

### 5.1.3 La signalisation

la signalisation comprend quatres composantes. Le but de la signalisation est double :

- Faire le lien entre l'adresse IP gérée au niveau du driver IP et l'adresse ATM.
- Etablir une connexion et renvoyer les identificateurs de la connexion ouverte VCI/VPI à IP comme s'il s'agissait d'une adresse MAC.

Les quatres composantes sont les suivantes :

- Le module ATMIP est utilisé pour piloter le driver AAR.
- Le driver AAR :
  - Gère la résolution d'adresse ATM/IP.
  - Ouvre la connexion avec son entité homologue en envoyant des messages au driver Q93B.
  - Renvoie les identificateurs de connexion à IP par l'intermédiaire de ATMIP.
- Le driver Q93B :
  - Gère l'état des connexions.
  - Vérifie le contenu des messages.
- Le module SSCOP est une implémentation du protocole SSCOP dont nous avons parlé au chapitre 2.

## 5.2 Implémentation

L'implémentation s'est traduite par un certain nombre de modifications et de suppressions. Celles-ci ont été soit réalisées par nous, soit réalisées dans le produit TCP sur ATM duquel nous nous sommes inspirés.

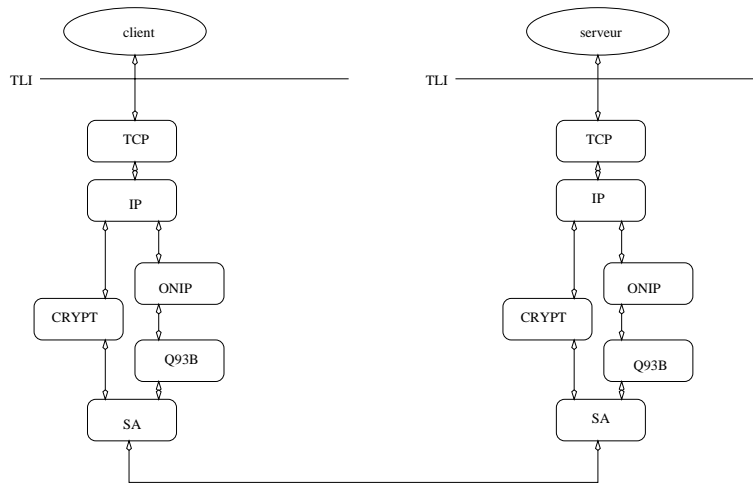


FIG. 5.2 - *Modèle d'une implémentation au niveau noyau*

### 5.2.1 Modifications réalisées dans TCP sur ATM

#### Suppression de AAR et de ATMIP

Comme on peut le voir sur la figure 5.2, ces deux composantes ont été supprimées car cette partie du driver est inadaptée. En effet lors de l'établissement d'une connexion TCP le driver AAR vérifie qu'une connexion entre les deux machines n'existe pas déjà afin de l'utiliser. Ceci va à l'encontre de notre modèle qui requiert qu'une connexion ATM soit ouverte pour chaque connexion TCP.

#### Modification des programmes utilisateurs

Le serveur de nom DNS ainsi que ses fichiers de configuration ont été modifiés afin de traiter les adresses ATM. Une bibliothèque permettant d'utiliser ce serveur a également été créée.

#### Création du driver ONIP

Le driver ONIP est un driver qui est inséré entre IP et Q93B. Il assure les fonctions suivantes :

- Il utilise la signalisation pour gérer les connexions.
- Il renvoie les VCIs/VPIs correspondants à la connexion.

### **Création de programmes de configuration de ONIP**

Afin de pouvoir utiliser ONIP deux programmes ont été créés :

- Laneplumb qui permet de placer ONIP dans le stream de signalisation.
- Laneunplumb qui permet d'enlever ONIP du stream de signalisation.

### **Modification du driver TCP**

Les modifications du driver TCP sont les suivantes :

- Gestion de la connexion ATM sous-jacente.
- Comme nous l'avons dit précédemment, le paquet IP est construit en partie dans le driver TCP. Le contenu des champs a donc été modifié afin d'y ajouter des informations telles que le VCI du côté du récepteur.
- Le driver a été également modifié afin qu'il puisse, côté récepteur, récupérer les informations passées par l'émetteur.
- Enfin le VCI sur lequel émettre est ajouté devant le paquet IP.

### **Modification du driver IP**

Les modifications du driver IP sont les suivantes :

- Suppression d'un certain nombre de fonctions déjà réalisées au niveau de l'AAL5 :
  - Suppression des fonctions de segmentation et de réassemblage.
  - Suppression des fonctions de calcul d'intégrité.
- Afin de diriger le paquet IP vers l'instance du driver TCP correspondante on utilise les ports TCP ainsi que le VCI.
- On utilise le VCI passé devant le paquet provenant de TCP afin de l'émettre sur la bonne connexion ATM.

### **5.2.2 Modifications touchant à la sécurité**

Ce sont celles que nous avons effectuées.

## Modification de la signalisation

Cette modification s'est traduite par quatre opérations :

- Adaptation des fonctions cryptographiques DES, RSA, MD5 et KeyedMD5 au noyau.
- Création de fonctions permettant d'ajouter et de lire l'IE de sécurité dont nous avons parlé précédemment. Cet IE est construit en utilisant les fonctions cryptographiques adaptées et est ajouté à la suite des IE déjà présents dans les messages CONNECT et SETUP.
- Création d'une fonction de négociation des paramètres de la politique de sécurité. Cette fonction prend en paramètres la politique de sécurité locale et la politique de sécurité désirée par le partenaire et renvoie une politique de sécurité acceptable pour les deux partis.
- Nous avons dit précédemment que le driver Q93B vérifiait le contenu des messages qui lui sont envoyés. Nous avons donc modifié le driver Q93B afin qu'il laisse passer les messages contenant l'IE de sécurité.

## Modification des programmes utilisateurs

Afin de faire le lien entre connexion TCP, connexion ATM et politique de sécurité on utilise un jeton. Le passage du jeton entre le milieu utilisateur et le driver TCP se fait au moyen d'une structure de donnée allouée dynamiquement. Nous avons modifié cette structure afin de pouvoir y passer des paramètres supplémentaires.

## Modification du driver ONIP

Nous avons modifiés ONIP afin qu'il négocie les paramètres de la politique de sécurité.

## Création de programmes de configuration de ONIP

Afin de pouvoir configurer ONIP nous avons créé *secplumb* qui permet de mettre à jour la politique de sécurité relative à un jeton dans ONIP.

## Modification du driver TCP

Nous avons modifié TCP afin qu'il prenne en compte le jeton passé par le programme utilisateur.

### 5.3 Utilisation de La politique de sécurité

Il est possible d'appliquer sur les données utilisateur la politique négociée dans ONIP de deux façon différentes :

- En utilisant un module cryptographique entre IP et SA qui communique avec ONIP par l'intermédiaire de IP.
- En utilisant une carte pilotée par ONIP assurant le chiffrement et la signature.



## Chapitre 6

# Conclusion

Le tableau 6.1 fournit une comparaison entre différentes solutions présentées dans l'état de l'art et notre modèle. Le résultat de cette comparaison est que notre modèle fournit des services supplémentaires par rapport aux autres solutions (Confidentialité de la signalisation, bourrage au niveau des données utilisateur). De plus notre modèle ne nécessite pas la modification des couches AAL ou ATM contrairement aux autres solutions proposées.

Afin de conclure notre rapport, nous proposons une comparaison entre le modèle développé par Maryline Laurent et notre maquette. Le tableau 6.2 montre que les différences au niveau des services rendus sont faibles voire nulles, ce qui prouve la validité du modèle.

Sur le plan personnel, ce stage m'a permis de approfondir mes connaissances dans les domaines de la sécurité, d'ATM ainsi que des systèmes UNIX en général et de SOLARIS en particulier. Ce stage m'a également permis de découvrir le monde de la recherche tout en travaillant sur un sujet à la fois intéressant et d'actualité.

Services	Objet	
	autres solutions	notre modèle
services portant sur la signalisation	A, PR	A, PR, I, C
services fournis par la signalisation	A, NS, EC	A, NS
services fournis par le flux de gestion	A, NS, EC	NON
services fournis par un canal auxiliaire	A, NS, EC	NON
services portant sur les données	RA, I, C	RA, I, PR, C, B
Chiffrement au niveau de	ATM, AAL	SDE

- A : Authentification.
- RA : Reauthentification.
- PR : Protection contre le rejeu.
- I : Intégrité.
- C : Confidentialité.
- EC : Echange de clefs.
- B : Bourrage.
- NS : Négociation des services de sécurité.
- NON : Ce service n'est pas rendu.

TAB. 6.1 - *Comparaison des services offerts par le modèle de M.L. et les autres solutions*

Services	Objet	
	implémentation	modèle
confidentialité de la signalisation	OUI	OUI
authentification de la signalisation	OUI	OUI
intégrité de la signalisation	OUI	OUI
confidentialité du flux de données	POSSIBLE	OUI
authentification du flux de données	POSSIBLE	OUI
intégrité du flux de données, protection contre le rejeu	POSSIBLE	OUI
bouffrage dans le flux de données	POSSIBLE	OUI

TAB. 6.2 - *Comparaison des services offerts entre le modèle de M.L. et notre implémentation*



# Chapitre 7

## Annexe

Dans ce chapitre nous présenterons l'implémentation d'une maquette de la solution proposée au chapitre quatre. Pour cela nous présenterons d'abord de manière générale l'environnement dans lequel s'est effectué cette implémentation. Nous décrirons ensuite de manière plus approfondie les parties dont nous nous sommes servis et que nous avons modifiées. Nous décrirons ensuite la solution proposée et les modifications qu'elle a entraînée.

### 7.1 Présentation générale du driver SUN

Dans tout ce qui suit nous désignerons par "package ATM" l'ensemble des composants fournis par SUN permettant la gestion d'une communication entre deux entités reliées par un réseau de type ATM.

#### 7.1.1 Architecture des drivers dans Solaris

Le package ATM fourni par SUN comme de nombreux drivers relatifs aux services de communication présents dans solaris utilise le concept des STREAMS pour sa mise en œuvre. Ce concept est en fait un ensemble d'outils permettant le développement de services de communication sous les systèmes UNIX à partir du système V.3. Un stream est une voie de communication bidirectionnelle pouvant exister entre des drivers dans le noyau, entre des processus de l'espace utilisateur ou entre un driver et un processus. Afin de simplifier notre exposé et parce que ces cas ne présentent pas de différences fondamentales, nous ne considérerons que le dernier cas. Comme le montre la figure 7.1 ce stream est construit à partir d'un ensemble d'éléments :

- Une tête de stream communiquant avec le processus utilisateur.
- Un driver (que ce soit le driver relatif à une carte physique ou un pseudo-driver, c'est à dire un driver logiciel).

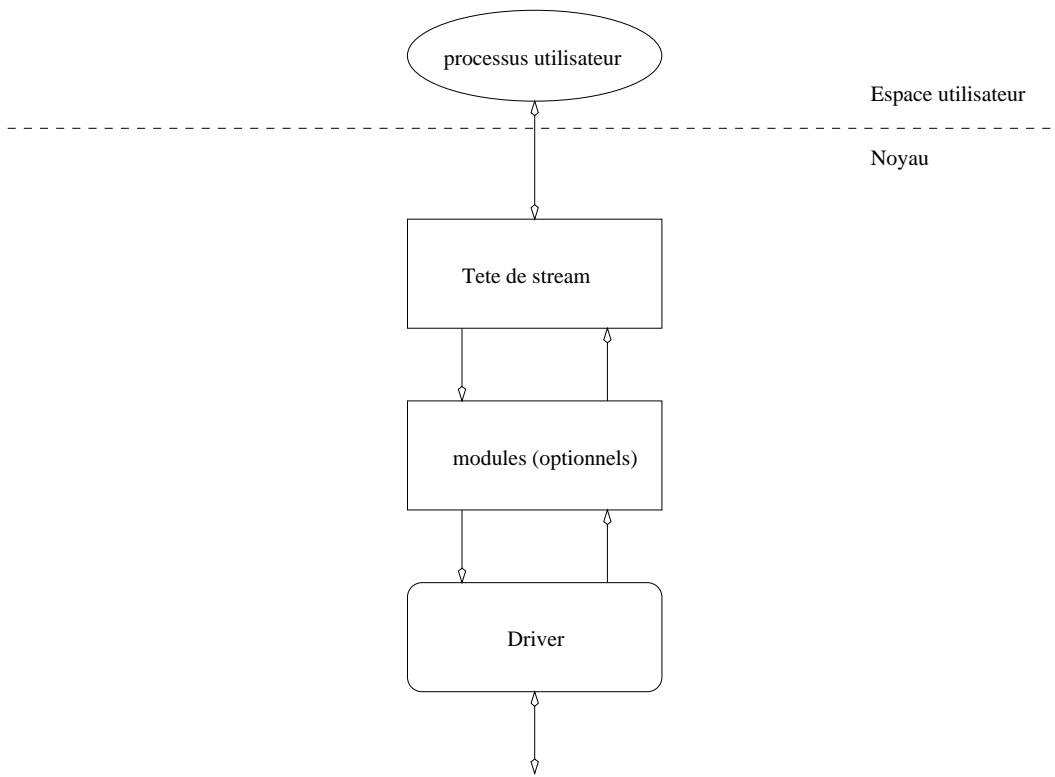


FIG. 7.1 - *Structure générale d'un Stream*

- Zéro, un ou plusieurs modules entre la tête de stream et le driver. Un module est composé de fonctions et de structures de données permettant de traiter les informations échangées entre le driver et le processus utilisateur.

La communication entre les différents éléments se fait au moyen de primitives spécifiques. A ces primitives on associe des données structurées sous forme de messages. Le passage des messages d'un module à ses modules adjacents se fait au moyen de deux files associées à chaque module. Une des files (Write Queue) est utilisée pour stocker les messages à destination du driver, l'autre (Read Queue) ayant un rôle similaire mais dans le sens inverse.

Les messages sont composés de deux parties :

- la première contient les données du message.
- la seconde contient des informations de contrôle comme le type du message ou sa priorité. Le placement des messages dans les files ainsi que les actions effectuées à la réception du message dépendent de cette seconde partie.

Les messages sont traités dans chaque module par des primitives associées à la file dans laquelle ils se trouvent. Le nombre de ces primitives varie suivant que le module est ou non un multiplexeur.

On peut distinguer deux types de primitives, les primitives de réception des messages et les primitives de traitement de ceux-ci (primitives de service). On effectue cette distinction afin de pouvoir traiter certaines informations de contrôle relatives aux messages (comme par exemple la priorité).

Bien que leurs noms soient totalement paramétrables, on choisit généralement des noms de type xxx\_??put pour les premières et xxx\_??srv pour les secondes. On distingue les primitives relatives à chaque file en introduisant un r (read) ou un w (write) à la place du deuxième ?, le premier étant utilisé dans le cas des modules multiplexeurs. Enfin xxx représente souvent le nom du module.

Le paramétrage s'effectue aux moyens de structures de données qui définissent le lien entre les fonctions à réaliser et le nom des primitives utilisées pour cela. Contrairement aux structures de données, les primitives peuvent ne pas exister.

Dans la pratique la distinction entre les deux catégories de primitives n'est pas aussi claire et il est fréquent de trouver des modules ne possédant pas de primitives de service, les actions relatives aux messages étant réalisées dans la primitive de réception.

Le stream est créé au moment de l'ouverture du driver et détruit au moment de sa fermeture. Les modules internes au stream sont gérés sous forme de pile LIFO.

Le concept des STREAMS est particulièrement intéressants car il apporte un certain nombre d'avantages dans la conception des drivers :

- Indépendance vis à vis de l'architecture et donc portabilité accrue.

- Réutilisabilité.
- Simplification de la mise en œuvre et du test.

Les ouvrages [Pad93],[Sun94] fournissent des informations complémentaires concernant la gestion, la construction et l'utilisation des STREAMS.

### 7.1.2 Description du package ATM

[Sun95] présente de manière succincte l'architecture du package ATM de Sun et en distingue deux parties :

- Une carte :  
Celle-ci met en œuvre les protocoles jusqu'au niveau de l'AAL5. Plusieurs types de cartes existent suivant le support physique que l'on désire utiliser (fibre optique, paire torsadée...).
- Un ensemble logiciel :  
Comme nous l'avons dit ci-dessus, les drivers de SUN utilisent le concept des STREAMS. De ce fait on peut classifier les composants logiciels suivant qu'ils sont des modules ou des drivers. En plus de ceux-ci le package ATM utilise des démons dans le milieu utilisateur. Cependant nous ne décrirons pas tous les modules ou drivers présents dans le package fourni par SUN, en effet certains de ceux-ci sont réalisés uniquement à des fins de tests et d'autres ne nous ont pas été utiles.

#### Les modules

ils sont au nombre de deux :

- Le module SSCOP :  
l'AAL5 sur lequel se base le package ATM ne garantit pas l'absence de perte de messages, aussi un protocole sur connexion, dit assuré, SSCOP est utilisé. Le module SSCOP qui est la mise en œuvre de ce protocole est inséré entre les drivers SA et Q93B afin de garantir la non perte de messages dans la signalisation.
- Le module ATMIP :  
Le modules ATMIP attend les demandes de résolution d'adresse de la part de IP. Lorsqu'il en reçoit une, il pilote le driver AAR afin d'obtenir l'adresse ATM et le VC correspondant. Lorsque ceux-ci sont obtenus il renvoie à IP le VC sous forme d'un message. En cas de fermeture de connexion, il avertit également IP afin que celui-ci mette à jour sa table des VCs.

## Les drivers

il y en a trois :

- Le driver SA :  
Il se situe le plus près de la carte ATM. Il reçoit des messages de SSCOP et d'IP. Ces messages peuvent être soit des messages de données en provenance d'IP, soit des messages de contrôle provenant de SSCOP permettant de gérer la répartition de la bande passante et les VCs.
- Le driver Q93B :  
C'est le driver gérant la signalisation. Il reçoit des messages du driver AAR et en fonction de ceux-ci gère l'état des connexions pour chaque communication. Nous décrirons plus précisément son fonctionnement dans la suite de notre rapport.
- Le driver AAR :  
Il est chargé de gérer la résolution d'adresse entre adresses ATM et IP, d'ouvrir une connexion avec son entité homologue en utilisant la signalisation puis de renvoyer à ATMIP le VC correspondant à la connexion établie afin qu'IP puisse envoyer les données directement à SA.
- Le driver IP :  
En plus de ses fonctions propres, il gère les différents types de supports disponibles (ethernet et atm dans notre cas).

## Les démons

- ilmid :  
Le démon ilmid met en œuvre le protocole ILMI défini dans [For94] afin :
  - D'obtenir le préfixe relatif au commutateur ATM pour construire l'adresse de chaque interface.
  - D'enregistrer la partie locale de l'adresse de la machine auprès du commutateur ATM.
- aarpd :  
Le démon aarpd se conforme au RFC 1577 afin d'effectuer la résolution entre adresses ATM et adresses IP. Il peut fonctionner en deux modes :
  - Serveur :  
Il attend les requêtes ATM ARP des clients et y répond en renvoyant soit l'adresse ATM correspondant à l'adresse IP reçue s'il la connaît soit en renvoyant un non acquittement (NAK) dans le cas contraire. Ces adresses IP sont soit directement récupérées dans le fichier */etc/aarconfig* , soit obtenue grâce à une requête "reverse ATM ARP" renvoyée au client lorsque celui-ci fait une requête.

- Client:

Dans le cas où le fichier */etc/aarconfig* ne suffit pas à trouver l'adresse ATM correspondant à l'adresse IP demandée, le client effectue une requête ATM ARP auprès de son serveur ATM ARP et répond aux requêtes "reverse ATM ARP" du serveur.

### 7.1.3 Outils de test

Afin de tester les drivers et modules précédemment décrits, SUN fournit un ensemble d'outils de tests permettant de vérifier leur bon fonctionnement. Deux nous semblent particulièrement intéressants pour des raisons diverses :

- Le premier, *tstqcc* (répertoire *drv/qcc/*) parce qu'il permet de tester le fonctionnement de la signalisation. Pour cela il est utilisé en mode serveur sur une machine et en mode client sur une autre. Afin de simuler l'établissement puis la fermeture d'une connexion, le client et le serveur procèdent à un échange de messages de signalisation (SETUP,CONNECT,CALL PROCEEDING,RELEASE,RELEASE COMPLETE).

En le modifiant nous avons créé un serveur *qccserv* et un client *qcc\_tcpdns* communiquant au travers de la signalisation afin de fixer les paramètres de sécurité dans chaque entité.

- Le second, *atmsnoop* (répertoire *drv/snoop/*) permet de comprendre et de vérifier le fonctionnement de toute la pile des drivers en récupérant tout le trafic passant par une interface. Il est possible en lui passant des paramètres appropriés d'isoler une partie de ce trafic (comme par exemple le trafic provenant d'ILMI) et de faire des recherches dans celui-ci au moyen de la description des chaînes de caractères désirées.

## 7.2 Structure et Implémentation

Avant de présenter les modifications que nous avons effectuées, nous présenterons de manière plus approfondie le fonctionnement de la signalisation dans le package ATM de SUN ainsi que les modules hors du package ATM que nous avons été amenés à modifier par la suite.

### 7.2.1 Fonctionnement de la signalisation

On peut répartir la signalisation sur deux drivers : le driver AAR et le driver Q93B. Ces deux drivers construisent et récupèrent les informations des messages de signalisation. Ceci est fait au moyen de la bibliothèque *libatm.a* qui est construite autour de *cc.c*, *qcc\_k.c*, *qcc\_u.c* dans *drv/qcc/*.

*cc.c* contient les fonctions qui permettent de construire et de lire les messages ainsi que les IEs qu'ils contiennent. Les fonctions concernant les messages

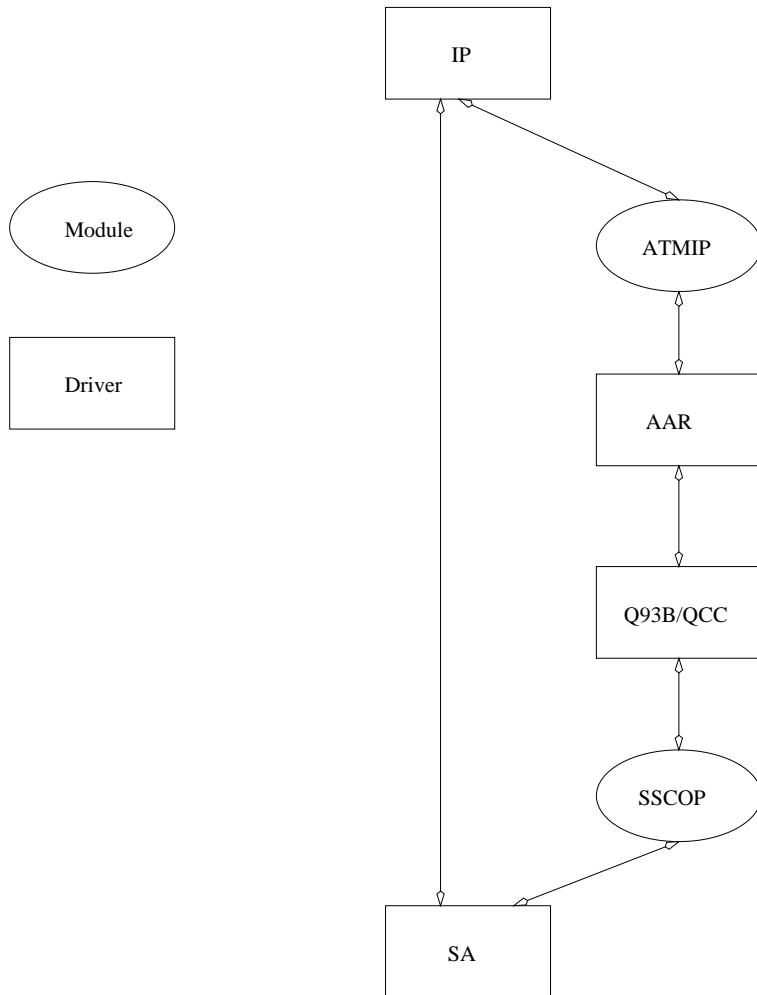


FIG. 7.2 - structure du package ATM de SUN

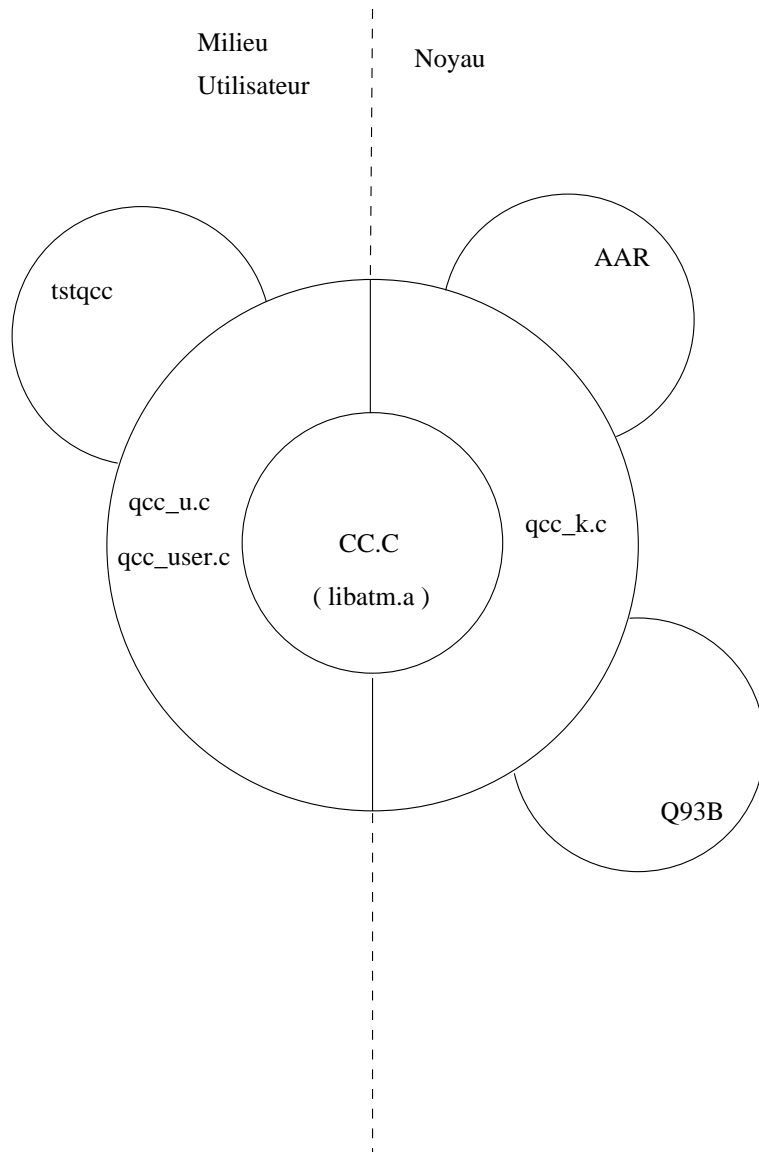


FIG. 7.3 - Relations entre les fichiers servant à construire les messages de signalisation

Messages	Drivers	
	Q93B	AAR
SETUP	PARSE	PARSE/BLD
CONNECT	PARSE	PARSE/BLD
CALL PROCEEDING	PARSE	PARSE/BLD
RELEASE		PARSE/BLD
STATUS	PARSE	
RESTART	PARSE	
ADD PARTY	PARSE	
ADD PARTY REJECT	BLD	
DROP PARTY	BLD	
DROP PARTY	BLD	
ACKNOWLEDGE		

TAB. 7.1 - Utilisation des fonctions de construction et de lecture des messages de signalisation dans les drivers AAR et Q93B

sont de la forme `cc_(parse|bld)*`, \* représentant le type du message à lire ou à construire. Les fonctions concernant les IEs sont de la forme `cc_(get|add)*`, où \* représente le type d'IE à récupérer ou à ajouter dans un message.

`qcc.k.c`, `qcc.u.c` et `qcc.user.c` contiennent des fonctions de type `qcc_(parse|bld)*`, où \* représente le type de message concerné. Chaque fonction est en fait un sous-appel à une fonction de `cc.c` avec des paramètres modifiés. On utilisera donc les fonctions de l'un ou de l'autre suivant que l'on désire accéder aux fonctions de `cc.c` à partir du noyau (dans AAR et Q93B) ou de l'espace utilisateur (avec `tstqcc` par exemple). La figure 7.3 synthétise les relations de ces différents fichiers ainsi que leurs fonctions. Ces deux fichiers contiennent également des fonctions permettant d'allouer les zones mémoires nécessaires au stockage des messages ainsi qu'à la construction des en-têtes (parties de contrôle des messages, utilisées par le stream).

Il est difficile de séparer les rôles des deux drivers dans la génération des messages car les deux produisent et récupèrent le contenu de certains messages comme le prouve le tableau 7.1. Cependant on remarque que la gestion des messages concernant l'établissement et la terminaison des connexions est essentiellement réalisée au niveau du driver AAR. Le driver Q93B a pour but de mettre en œuvre le protocole de même nom afin de gérer l'état des connexions et, de vérifier le contenu des messages qui lui sont transmis. La gestion de l'état des connexions se fait au moyen d'un automate d'état fini associé à chacune d'entre elles, le passage d'un état à l'autre se faisant soit par le déclenchement d'un timer, soit par la réception d'un message d'un des modules adjacents. Les états, messages, timers ainsi que la valeur des timers sont spécifiés dans [For94]. La vérification consiste à repérer les IEs non valides inclus dans le message. Les

Version	Header Length	Total Length	
Identification		Flags	Offset
Time to live	protocol	Header checksum	
IP source adress			
IP destination adress			
Options			
			Padding
IP DATA			

FIG. 7.4 - *Structure d'un paquet IP*

IEs absents ne sont pas repérés. Dans le cas où un IE non valide est repéré, le message est détruit. Un IE est déclaré invalide lorsque son numéro est inconnu de Q93B.

Les IEs comme les messages sont construits conformément à la norme Q93B décrite dans [For94]. Une fois un message construit, un pointeur sur celui-ci est passé au module inférieur adjacent.

### 7.2.2 Structure d'IP

#### Fonctionnement du protocole:

Le protocole IP est un protocole de niveau réseau fournissant un service best-effort, non fiable, sans connexion. Pour fournir ce service, IP utilise des PDUs appelés paquets dont la structure est décrite dans 7.4. Les personnes désirant obtenir des renseignements plus précis sur IP peuvent consulter [Com91].

#### Fonctionnement du driver:

Le driver IP est un driver utilisant le concept des streams; il reçoit les messages provenant de TCP, ICMP et des resolvers liés au support (AAR, ARP). Il effectue les traitements appropriés (construction de l'en-tête, segmentation, réassemblage, ...) afin de mettre en œuvre le protocole IP. Une fois ceux-ci

Numéro de port source				Numéro de port destination				
Numéro de séquence								
Numéro d'acquittement								
Longueur Header	Réservé	URG	ACK	PSH	RST	SYN	FIN	Taille de la fenêtre
TCP Checksum				Pointeur de données urgentes				
Options								
Données								

FIG. 7.5 - Description d'un paquet TCP

effectués, il passe les messages modifiés à son entité adjacente inférieure (LE ou SA dans notre cas). En pratique, contrairement à TCP qui utilise une instance du driver par connexion, IP est un multiplexeur qui reçoit des paquets de toutes les connexions. Pour faire le lien entre un paquet reçu et l'instance de TCP à laquelle il doit être associé, il utilise des informations du paquet IP (couple d'adresses) et de la trame TCP (numéros de ports).

### 7.2.3 Structure de TCP

#### Fonctionnement du protocole

Le protocole TCP est un protocole de niveau transport orienté connexion offrant un service fiable de bout en bout. Une connexion TCP est identifiée par deux couples de valeurs (adresse source, port source), (adresse destination, port destination). Les données TCP sont encapsulées dans des paquets dont la figure 7.5 montre la structure.

Le schéma 7.6 décrit le fonctionnement du protocole.

Afin de gérer le fait que TCP travaille au dessus d'un protocole fonctionnant en commutation de paquets mode datagramme on utilise un ensemble de

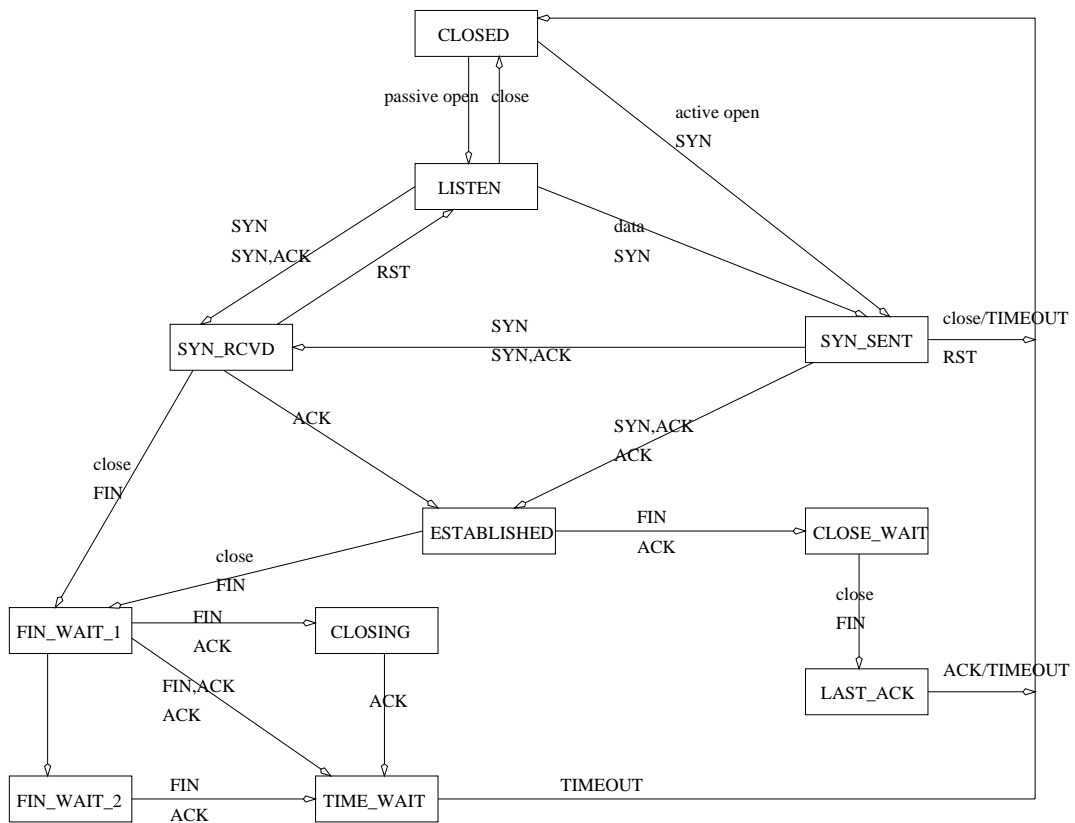


FIG. 7.6 - Automate de Mealy décrivant le fonctionnement du protocole TCP

mécanismes :

- Protocole en trois temps pour établir et libérer la connexion.
- Mécanismes de temporisation.
- Numérotation des paquets en séquence.
- Mécanisme d'acquittement.
- Retransmission des paquets perdus et utilisation d'une fenêtre d'anticipation.

Pour gérer le fait qu'IP n'assure pas l'intégrité des données on utilise un checksum sur l'en-tête et le contenu du paquet. Pour le lecteur intéressé [Com91] propose une description complète du protocole et de sa mise en œuvre.

### Fonctionnement du driver

Le driver TCP est un driver utilisant le concept des streams ; il reçoit les messages provenant des TLIs. Il effectue les traitements appropriés (construction de l'en-tête, ...) afin de mettre en œuvre le protocole TCP. Une fois ceux-ci effectués, il passe les messages modifiés à IP. En fait le passage des messages entre le milieu utilisateur et le driver TCP n'est pas direct ; un certain nombre de modules (*timod*, *sockmod*, *tlimod*, *tirdwr*) sont intercalés entre les deux et traduisent les primitives utilisateur en appel systèmes au niveau TCP.

A chaque connexion correspond une structure de donnée particulière (*tcp\_t*) qui permet la gestion de la connexion dans un thread dédié.

Le driver TCP est construit à partir du fichier *drv/tcp/new/tcp.c* .

Nous détaillons ci-dessous le fonctionnement des principales fonctions de communications au niveau TCP :

- Allocation du port de communication :  
Le serveur ou le client émet un message de type *T\_BIND\_REQ*. Ce message est reçu par la procédure *tcp\_wput* de TCP. Celle-ci fait appel à *tcp\_bind* qui alloue le port de communication. En fonction du résultat de l'allocation un message d'acquittement ou d'erreur est généré et envoyé à l'entité adjacente supérieure par la procédure *tcp\_rput* .
- Demande de connexion :  
Le client fait une demande de connexion en envoyant un message de type *TCP\_CONN\_REQ*. Ce message est récupéré par la procédure *tcp\_wput* de TCP. Celle-ci fait appel à *tcp\_connect* qui extrait les informations relatives à la communication et envoie un message au driver TCP du serveur par un *putnext*. Celui-ci reçoit le message et le traite par la procédure *tcp\_rput*. Celle-ci fait appel à la procédure *tcp\_accept* qui envoie des messages d'établissement de connexion de type *TCP\_OK\_AK* à son entité adjacente supérieure et à son entité homologue.

- Emission de données :  
Le client envoie un message contenant les données de type M\_DATA au driver TCP. Celui reçoit ce message par la procédure *tcp\_wput*. Celle-ci effectue les traitements appropriés et le passe à son entité homologue en faisant un *putnext*.
- Réception de données :  
Les messages de données sont reçus par la procédure *tcp\_rput*. Celle-ci les passe à son entité adjacente supérieure en faisant un *putnext*.

Il faut noter que le paquet IP est construit en partie au niveau du driver TCP.

#### 7.2.4 Fonctionnement des TLIs

Les TLIs (Transport Level Interface) sont un ensemble de procédures de programmation basées sur les fonctionnalités définies par la spécification ISO du service de niveau transport (ISO 8072). Les procédures TLI permettent d'établir, de gérer une connexion et de transférer des données (elles permettent aussi de gérer un transfert de données sans connexion mais nous ne nous intéresserons pas à ce point).

Les TLIs ne sont pas liées à un protocole particulier. Cette indépendance se fait au moyen de structures de données paramétrables dont l'allocation et l'initialisation se fait au moment de leur utilisation. Les primitives TLIs utilisent le contenu de ces structures pour construire des messages qui sont passés à l'entité de niveau transport sous-jacente. L'exemple ci-dessous montre le fonctionnement d'un client-serveur utilisant les TLIs sur TCP :

##### Côté client :

- Ouverture d'un sap de niveau transport (*t\_open*).
- Attachement au sap (*t\_bind*).
- Récupération de l'adresse du sap transport du serveur (traduction de son nom en son adresse par une fonction propre au protocole de niveau transport utilisé).
- Allocation et initialisation de la structure de données permettant de stocker l'adresse obtenue (*t\_alloc*).
- Envoi d'un message de connexion à cette adresse (*t\_connect*).
- Echange de données avec le serveur (*t\_snd* et *t\_rcv*).
- Libération ordonnée de la connexion (*t\_sndrel* et *t\_rcvrel*).

**Côté serveur :**

- Ouverture d'un sap de niveau transport (*t\_open*).
- Attachement au sap (*t\_bind*).
- Allocation et initialisation de la structure de données permettant de stocker l'adresse (*t\_alloc*).
- Attente d'une connexion sur le premier sap (*t\_listen*).
- Allocation et initialisation de la structure de données permettant de stocker l'adresse de l'appelant (*t\_alloc*).
- Ouverture d'un autre sap de niveau transport pour accepter la connexion et libérer le premier sap (*t\_open*).
- Attachement à ce second sap (*t\_bind*).
- Acceptation de l'appel (*t\_accept*).
- Echange de données avec le client jusqu'à une demande de déconnexion (*t\_snd* et *t\_rcv*).
- libération ordonnée de la connexion (*t\_sndrel* et *t\_rcvrel*).

### 7.3 Modification de la signalisation

La modification de la méthode de construction et de récupération du contenu des IEs a eu un impact sur toute la signalisation. La figure 7.7 présente les modifications effectuées.

Les fonctions ajoutées et modifiées utilisent des structures de données de grande taille. Nous nous sommes aperçu que l'utilisation de variables statiques de grande taille augmente l'instabilité des drivers. Or les variables statiques sont la seule solution si l'on désire pouvoir utiliser un source aussi bien dans le noyau que dans l'espace utilisateur. Afin de répondre à ce problème, nous avons dû créer deux types de bibliothèques, le premier utilise un mécanisme d'allocation dynamique de la mémoire propre au noyau alors que le second utilise des variables statiques et des fonctions d'allocation de mémoire dynamique du milieu utilisateur. Mis à part la gestion de la mémoire, les sources servant à construire les deux types de bibliothèques sont identiques.

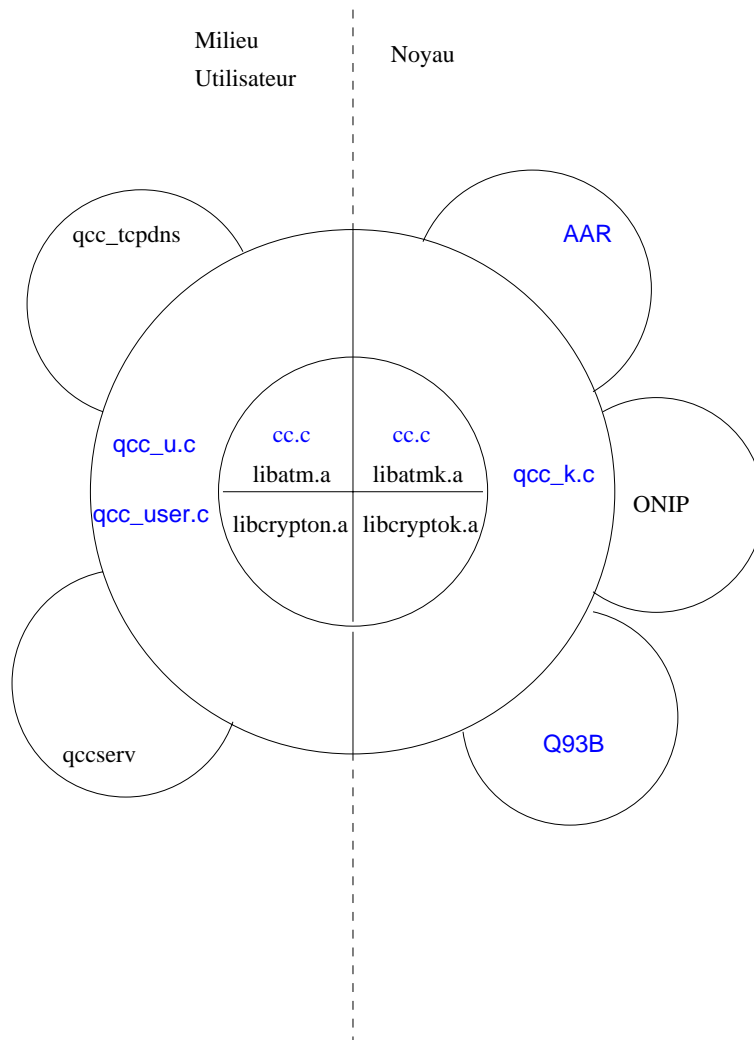


FIG. 7.7 - Relations entre les fichiers servant à construire les messages de signalisation après modification

### 7.3.1 libcrypton.a et libcryptok.a

Nous avons créé deux bibliothèques dédiées aux fonctions cryptographiques ; Une première destinée à être utilisée dans le milieu utilisateur et une seconde adaptée au noyau. Elles rendent les services suivants :

- Ajout des fonctions de chiffrement et de déchiffrement DES :

Nous avons adapté un DES en mode ECB afin qu'il fonctionne aussi bien dans le noyau que dans le milieu utilisateur. Nous avons ensuite ajouté ce DES aux librairies cryptographiques libcrypton.a et libcryptok.a. Le DES est composé d'un ensemble de fichiers contenus dans les répertoires `drv/crypto/DES/` et `drv/cryptok/DES/`. Les fonctions de chiffrement et de déchiffrement sont respectivement les suivantes :

```
void edes(unsigned char*, unsigned char*, int, unsigned char*);
```

```
void ddes(unsigned char*, unsigned char*, int, unsigned char*);
```

les paramètres étant un pointeur sur le message à (dé)chiffrer, un pointeur sur la clef, la taille du message, un pointeur sur le message (dé)chiffré. Le fichier d'en-tête correspondant se trouve dans `drv/include/crypto/`.

- Ajout de la fonction de calcul de résidu MD5 :

Comme dans le cas du DES nous avons adapté le MD5 et nous l'avons ajouté à libcrypton.a et à libcryptok.a. Le MD5 est composé d'un ensemble de fichiers contenus dans les répertoires `drv/crypto/MD5/` et `drv/cryptok/MD5/`. La fonction de calcul du MD5 est la suivante :

```
void MDString(u_char *, unsigned int, u_char *);
```

Les paramètres étant un pointeur sur le message dont on doit calculer le résidu, la longueur de ce message et un pointeur sur le résidu. Le fichier d'en-tête correspondant se trouve dans `drv/include/crypto/`.

- Ajout des fonctions de chiffrement et de déchiffrement RSA :

La fonction de (dé)chiffrement RSA a été modifiée et ajoutée à libcrypton.a et à libcryptok.a. Le RSA est composé d'un ensemble de fichiers contenus dans les répertoires `drv/crypto/RSA/` et `drv/cryptok/RSA/`. Ces fichiers sont identiques à l'exception du fait que les instructions d'allocation et de libération de mémoire n'existant pas dans le noyau nous avons dû les réimplémenter à l'aide des primitives noyau. Ceci est réalisé dans le fichier `malloc.c`. La fonction de calcul du RSA est la suivante :

```
unsigned char *M RSA(unsigned char *, unsigned char *, unsigned char *, unsigned char *, long int, long int, long int, long int *);
```

Les paramètres étant un pointeur sur le message à (dé)chiffrer, un pointeur sur l'exposant, un pointeur sur le modulo, un pointeur sur le message

(dé)chiffré, la longueur du message à (dé)chiffrer, la longueur de l'exposant, la longueur du modulo, et un pointeur sur la longueur du message (dé)chiffré. La fonction renvoie 1 en cas d'erreur. Le fichier d'en-tête correspondant se trouve dans `drv/include/crypto/`.

- Ajout des fonctions de signature KeyedMD5 :

La fonction `KeyedMD5` est une fonction de signature beaucoup plus rapide que le RSA basée sur le MD5. Elle a été modifiée et ajoutée à `libcrypton.a` et à `libcryptok.a`. La fonction est composée du fichier `keyedmd5.c` contenu dans les répertoires `drv/crypto/KeyedMD5/` et `drv/cryptok/KeyedMD5/`. La fonction de calcul du KeyedMD5 est la suivante :

```
int keyedmd5(unsigned char *, int, unsigned char *, int, unsigned char *)
```

Les paramètres étant un pointeur sur la clef, la longueur de celle-ci, un pointeur sur le message à signer, la longueur de ce message et un pointeur sur la signature. La fonction renvoie la taille de la signature. Le fichier d'en-tête correspondant se trouve dans `drv/include/crypto/`.

### 7.3.2 libatm.a et libatmk.a

Comme pour les fonctions cryptographiques, nous avons créé deux bibliothèques afin de régler les problèmes posés par l'allocation mémoire.

- Ajout des fonctions concernant la signature et le chiffrement. Nous avons ajouté dans `cc.c` des fonctions permettant d'utiliser les fonctions décrites précédemment.

```
- unsigned char *get_sign(security_policy*, unsigned char *, int, unsigned char, unsigned char, qcc_sec_params_t*);
```

Cette fonction a pour but la construction ou la vérification de la signature conformément aux paramètres de la politique de sécurité.

Les paramètres sont : un pointeur sur les paramètres de la politique de sécurité, un pointeur sur le message, la longueur de ce message, le type d'action à effectuer, la taille du bourrage (celui-ci ne devant pas être signé car son contenu est variable), un pointeur sur le contenu de l'IE de sécurité (dans laquelle nous stockerons la signature). Dans le cas où le type d'action est une vérification de la signature, la fonction renvoie 1 pour une signature valide et 0 dans le cas contraire.

Pour calculer la signature dans le cas d'une signature RSA, nous construisons un message contenant tous les IEs à signer, puis nous calculons le résidu MD5 relatif à ce message. Enfin nous chiffons ce résidu par l'algorithme RSA. Pour vérifier la signature, nous procédons de même mais au lieu de chiffrer le résidu MD5, nous déchiffrons la signature reçue et la comparons avec le résidu calculé.

Dans le cas d'une signature KeyedMD5 nous construisons un message contenant tous les IEs à signer puis nous signons ce message avec l'algorithme KeyedMD5. Pour vérifier la signature, nous agissons de même, puis nous comparons la signature calculée avec la signature reçue.

- `int sizeofs(unsigned char);`  
Cette fonction calcule la taille de la signature dont le type lui est passé en paramètre.  
La fonction renvoie la taille de la signature correspondante.
- `unsigned char inic(unsigned char*, unsigned char, unsigned char);`  
Lorsque nous cherchons à savoir si un IE doit être intégré dans l'IE de sécurité afin d'être chiffré nous utilisons cette fonction afin de savoir si le descripteur de l'IE est contenu dans la liste des IEs à chiffrer fournie dans les paramètres de sécurité.  
Les paramètres sont un pointeur sur la liste d'IEs, la taille de cette liste et l'IE recherché. La fonction renvoie 1 lorsque l'IE est présent dans la liste et 0 dans le cas contraire.
- `int get_enc_size(unsigned char*, int);`  
Cette fonction permet de calculer la taille de la zone chiffrée dans un message.  
Les paramètres sont un pointeur sur le message et sa taille.

- Ajout des fonctions de manipulation de l'IE de sécurité :  
Afin de traiter l'IE de sécurité, nous avons ajouté dans le fichier *cc.c* contenu dans *drv/qcc/* une fonction de construction de l'IE de sécurité et une fonction de récupération des valeurs de l'IE de sécurité. Ces fonctions sont les suivantes :

```
static void cc_add_sec_params(unsigned char**, int*, qcc_sec_params_t*, security_policy*);
```

Les paramètres sont un pointeur sur le message à construire, un pointeur sur la longueur de ce message, un pointeur sur le contenu de l'IE de sécurité, un pointeur sur les paramètres de la politique de sécurité.

Les paramètres de l'IE de sécurité sont ajoutés d'une manière analogue à celle des autres IEs. Cependant il existe trois particularités :

- La signature étant calculée en partie sur l'IE de sécurité, il est impossible de remplir correctement le message à ce niveau. On effectue donc seulement une réservation de la place pour la signature.
- Le chiffrement DES ECB se faisant sur un message de taille multiple de huit octets, nous ajoutons un paramètre de bourrage afin que le dernier bloc de la signature n'ait pas un contenu aléatoire (ce qui engendrerait l'impossibilité de le déchiffrer pour la partie homologue).

- Certains champs étant de longueur variable, nous ajoutons, lorsque c'est le cas, la taille du champ après son descripteur dans le message.

```
static void cc_get_sec_params(unsigned char**, qcc_sec_params_t*);
```

Les paramètres sont un pointeur sur le message à construire et un pointeur sur le contenu de l'IE de sécurité. La récupération des paramètres de l'IE de sécurité se fait d'une manière analogue à celle des autres IEs.

- Ajout de la fonction de négociation des paramètres de la politique de sécurité :

Afin de déterminer une politique de sécurité acceptable des deux partis désirant communiquer, nous avons créé une fonction de négociation des paramètres. Cette négociation se fait en comparant les préférences en matière de politique de sécurité du parti local exprimées par les paramètres de la politique de sécurité passés en paramètres et celles du parti homologue reçues dans le message de signalisation SETUP. En cas de désaccord (par exemple dans le cas où aucun algorithme de chiffrement commun n'a été trouvé) le message d'erreur correspondant est renvoyé.

```
unsigned char negociation(qcc_sec_params_t*, security_policy*);
```

Les paramètres sont un pointeur sur l'IE de sécurité et un pointeur sur les paramètres de la politique de sécurité. La fonction renvoie le résultat de la négociation.

- Modification des fonctions de construction et d'analyse des messages :

Nous avons modifié les fonctions de construction et d'analyse des messages SETUP et CONNECT c'est à dire les fonctions `cc_bld_setup`, `cc_bld_connect`, `cc_parse_setup`, `cc_parse_connect`.

L'en-tête de chaque fonction a été modifiée afin de prendre en compte les paramètres de la politique de sécurité. Cela se traduit par un paramètre supplémentaire (le dernier) de type `security_policy*`.

La construction des messages de signalisation se fait de la même manière dans les deux cas. On sépare d'abord le message en deux parties, d'une part le message contenant les IEs ne devant pas être chiffrées, de l'autre les IEs devant être contenues dans l'IE de sécurité. Le fait qu'une IE appartienne à une partie du message ou à une autre se décide au moyen de la liste des IEs à chiffrer incluse dans les paramètres de la politique de sécurité. On ajoute ensuite l'IE de sécurité puis on effectue la signature sur les IEs à signer. Enfin on chiffre une partie de l'IE de sécurité. Dans le cas où la politique de sécurité n'est pas définie, on construit un message normal.

L'analyse d'un message de signalisation se fait de la même manière dans le cas du SETUP et du CONNECT. On récupère d'abord les IEs non

chiffrées. On cherche ensuite si le message est sécurisé ou non en essayant de trouver l'identificateur de l'IE de sécurité dans le message. Si c'est le cas on déchiffre la partie chiffrée du message et on récupère le contenu des IEs chiffrées. On négocie ensuite la politique de sécurité à appliquer au message. Une fois celle-ci déterminée, on vérifie la validité du message reçu en examinant la signature ainsi que les horodateurs. La fonction renvoie le résultat de la négociation. Dans le cas où la politique de sécurité n'est pas définie, on traite le message comme non sécurisé.

### 7.3.3 Dans les fichiers d'en-tête

Nous avons défini dans les fichiers d'en-tête l'IE de sécurité, les valeurs pouvant être affectées aux champs de cet IE, la politique de sécurité, les valeurs pouvant être affectées aux paramètres de la politique de sécurité. Nous avons également mis à jour les en-têtes des fonctions modifiées.

- Dans *drv/include/atm/qcc.h* :
  - Modification des en-têtes des fonctions `qcc_bld_setup`, `qcc_parse_setup`, `qcc_bld_connect`, `qcc_parse_connect`.
  - Définition de l'identificateur de l'IE de sécurité.
  - Définition des identificateurs des champs de l'IE de sécurité. Les identificateurs sont des valeurs permettant de coder la structure de l'IE ou des IEs dans le message envoyé.
- Dans *drv/include/atm/qcctypes.h* :
  - Définition des tailles maximales des paramètres de l'IE de sécurité.
  - Définition de la structure contenant les paramètres de la politique de sécurité.
  - Définition de la structure de l'IE de sécurité. Cette structure est proche de celle employée pour stocker les paramètres de la politique de sécurité avec cependant certaines différences afin d'être conforme à la structure de l'IE de sécurité définie précédemment :
    - La structure contenant les paramètres de la politique de sécurité contient les préférences en matière de chiffrement ce qui n'est pas le cas de l'IE de sécurité car l'IE ne contient que l'élément d'authentification.
    - la taille de `ie_ic_list` est différente dans les deux structures car dans la première elle contient la liste des numéros d'IEs à chiffrer alors que dans la seconde elle contient les IEs eux-mêmes.
  - Modification de la structure des messages.

- Dans *drv/include/atm/limits.h* :
  - Définition de la taille maximale de l'IE de sécurité.
  - Modification de la taille maximale du message afin de prendre en compte la taille maximale de l'IE de sécurité.
- Dans *drv/include/q93b.h* :
  - Définition de l'identificateur de l'IE de sécurité (pour q93b).

### 7.3.4 Dans les fichiers *qcc\_k.c*, *qcc\_u.c* et *qcc\_user.c*

Les modifications de *cc.c* ont entraîné la modification des fonctions utilisant des fonctions définies dans *cc.c*. Nous les avons modifiées afin qu'elles prennent en compte les paramètres de la politique de sécurité.

### 7.3.5 Dans Q93B

- Modification des appels aux fonctions modifiées précédemment afin de prendre en compte la politique de sécurité. Cependant pour des raisons de performances (afin de ne pas vérifier deux fois la signature des messages en particulier) aucune politique de sécurité n'est définie ce qui conduit à ne pas vérifier la validité des informations récupérées et à ne pas pouvoir récupérer les informations contenues dans les IEs chiffrées. Cette modification n'a pas d'importance car :
  - En cas de problème (par exemple concernant l'authentification du message) la connexion sera libérée par une entité supérieure et les informations si elles sont erronées ne seront pas utilisées.
  - Les informations récupérées au niveau de q93b dans le message ne peuvent pas être chiffrées.
- Modification de la liste des IEs pouvant être contenues dans les messages de SETUP et de CONNECT.
- Modification de la valeur des timers afin de permettre la mise en œuvre des fonctions de sécurité.

### 7.3.6 Dans AAR

Modification des appels aux fonctions modifiées précédemment afin de prendre en compte la politique de sécurité. Cependant pour le moment la politique de sécurité n'est pas fixée. Il est à noter que nous aurions pu ne pas modifier ce driver sans que cela ne provoque d'erreurs au niveau de la vérification des IEs dans Q93B mais, la non modification aurait entraîné une incohérence dans les drivers (mélange de drivers utilisant des bibliothèques différentes) que nous avons préféré éviter.

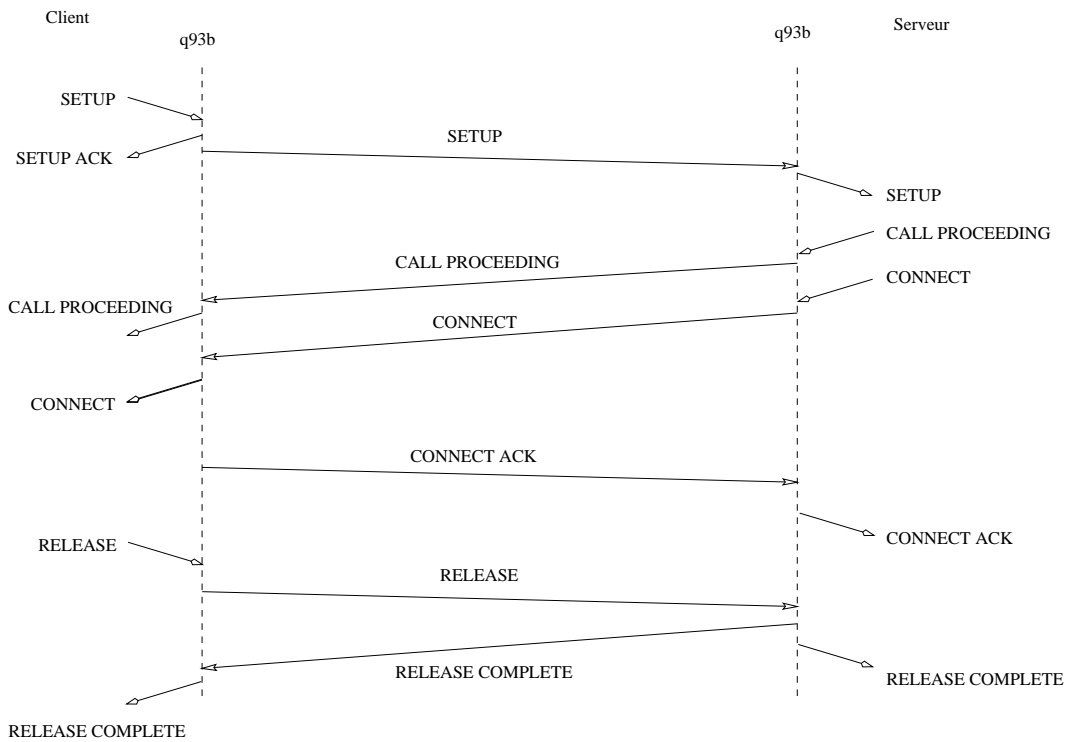


FIG. 7.8 - Fonctionnement du serveur de signalisation

### 7.3.7 Dans tstqcc

La figure 7.8 présente le fonctionnement des deux programmes obtenus à partir de la séparation de `tstqcc` en deux parties. Le serveur `qccserv` attend les demandes d'ouverture de connexions des clients `qcc_tcpdns`, c'est à dire des messages `SETUP`. Lorsqu'un `SETUP` est reçu, `qccserv` utilise les fonctions modifiées de `qcc_user.c` et négocie les paramètres de la politique de sécurité. Une fois ces paramètres fixés, `qccserv` choisit les paramètres de la connexion et envoie un `CALL PROCEEDING` puis un `CONNECT` au client. Celui-ci vérifie que les paramètres négociés sont conformes à sa politique de sécurité, puis, si c'est le cas accepte ces paramètres comme la nouvelle politique de sécurité à mettre en œuvre pour communiquer avec le serveur. La politique de sécurité est exprimée dans un fichier `/etc/atmsecparam`. Elle est lue au lancement du serveur ou du client par un traducteur dirigé par la syntaxe contenu dans le répertoire `drv/tests/analyseur/`. A chacun des programmes correspond un fichier C, respectivement `qccserv.c` et `qcc_tcpdns.c` contenus dans le répertoire `drv/tests/testq93b/`.

## 7.4 Utilisation des services de sécurité

### 7.4.1 Introduction

Pour pouvoir utiliser la politique de sécurité définie de part et d'autre au travers de la signalisation, deux approches sont possibles :

- Une première approche utilisant des modifications des drivers TCP, IP et de certaines bibliothèques. Les échanges fixant les paramètres de la politique de sécurité se déroulant dans le milieu utilisateur.
- Une approche utilisant une modification des drivers IP et TCP et la construction d'un module du noyau chargé des échanges des paramètres de la politique de sécurité et de l'établissement de la connexion ATM.

Ces propositions sont guidées par plusieurs raisons :

- Inadaptation d'une partie du driver SUN à notre modèle :  
Lorsqu'il désire établir une connexion entre deux machines, le driver vérifie d'abord si une connexion au niveau ATM existe déjà entre celles-ci. Si c'est le cas il l'utilise. Ceci va à l'encontre de notre modèle qui prévoit un établissement de connexion pour chaque demande d'ouverture de connexion afin de fixer une politique de sécurité pour chacune d'elle.
- Complexité du driver SUN.
- Existence de travaux se rapprochant de notre modèle :  
Les articles [Lam95] et [HA96] proposent l'utilisation directe de versions modifiées de TCP et d'IP sur ATM intégrant des mécanismes de contrôle de flux. Cette approche est plus adaptée à notre modèle puisqu'à chaque connexion TCP correspond une connexion ATM. De plus il est possible d'utiliser l'expérience de ceux ayant travaillé sur ce domaine afin d'accélérer notre implémentation.

Si on compare les deux approches proposées, on peut constater que la première est plus facile à mettre en œuvre. Cependant la seconde est plus sûre car l'utilisateur n'a pas accès aux variables cryptographiques et il ne peut pas utiliser la connexion d'un autre utilisateur comme pourrait le faire un utilisateur malintentionné dans le premier cas. De plus la seconde approche nécessite des modifications moindres des éléments existants. Ces deux raisons nous ont poussés à choisir la seconde approche.

### 7.4.2 Schéma général

La figure 7.9 présente le schéma général de notre implémentation. Elle se caractérise par les points suivants :

- Suppression d'une partie du package ATM original. Cette suppression englobe les modules et drivers ATMIP et AAR.

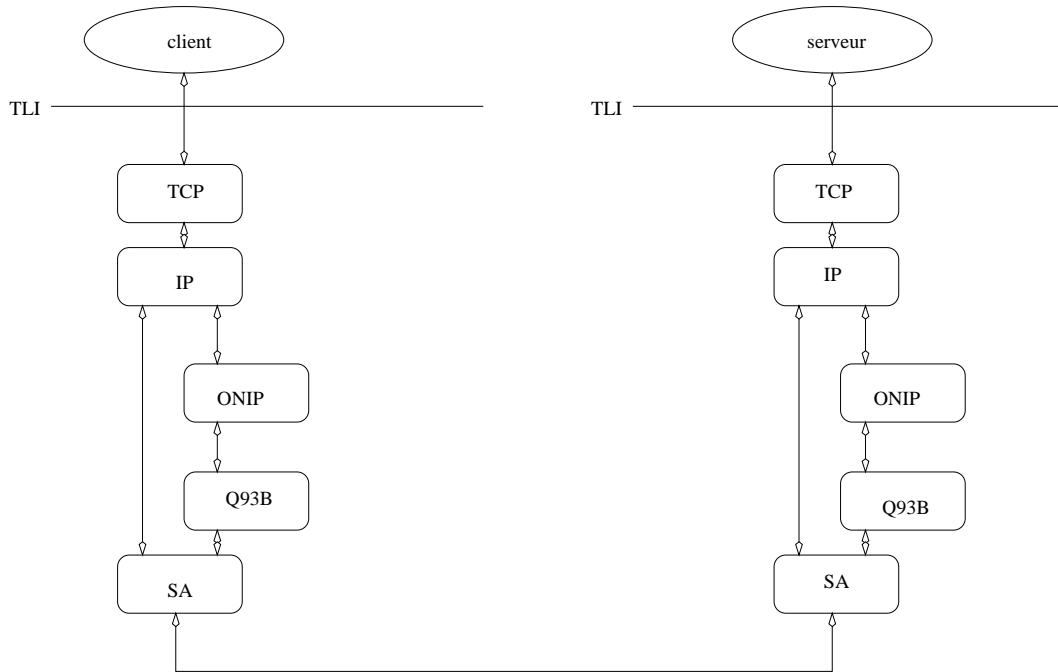


FIG. 7.9 - *Modèle d'une implémentation au niveau noyau*

- Introduction d'un nouveau driver chargé de la signalisation ; ONIP.
- Modification des drivers IP et TCP.
- Utilisation des modifications effectuées dans la signalisation.

Les modifications effectuées s'inspirent de travaux existants ([Lam95] et [HA96]) ayant été effectués dans le laboratoire. Afin de la rendre plus cohérente, la description qui suit ne fait pas la distinction entre les travaux que nous avons effectués et ceux dont nous nous sommes inspirés. Le lecteur intéressé par ces différences se reportera au chapitre 5.

### 7.4.3 Modification au niveau utilisateur

Comme nous l'avons dit précédemment, notre objectif est de pouvoir associer une connexion ATM à chaque connexion TCP et de pouvoir utiliser une politique de sécurité pour chacune de ces connexions. Le problème principal au niveau utilisateur est donc de pouvoir faire le lien entre une connexion TCP et une politique de sécurité. Une solution pourrait consister à associer de manière fixe à chaque couple (utilisateur, adresse appelée) une politique de sécurité, mais cette approche n'est pas envisageable car le nombre de couples peut être extrêmement grand ce qui peut rendre l'administration d'un tel système impossible. La solution que nous avons adoptée est celle des jetons. Elle présente de nombreux avantages :

- Elle est plus souple ; On peut associer des jetons à des utilisateurs, des groupes d'utilisateurs, des adresses, des groupes d'adresses, des applications ou à des associations de ces différents éléments.
- Elle est plus simple à gérer puisque l'administrateur peut synthétiser par un index un ensemble de paramètres définissant un ensemble de connexions. De plus cette simplicité augmente la sécurité de l'ensemble du système.
- Elle est plus simple à implémenter.
- Elle est plus rapide. La recherche de la politique de sécurité par index est plus efficace qu'une recherche par identificateur de l'utilisateur et par adresse.

Cependant l'utilisation d'un tel système a pour inconvénient de ne pas être transparent et entraîne des modifications dans les programmes utilisateurs. Ces modifications sont de deux types :

- Modifications permettant de passer l'index au driver TCP (pour faire le lien entre connexion TLI et connexion TCP):  
On passe l'index relatif à la politique de sécurité à appliquer à TCP au

moyen d'une structure de donnée modifiée; *sockaddr\_in* définie dans le fichier */usr/include/netinet/in.h*. Cette structure de donnée est utilisée afin de construire les messages passant entre l'interface des TLIs et le driver TCP, aussi, en la modifiant, on peut y passer des informations supplémentaires. Plus précisément on utilise le second paramètre de type *t\_call*.

- Modifications permettant de connaître l'index. Ces modifications dépendent de l'application utilisée. Ainsi si l'on désire faire un contrôle d'accès par application, il est possible d'utiliser pour chaque application un index pré-fixé. En revanche si l'on désire faire un contrôle d'accès par personne, il sera possible d'attribuer un index à chaque utilisateur et de modifier l'application afin de lui demander celui-ci à chaque demande de connexion ou au lancement de l'application.

La sûreté du contrôle d'accès vient du fait que la relation entre index et politique de sécurité est bijective car les index sont sensés être confidentiels. Cette confidentialité est assurée par l'officier de sécurité.

Une seconde modification du milieu utilisateur permet de trouver l'adresse ATM associée au nom ou à l'adresse IP désirée par l'utilisateur. Cette modification est réalisée en modifiant le serveur de nom DNS, les fichiers lui étant associés ainsi qu'en construisant une librairie de résolution de noms DNS en adresses ATM. L'adresse ATM est passée au driver TCP de la même manière que l'index.

#### 7.4.4 Le module ONIP

Le module ONIP a trois fonctions :

- établir une connexion ATM.
- établir par une négociation des paramètres de sécurité acceptables pour les deux partis. Ces paramètres de sécurité ne seront pas utilisés dans ONIP mais dans le module CRYPT.
- Passer les informations recueillies au cours des deux actions précédentes aux drivers les utilisant.

La figure 7.10 synthétise les actions effectuées par le driver ONIP au cours d'une demande de connexion.

La structure générale du module ONIP est calquée sur celle du module LANE. Toutes les fonctions de gestion du module sont identiques. Les modifications réalisées concernent la gestion des messages :

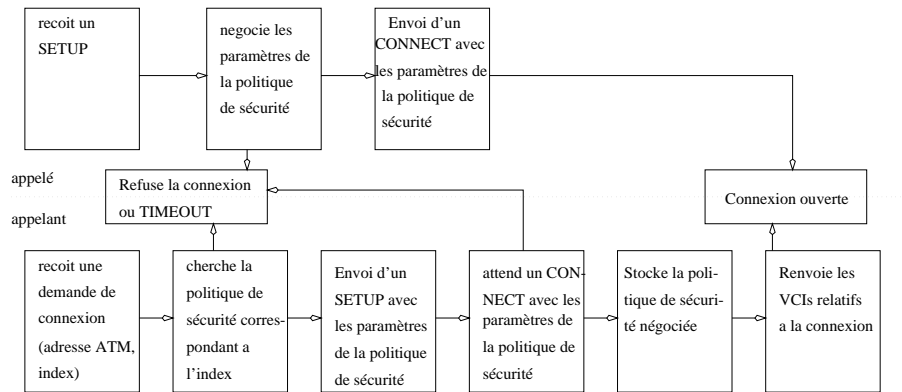


FIG. 7.10 - Diagramme représentant les actions effectuées au cours d'une demande de connexion chez l'appelant et l'appelé

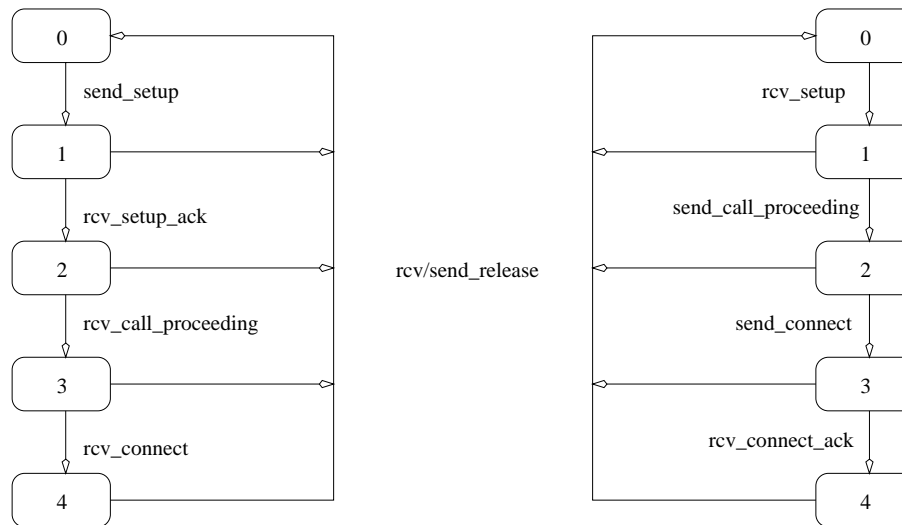


FIG. 7.11 - Automates de Moore décrivant les protocoles chez l'émetteur et le récepteur

### Gestion des messages de gestion de la connexion

Afin d'établir la connexion ATM, nous avons modifié la gestion des messages. On associe à l'état de la connexion l'état d'un automate. On passe d'un état à un autre par la réception ou l'émission d'un message. La figure 7.11 présente les différents états du protocole chez l'émetteur et le récepteur.

Les messages de gestion de la connexion (SETUP, CONNECT...) ont été modifiés afin d'utiliser les paramètres de la politique de sécurité.

Une fois la connexion établie, un message contenant les VCIs utilisés ainsi qu'un pointeur sur la politique de sécurité établie au cours de la négociation sont renvoyés au module supérieur adjacent.

### Gestion des messages de configuration du module

Afin de pouvoir configurer le module ONIP nous avons défini des messages de contrôle particulier SEC\_CTL et INT\_CTL permettant de passer respectivement des informations concernant les paramètres de la politique de sécurité et l'interface à utiliser.

- Le message de type SEC\_CTL : Il contient un pointeur sur une structure de donnée dans le milieu utilisateur. Cette structure de donnée définit l'opération à effectuer ainsi que ses paramètres. Les opérations possibles sont :
  - l'ajout d'une politique de sécurité correspondant à un index. Dans ce cas on recopie le contenu de la politique de sécurité dans une structure du noyau allouée dynamiquement.
  - La suppression d'une politique de sécurité correspondant à un index. Dans ce cas la structure du noyau correspondant à celle-ci est supprimée.
- Le message de type INT\_CTL : Il contient un pointeur sur le nom de l'interface que doit utiliser le module. Lorsque ce message est reçu, ce nom est recopié dans une structure interne au module.

#### 7.4.5 Modification de TCP

Nous proposons trois modifications principales pour TCP :

#### Récupération des paramètres de la politique de sécurité et des VCs utilisés

Comme nous l'avons dit précédemment, à chaque connexion correspond une structure de donnée permettant sa gestion (*tcp\_t*). En modifiant celle-ci nous pouvons stocker les VCs correspondant à la connexion TCP ainsi qu'une information permettant de trouver la politique de sécurité correspondante. Dans

notre cas cette information est un pointeur sur une structure contenant la politique de sécurité à appliquer. Ces informations sont obtenues de la manière suivante :

- Lorsqu'il reçoit une demande de connexion, TCP construit un message particulier contenant l'adresse ATM et l'index correspondant à la politique de sécurité et le passe à IP qui lui-même passe ce message à ONIP.
- Comme nous l'avons dit plus haut, ONIP établit la politique de sécurité à appliquer et les VCs utilisés et renvoie ces informations dans un message à IP qui le passe à TCP.
- Ces informations sont stockées dans une structure de données du driver TCP.

### **Modification de la structure du paquet IP**

Le paquet IP est en partie construit au niveau du driver TCP. Nous utilisons certains des champs inutilisés du paquet IP afin d'y stocker des informations servant soit dans des modules inférieurs, soit au niveau de l'entité homologue. Ces informations sont :

- Le VCI existant entre le récepteur et son commutateur d'accès correspondant à la connexion TCP. Cette information est utilisée au niveau du driver TCP homologue.
- On ajoute devant le paquet IP passé à IP le VCI sur lequel le paquet doit être émis.

### **Utilisation des informations ajoutées au paquet IP**

A la réception d'une demande de connexion TCP, on récupère le VCI placé dans le paquet IP. Celui-ci sera ensuite ajouté devant chaque paquet émis.

#### **7.4.6 Modification d'IP**

Les modifications sont les suivantes :

- suppression des fonctions de segmentation et de réassemblage, celles-ci étant réalisées par l'AAL5.
- suppression des fonctions de calcul de l'intégrité, ceci étant réalisé par l'AAL5.
- Afin de pouvoir diriger les paquets IP vers le thread TCP correspondant à la connexion à laquelle ils appartiennent, on utilise les numéros de ports mais également le VCI contenu dans l'en-tête du paquet IP (ceci est fait dans la fonction *atm\_ip\_rput\_local* ).

- À l'émission on récupère le VCI passé par TCP devant le paquet IP et on construit une en-tête SNAP de 12 octets que l'on place devant le paquet IP avant de le passer à SA. Cet en-tête est normalement fourni par le resolver lié au protocole sous-jacent (AAR ou ARP dans notre cas). Cet en-tête est composé du VCI (4 octets) et de l'encapsulation SNAP (8 octets).

#### 7.4.7 Les programmes annexes à ONIP : `laneplumb`, `laneunplumb`, `seclumb` et `intset`

Nous avons construit un certain nombre de programmes permettant de mettre en œuvre et de configurer le module ONIP :

##### **laneplumb et laneunplumb**

Ces deux programmes permettent respectivement d'insérer et de retirer le module ONIP du STREAM. Le module est placé entre le module IP et le driver SA. Pour cela on utilise des messages de contrôle.

La syntaxe d'utilisation des deux programmes est la suivante :

- `laneplumb` : insère le module ONIP dans le STREAM.
- `laneunplumb` : enlève le module ONIP du STREAM.

##### **seclumb**

`Seclumb` permet de fixer ou de supprimer les paramètres de la politique de sécurité relatifs à un index dans le noyau. Pour cela on passe dans un message de type contrôle l'adresse d'une structure de données contenant l'opération à réaliser, l'index sur lequel doit s'appliquer l'opération ainsi qu'éventuellement les paramètres de la politique de sécurité relatifs à cet index.

Les paramètres de la politique de sécurité sont fournis dans un fichier : `/etc/secparams` ]. Ces paramètres sont récupérés avant l'envoi du message au moyen d'un traducteur dirigé par la syntaxe contenu dans le fichier `getsp.c` situé dans le répertoire `/drv/kernel_level/`. L'exécutable correspondant se trouve dans le même répertoire.

Sa syntaxe d'utilisation est la suivante :

- `seclumb index [Set|Del]` : Insère ou supprime les paramètres de la politique de sécurité correspondant à `index` dans le noyau.

##### **intset**

`Intset` permet de spécifier sur quelle interface la signalisation doit s'effectuer. Son mode de fonctionnement est le même que celui de `seclumb`. Sa syntaxe est la suivante :

- `intset interface` : sélectionne l'interface pour la signalisation.

### 7.4.8 Utilisation de la politique de sécurité modifiée

L'utilisation sur les données utilisateur de la politique de sécurité négociée peut se faire de deux manières :

#### Le module CRYPT

Le module CRYPT est un module se situant entre IP et SA et qui a pour objectif d'effectuer le chiffrement et/ou la signature des paquets en provenance de IP ainsi que le déchiffrement et/ou la vérification de signature des paquets en provenance de SA. La récupération des paramètres de la politique de sécurité impose trois modifications :

- Au niveau de ONIP :  
On transmet un message de type particulier à IP au moment où la politique de sécurité a été négociée. Ceci se produit avant l'envoi du CONNECT chez le récepteur et après sa réception chez l'émetteur. Ce message contient un pointeur sur la politique de sécurité à appliquer ainsi que l'index de sécurité correspondant. Cet index peut être remplacé par le VCI correspondant à la connexion.
- Au niveau de IP :  
A la réception de ce message, IP envoie à son tour un message de type particulier au module CRYPT contenant les mêmes informations.
- Au niveau de CRYPT :  
On récupère les informations envoyées par IP. On se sert de celles-ci pour effectuer les chiffrements/déchiffrements ainsi que les autres services de sécurité du plan utilisateur.

Cependant cette méthode possède deux inconvénients :

- L'information de sécurité voyage en clair dans plusieurs modules du noyau. Ceci peut poser des problèmes de confidentialité et/ou d'intégrité de cette information.
- Cette méthode n'est pas transparente et nécessite la modification de IP. Cette modification diminue la portabilité du module puisqu'il n'y a plus d'indépendance entre les couches.
- Cette méthode n'est pas évidente à mettre en œuvre car il faut assurer la synchronisation entre la politique de sécurité et l'ouverture de connexion TCP.

### **Utilisation au niveau de la carte**

Il serait envisageable d'utiliser une carte ATM (dotée de composants programmables) qui en plus de ses fonctions habituelles réaliserait des fonctions cryptographiques de chiffrement, de signature ainsi que des fonctions de bourrage. Il serait possible dans ce cas de modifier ONIP afin qu'il pilote la carte en lui fournissant les paramètres de la politique de sécurité.

Ceci peut être fait en envoyant des messages spécifiques au driver SA. Ces messages pourraient permettre de télécharger des microcodes dans les composants programmables de la carte et d'informer SA des politiques de sécurité à appliquer sur les connexions ouvertes.

Avec cette solution, on pourrait réaliser les opérations cryptographiques ou de bourrage au niveau AAL ou ATM de manière plus sécurisée que dans le module CRYPT.



# Bibliographie

- [CB94] B. Cheswick and S. Bellovin. *Firewalls and internet security, Repelling the wily hacker*. Addison-wesley Publishing company, 1994.
- [Chu96] Saw-Cheng Chuang. Securing atm networks. *Third ACM conference on computer and communication security*, 1996.
- [Com91] Douglas Comer. *Internetworking with TCP/IP, Volume 1: principles, protocols, and architecture, second edition*. Prentice hall, 1991.
- [For94] ATM Forum. Atm user-network interface specification version 3.1. Technical report, ATM Forum, 1994.
- [HA96] O. Elloumi H. Afifi, D. Bonjour. Tcp over non-existent ip for atm networks. *Proceedings of the 7th Joint European Networking Conference.*, 1996.
- [IB93] J. Ioannidis and M. Blaze. The architecture and implementation of network-layer security under unix. *USENIX Conference Proceedings*, 1993.
- [ISO90] ISO. Interconnection de systèmes ouverts , modèle de référence de base, partie 2 : Architecture de sécurité. Technical report, AFNOR, 1990.
- [Lam95] L. Lamti. Contrôle de trafic des applications multimédia sur les réseaux atm. Master's thesis, Telecom Bretagne, 1995.
- [Lau96a] Maryline Laurent. L'intégration d'un module de sécurité dans les commutateurs atm. Technical report, Telecom Bretagne, 1996.
- [Lau96b] Maryline Laurent. Sécurité atm : analyse de la sécurité des échanges sur l'architecture du lan emulé. *CFIP 96*, 1996.
- [Lau96c] Maryline Laurent. Security flows analysis of the atm emulated lan architecture. *IFIP Conference on Communications and Multimedia Security*, 1996.

- [ML95] Pierre Rolin Maryline Laurent. Sécurité atm : une analyse de flux menée sur quatre architectures de réseaux. *GRES'95*, 1995.
- [Pad93] Michael Padovano. *Networking Application on UNIX System V Release 4*. Prentice hall, 1993.
- [RD95] A. Lazar R. Deng, L. Gong. Securing data transfer in asynchronous transfer mode networks. *Proceedings of Globecom'95*, 1995.
- [Rol95] Pierre Rolin. *Réseaux haut débit*. Edition HERMES, 1995.
- [Sun94] Sunsoft. *Streams programmers guide*. Sun Microsystems Computer Company, 1994.
- [Sun95] Sun Microsystems Computer Company. *SunATM-155 SBUS Cards Manual*, 1995.