HAL
archives-ouvertes.fr

Génération des séquences de test pour les protocoles de sécurité dans IPv6

Vladimir Nemchenko

▶ **To cite this version:**

Vladimir Nemchenko. Génération des séquences de test pour les protocoles de sécurité dans IPv6. [Interne] A00-R-450 || nemchenko00a, 2000, 16 p. <inria-00107868>

HAL Id: inria-00107868
https://hal.inria.fr/inria-00107868

Submitted on 19 Oct 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

GENERATION DES SEQUENCES DE TEST POUR LES PROTOCOLES DE SECURITE DANS IPv6

Rapport de Recherche de Professeur Invité
V. NEMCHENKO
Décembre 2000

Projet RESEDAS, INRIA Lorraine
Responsable Prof. A. SCHAFF
615 rue du Jardin Botanique
54600 VILLERS LES NANCY

CONTENU :

1. TEST DES PROTOCOLES

1.1. Modèles des réseaux

Le premier problème qui se pose dans le test des protocoles c'est le choix et la détermination d'un modèle de représentation formelle d'un protocole. Aujourd'hui, le modèle d'automate d'états finis est choisi parmi d'autres comme un modèle de base.

Un automate S est décrit comme suit :

$$\mathbf{S = \{A, X, Y, a_0, f, f'\}}$$

où **A** – ensemble des états d'un automate,
X - ensemble des événements d'entrée,

Y - ensemble des événements de sortie,
$\mathbf{a_0}$ - état initial,
f – fonction de transitions d'automate,
f' - fonction des événements sorties.

De la théorie des automates d'états finis qu'il y a certaine nombre de modèles. Parmi ceux-ci on distingue deux modèles principaux [MEALY55]:

- Modèle de *Moore*,
- Modèle de *Mealy.*

La description de ces modèles peut se faire sous la forme d'un graphe dont les nœuds sont les différents états possibles du système, et les arcs (les transitions), les événements qui font passer l'automate d'un état à un autre. Ici les événements sont les mots d'entrée x_i et les mots de sortie y_j **.**

Pour l'automate de Moore les mots d'entrée correspondent aux transitions et les mots de sortie correspondent aux nœuds, c'est à dire, les événements de sortie ne se produisent que dans l'état stable de l'automate. Lui limite essentiellement l'utilisation du modèle de Moore pour le test des protocoles. Dans notre cas ce modèle ne nous intéresse pas car c'est un modèle théorique sans application pratique.

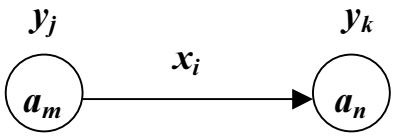

Figure 1 : Automate de Moore

La figure 1 représente le graphe dit automate de Moore ayant deux états stables a_m et a_n auxquels les mots de sortie y_j et y_k correspondent. La transition d'un état à l'autre est provoquée par le mot d'entrée x_i.

La forme analytique présentant l'automate de Moore est la suivante :

$$a(t+1) = f\ [a(t), x(t)]\ ;$$
$$y(t) = f'[a(t)].$$

Présentons l'automate de Mealy de la même manière.

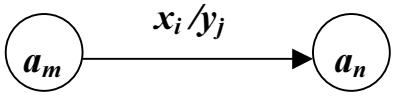

Figure 2 : Automate de Mealy

Sur la figure 2 le graphe d'un automate de Mealy est représenté. Il a aussi deux états stables a_m et a_n. Le mot de sortie y_j est apparu pendant la transition d'un état à l'autre provoquée par le mot d'entrée x_i.

La forme analytique de cet automate de Mealy est la suivante :

$$a(t+1) = f\ [a(t), x(t)] ;$$
$$y(t) = f\ '[a(t), x(t)].$$

Les études de spécifications formelles des protocoles nous disent qu'une spécification en Estelle [EST89], Lotos [LOT89] ou SDL [SDL88] peut se ramener à l'automate sous-jacent [CAS94]. Il faut remarquer que c'est le modèle de Mealy qui décrit bien cet automate.

Vu tout cela, il est évident de prendre le modèle de Mealy comme base pour la génération des séquences de test.

1.2. Modèles des fautes

Il faut remarquer qu'un protocole peut être présenté soit comme un graphe de contrôle représentant l'automate, soit comme un graphe de flux de données qui représente le flux global des données entre les paramètres d'entré et de sortie et les variables de contexte. On examine dans la plupart des travaux comme [CAS94], [TASC98] la génération de suites de test pour le graphe de contrôle ayant la forme de modèle de Mealy. En se basant sur ce modèle le document [CAS94] classifie les fautes dans un automate d'états finis comme suit :

- **Faute de sortie.** A titre d'exemple, on prend l'automate présenté sur la figure 2 où y_j est l'événement de sortie correct. Dans le cas de faute de sortie y_j sera remplacée par un autre y_k . Avec cela, le transfert de l'état a_m vers l'état a_n est correct.
- **Faute de transfert.** Soient deux états a_m et a_n caractérisent un graphe où le transfert $a_m \rightarrow a_n$ est réalisé sous l'action du mot d'entré x_i en provoquant le mot de sortie y_j (voir figure 2). Si on arrive dans ces conditions à l'état a_s non prévue, on dit qu'une faute de transfert a lieu.
- **Faute d'état.** Elle a lieu quand les nombres d'états de spécification et d'implantation sont différents.

Il faut remarquer que dans le plupart de cas réels plusieurs fautes peuvent se présenter en même temps sur un automate d'états finis comme il est montré sur la figure 3.

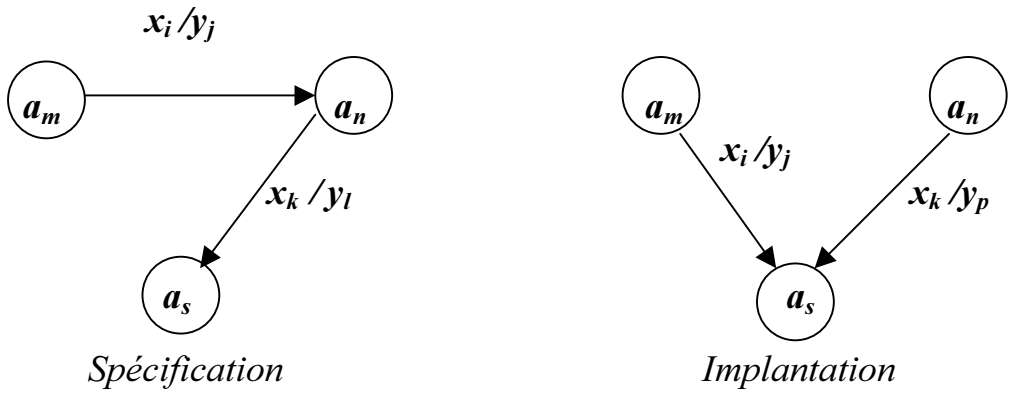

Figure 3 : Erreur d'implantation

Soient les trois états a_m , a_n , a_s présentés dans un graphe (cf. figure 3). Sous l'action de x_i un transfert $a_m \rightarrow a_n$ avec le mot de sortie y_j est prévu. Supposons qu'un transfert $a_m \rightarrow a_s$ avec x_i / y_j ainsi qu'un transfert $a_n \rightarrow a_s$ avec x_k / y_p sont réalisés.

Dans ce cas les fautes de sortie et de transfert ont lieu. A savoir, la transition $a_n \rightarrow a_s$ donne la valeur de sortie y_p à la place de y_l prévue par la spécification. Les événements x_i / y_j provoquent la transition $a_m \rightarrow a_s$ et non $a_m \rightarrow a_n$ comme prévu par la spécification.

1.3. Méthodes de base pour la génération des séquences de test

On peut distinguer les types de tests qui suivent [CAS94].

1. **Tests de conformité** comparent un comportement de l'IST (Implantation Sous Test) avec celui prévue par la spécification initiale.

   Citons quelques causes de *non conformité*. Tout d'abord, c'est une erreur d'opération qui correspond à une erreur au niveau des sorties. Puis, c'est une faute de transfert qui correspond à une erreur de transition entre états du système (voir *1.2. Modèles de fautes*).

   Il est évident que pour mettre en place un test de conformité il faut avoir une spécifisscation de système, une architecture qui permet de placer l'IST pour pouvoir observer son comportement, un jeu de tests et enfin, un moyen d'analyser l'observation du comportement de l'IST.

Concernant le test de conformité on peut citer la notion de test standardisée dans l'ISO [ISO9646] :

- **Une interaction** est un événement interne, entrant ou sortant pour l'IST ;
- **Un test** (ou « *test case* ») est un ensemble d'interactions (appelé aussi *séquence de test*) , vérifiant une propriété ou un comportement de l'IST ;
- **Un groupe de tests** est un ensemble de tests, vérifiant un type de fautes ;
- **Une suite de tests** est un ensemble de groupes de test, vérifiant la conformité globale du protocole.

Suivant ISO [ISO9646] les tests de conformité comprennent quatre types de tests :

1.1. **Tests d'interconnexion de base** qui vérifient un comportement élémentaire du protocole (ouvrir/fermer une connexion, transmettre des données…).

1.2. **Tests de capacité** d'un protocole (option, classes de protocole…).

1.3. **Tests de comportement** dynamique du protocole.

1.4. **Tests pour la résolution de question de conformité** qui sont destinés à vérifier des propriétés particulières d'un protocole.

2. **Tests de robustesse** qui vérifient la capacité de l'IST à fonctionner dans un environnement «hostile», c'est à dire avec des paramètres non prévues par la spécification .
3. **Tests de performance** pour mesurer les paramètres de performance de l'IST.
4. **Tests d'interopérabilité** qui vérifient la capacité d'au moins deux entités à interopérer dans un contexte réel.

Parce qu'on utilise un graphe comme modèle d'automates d'états finis pour définir un protocole, le problème de génération de suites des tests peut être formulé aussi comme le problème de couverture d'un graphe. Autrement dit, il faut trouver un compromis entre la nécessité de passer tous les nœuds et toutes les transitions d'un graphe et la nécessité de diminuer la grandeur de la suite des tests.

Le problème en question fait référence à la complexité des problèmes en général et aux problèmes NP-complets en particulier. Examinons en qualité d'exemple très typique le problème du Poster Chinois (CPP : Chinese Postman Problem).

On sait que c'est le mathématicien suisse Léonard Euler qui en 1736 a formulé et a résolu le problème de ponts de Konigsberg. Pour le réaliser il a représenté la carte d'une région de la ville comme un graphe où les nœuds sont certains points de la ville reliés par les ponts (figure 4).

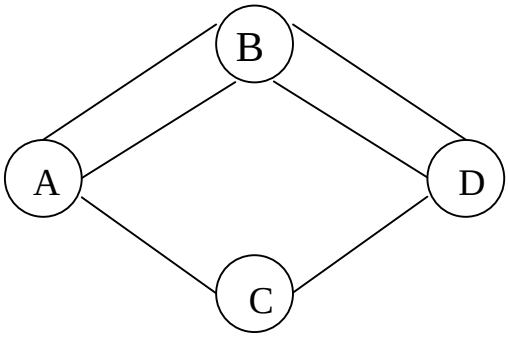

Figure 4 : Graphe du problème d'Euler

Après cela le problème a été formulé comme la nécessité de traverser chaque pont une fois en visitant tous les points donnés de la ville A,B,C,D en commençant et en terminant une voyage par le même point disons A.

L'algorithme et le programme le résolvant sont présenté dans http://www.geocities.com/SiliconValley/Peaks/1667/engl_start.ht

On peut aussi citer d'autres méthodes comme *Tour de transition* [NAITO81], *Méthode W* [CHOW78], *Méthode UIO (Unique Input Output)* [SAB88] et d'autres.

1.4. Expression des tests

Comme indiqué dans le titre de ce travail il s'agit de tests de protocoles IPv6 qui sont en train d'être développés actuellement pour les réseaux distribués. Et surtout des protocoles de sécurité et de mobilité. Dans le cadre de ces activités, on s'intéresse plus particulièrement aux tests de conformité qui correspondent mieux à ce type de réseaux.

En général, ce sont des experts qui connaissent profondément le comportement du protocole et qui donnent les suites des tests pour un protocole donné. La nécessité de standardisation d'une forme des suites de tests a provoquée la définition de la notation TTCN (Tree and Tabular Combined Notation) par l'ISO en 1986 [ISO9646, part 3]. Cette notation donne la description des séquences de test indépendante de l'architecture de test. Cette nsotation se compose de deux parties :

- **Partie déclarative** où on déclare des PAS (Points d'Accès au Service). Normalement les PAS sont notés : L (Lower Tester) et U (Upper Tester).

On déclare aussi des horloges et des UDS et UDP avec la valeur des paramètres.

- **Partie dynamique** définit une table présentant la séquence du test sous forme arborescente.

On trouve certains exemples du cas de test « sc 5 » dans [CAS94]. Donnons l'exemple d'expression de cas de test en TTCN cité dans cet article.

Dynamic Behavior			
Reference : transport protocol class 4 Identifier : sc5 Purpose : IST accepte un refus de connexion venant du testeur			
Dynamic Description	Label	Verdict	Comments
L ! CR U ? TDISind U ? TCONind U ! TCONresp L ? CC L ! AK L ! DT(TMP) U ? TDTind L ? AK U ! TDISreq L ? DR L ! DC U ! TCONreq L ? CR L ! DR U ? TDISind U ? others		Inconclusive Passed Fail	

Figure 5 : Expression TTCN du cas de test sc 5

Les caractéristiques principales de TTCN sont évidents àpartir de cet exemple présenté sur la figure 5. A savoir, les informations à identifier le cas de test sont présentées dans l'en-tête. L'axe horizontal nous donne l'évolution dans le temps et l'axe vertical donne la succession des interactions. Chaque interaction a son préfixe où **L** : Lower Tester, **U** : Upper Tester, **!** : émission, **?** : réception.

On a trois verdicts possibles :

- **Passed :** test réussi et conforme ;
- **Fail :** échec et non conformité ;
- **Inconclusive :** test inconcluant.

Il faut souligner que le principe de test de conformité de protocole est basé sur sa présentation sous la forme d'un automate à états finis (cf. *1.1. Modèles des réseaux)*. Pour établir la conformité ou non de l'IST il faut faire des mesures à

l'aide des testeurs sur chaque porte de l'IST. Naturellement, cela pose le problème de synchronisation entre ports. Dans [SAR84] un modèle d'automates à états finis à deux portes est introduit. [BOYD91] propose un algorithme de génération de séquences implicitement synchronisées de longueur minimale NP-complet.

1.5. Couverture de test

Il est évident que la spécification dans le cas général peut donner une suite théorique de test presque infinie. Mais un utilisateur d'un protocole pose toujours une question concernant une couverture de la suite des test qui donne la possibilité de déclarer ce protocole « conforme ». En autres termes, le problème consiste à déterminer la suite de test minimale qui couvre la spécification de manière satisfaisante.

Dans [CAS94] nous trouvons plusieurs approches. Par exemple, la couverture empirique d'une suite des tests donnée est examinée. On propose de mesurer la couverture comme :

- Le rapport du nombre des états rencontrés lors de l'exécution de la suite des tests et du nombre d'états de la spécification ;
- Le rapport du nombre des transitions passées dans le test et du nombre total de transitions de la spécification.

On propose aussi la génération d'une suite des tests pour une couverture donnée. On détermine l'ensemble des fautes possibles de l'implantation. Puis la génération de suite des tests couvrant un type de fautes est réalisée.

Une méthode intéressante est basée sur le calcul de métrique basé sur la distance entre une suite des tests et le jeu des séquences d'exécution dérivé de la spécification.

2. SECURITE DANS LES PROTOCOLES IPV6

2.1. Généralités

La solution optimales de réalisation de pratiquement tous les mécanismes de sécurité de réseaux est l'utilisation de troisième niveau de modèle ISO/OSI. C'est pourquoi la sécurité dans les réseaux est réalisée en utilisant des protocoles de niveau IP. De celle manière on donne la possibilité d'utiliser les mécanismes de sécurité standardisées aux protocoles de niveau supérieur. Ce sont des protocoles de contrôle de configuration et de routage [CIZ98].

On utilise la famille de spécifications **IPsec** élaborée par le groupe IP Security. Ce que est intéressant c'est que toutes ces spécifications peuvent être utilisées pour la sécurité de IPv4 ainsi que pour celle de IPv6.

On trouve la structure d'organisation de sécurité dans [KENTS98] . Présentons les composantes principales de cette structure sur la figure 6.

On voit ici des protocoles d'authenticité (AH) et de confidentialité (ESP) ainsi que des mécanismes réalisant la gestion de clés cryptographiques et plus bas, des algorithmes concrets de codage et de contrôle d'authenticité.

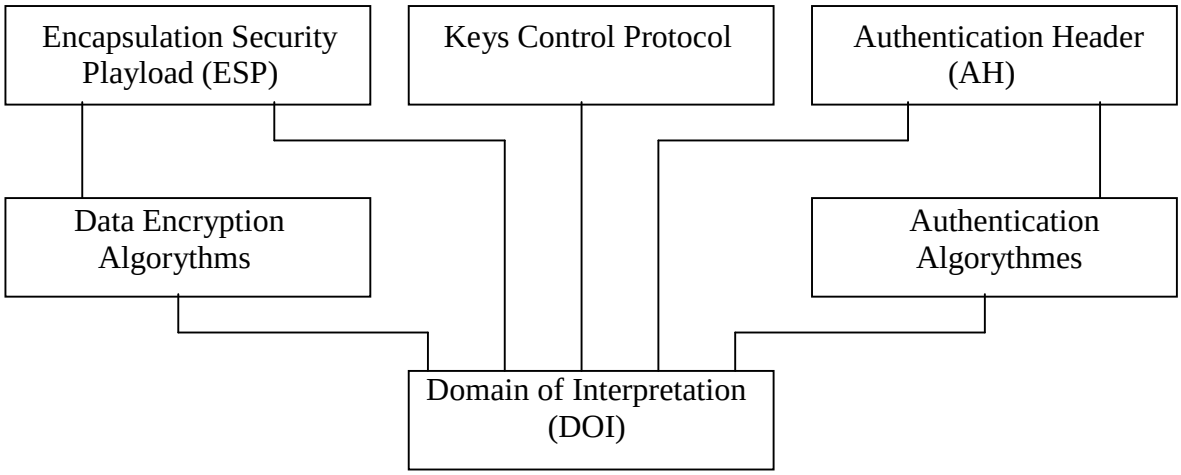

Figure 6 : Organisation de la sécurité dans IPv6

« Domain of Interpretation » jouant le rôle d'un fondement est une Base de Données où les informations sur les algorithmes, sur ses paramètres, sur ses identificateurs des protocoles sont stockées.

Il faut souligner que les protocoles d'authenticité et de confidentialité sont indépendants des algorithmes cryptographiques concrets. Ils peuvent être utilisés dans le régime de transport et dans le régime de tunnel. Nous nous n'intéressons ici qu'au régime de transport. Dans ce régime c'est le contenu du paquet qui est protégé.

L'indépendance des protocoles par rapport aux algorithmes cryptographiques exige la nécessité d'établir par les deux côtés un contexte de sécurité commun (Security Association : SA). Les questions générales de formation du contexte SA et de gestion des clés sont présentées dans la spécification «Internet Security Association and Key Management Protocol : ISAKMP [MAU98].

La formation du contexte SA se compose de deux phases :

- **Formation du contexte de contrôle** qui assure une route fiable autrement dit un canal authentifié et protégé .

- **Formation du contexte de protocoles** qui consiste à former des clés cryptographiques utilisées par les protocoles AH et ESP.

La structure de formation du contexte de contrôle est présentée sur la figure 7.

	Initiateur		*Partenaire*
(1)	Demande du contexte (proposition des algorithmes et de leurs paramètres).	→	
(2)		←	Algorithmes et paramètres acceptés.
(3)	Clé d'Initiateur et No. «jetable» d'Initiateur.	→	
(4)		←	Clé de Partenaire et No. «jetable» de Partenaire.
(5)	Message chiffré contenant l'identificateur d'Initiateur signé par sa clé secrète.	→	
(6)		←	Message codé contenant l'identificateur de Partenaire signé par sa clé secrète.

Figure 7 : Formation du contexte de contrôle.

Examinons cette structure.

- **Message (1) :** Dans sa demande du contexte Initiateur présente au Partenaire ses propositions concernant un ensemble d'algorithmes cryptographiques et des mécanismes pour les réaliser.
- **Message (2) :** Dans sa réponse, le Partenaire fait savoir son choix des algorithmes et des mécanismes.
- **Messages (3) et (4) :** Initiateur et Partenaire échangent des informations nécessaires pour créer une clé secrète commune. L'algorithme de Diffie-Hellman est utilisé pour cela [HARK98].
- **Messages (5) et (6) :** L'échange d'information d'identification signée à l'aide de la clé secrète de l'émetteur et chiffrée par la clé secrète commune est réalisé.

Il faut remarquer que Initiateur et Partenaire peuvent échanger leurs rôles. C'est à dire, la formation du contexte de contrôle est un processus bidirectionnel.

En principe, n'importe quel protocole peut être utilisé pour transmettre ISAKMP – messages. Mais c'est le protocole UDP sur le port No.500 qui est utilisé en pratique.

La figure 8 nous montre la structure de formation du contexte des protocoles.

	Initiateur		*Partenaire*
(1)	Résultat de HASH-fonction, demande du contexte (proposition des algorithmes et de leurs paramètres), No. « jetable » d'initiateur. Tous chiffré.	→	
(2)		←	Résultat de HASH-fonction, acceptés par Partenaire et les paramètres. No. « jetable » de Partenaire. Tous chiffré.
(3)	Résultat de HASH-fonction. Parmi des Arguments se trouvent deux No. «jetables».	→	

Figure 8 : Formation du contexte des protocoles

HASH-fonction présentée sur la figure 8 est une fonction destinée à assurer une transmission d'un test non modifié. C'est une fonction mathématique créant un nombre correspondant au texte. HASH-fonction est très sensible aux changements d'information dans un texte et elle se modifie beaucoup si même un caractère du texte est changé.

Un contexte de protocoles est formé sur la base du contexte de contrôle. Remarquons qu'une suite de messages présentées sur la figure 8 nous montre le cas simple quand la génération des clés des protocoles est réalisée sur la base de clés déjà existantes.

Citons les types des clés utilisées pour effectuer l'échange présentée sur la figure 8.

- **SKEYID_a** est un argument de HASH-fonction et il est utilisé pour assurer l'authenticité d'un message.
- **SKEYID_d** sert à générer des clés des protocoles à l'aide d'une HASH-fonction.
- **SKEYID_e** est utilisée pour réaliser le chiffrement d'un message.

Il faut souligner que la formation du contexte des protocoles n'est pas bidirectionnelle comme cela a eu lieu dans le cas du contexte de contrôle. On réinitialise ce processus si Initialisateur et Partenaire veulent échanger leurs rôles.

Passons maintenant à l'examen d'une question qui peut être formulée comme suit : de quelle manière l'authenticité des IP-paquets est assurée dans Ipv6 ?

C'est le protocole AH, (Authentication Header) qui sert à résoudre cette problème (cf. figure 7). Outre cela, AH protège aussi les données des protocoles des niveaux supérieurs ainsi que des champs « non modifiables » des IP en-têtes ou modifiables mais de la manière prévue. A propos, la

quantité des champs modifiables est très limitée. Ce sont des champs Prio (Traffic Class), Flow Label et Hop Limit.

Sur la figure 9 le format d'en-tête AH est donné.

0	*8*	*16*	*24*	*32*
En-tête suivant *(Next Header)*	Lg. Extension *(Playload Len)*	Réservé *(Reserved)*		
Indice des paramètres de sécurité *(Security Paramaters Index : SPI)*				
Numéro de séquence *(Sequence Number Field)*				
Nombre de variable de mots de 32 bits *(Autentication Data, variable)*				

Figure 9 : Format d'en-tête du protocole AH

Si avant l'utilisation du protocole AH dans un mode transport un paquet de IPv6 a la structure donnée sur la figure 10 (a), après une telle utilisation la structure d'un paquet sera changée comme montré sur la figure 10 (b).

En-tête IPv6 standard	En-têtes supplémentaires	En-tête transport	Données

a) Paquet IPv6 avant l'utilisation du protocole AH dans une mode transport

En-tête IPv6 standard	En-têtes supplémentaires Partie 1	AH	En-têtes supplémentaires Partie 2	En-tête transport	Données

b) Paquet IPv6 après l'utilisation du protocole AH dans un mode transport

Figure 10 : Structure du paquet IPv6 sous le protocole AH utilisé dans un mode transport

Concernant le protocole ESP (Encapsulating Security Payload) il faut souligner qu'il assure trois types de sécurité :

- **Confidentialité** consistant tout d'abord à chiffrer le contenu d'un paquet IP et puis à assurer une protection partielle contre une analyse du trafic .
- **Intégrité** des paquets IP et l'authentification de la source des données.
- **Protection** contre la reproduction des paquets IP.

Quand on compare les protocoles AH et ESP on voit que la fonctionnalité de l'ESP est plus large que celle du AH grâce à un chiffrage qui manque dans AH et est présent dans ESP.

Le format de l'extension du protocole ESP est bien analysé dans [KENTW98]. Remarquons seulement que ce n'est pas une extension dans le sens propre du mot, mais c'est un « emballage » d'un contenu chiffrée plutôt.

Deux protocoles AH et ESP peuvent être combinés de manière différente. Si on est en mode transport, le protocole AH doit suivre celui ESP. Dans le cas où on est en mode tunnel ses deux protocoles sont appliqués aux paquets différents et la quantité des combinaisons possibles est assez grande.

2.2. Tests

L'histoire de la création et du développement de jeu des protocoles IPv4 nous montre qu'il y avaient plusieurs succès et certaines échecs dans ce processus. Pour ne pas passer deux fois par les mêmes difficultés il est raisonnable de prendre en compte des « leçons » quand il s'agit de développement du jeu des protocoles IPv6.

C'est l'utilisation des tests pour les protocoles IPv6 qui nous aidera au développement des technologies nouvelles de réseaux et surtout celles de IPv6.

Le développement du logiciel nouveau de système nécessite toujours de passer deux étapes :

1 étape – on utilise le test de conformité pour le logiciel créé.

2 étape - on utilise le test d'interopérabilité pour tester un logiciel créé par rapport au logiciel existant déjà.

Les problèmes liés au test de conformité ont été examinés dans le chapitre 1 du travail présent (cf. pp. 1-8). Examinons la deuxième étape.

Le test d'interopérabilité consiste à régler les problèmes ayant lieu pendant :

- L'utilisation des paramètres d'entrée communs.
- L'utilisation des bases de données communes.
- La production des données de sortie dans un format nécessaire.
- La réalisation de correspondance à la « politique » commune pour tous les logiciels développés et existants déjà.

Autrement dit, chaque protocole de IPv6 doit passer son test d'interopérabilité.

Parmi d'autres travaux concernés choisissons celui de [TAHI00] consacré au développement des tests pour certain jeu de protocoles IPv6. Pour les informations plus amples on peut s'adresser à http://www.tahi.org/release. Deux types de tests sont présentés : tests de conformité et tests d'interopérabilité. Examinons le dernier type présenté par un jeu de programmes *IPv6 Interoperability Test Tool* (ITT).

ITT est développé pour réaliser les test d'interopérabilité pour les protocoles IPv6. ITT donne la possibilité d'analyser les paquets reçus par TCPdump. Un certaine nombre des programmes sont destinés à cela. A savoir :

- ✓ ***flow*** donne la possibilité de créer un flux des paquets.
- ✓ ***padivide*** permet voir chaque paquet du flux navigué en créant un certain nombre de fichiers.
- ✓ ***packet.cgi*** donne la possibilité d'examiner des paquets d'un flux dans le mode dynamique sans création de fichiers.
- ✓ ***pareport.pl*** prépare un rapport sur un des paquets examinés.
- ✓ ***rpdump*** réalise un analyse de *routing protocol.*

Ce logiciel est accessible sur la page web citée ci-dessus : http://www.tahi.org/release.

RESUME

Le présent rapport est le résultat des recherches menées par l'auteur en qualité de professeur invité au sein du groupe RESEDAS, INRIA Lorraine, Nancy durant le mois du décembre 2000 sous la direction de responsable professeur André SCHAFF à qui l'auteur est très reconnaissant.

Dans le présent travail un résumé des travaux menés dans le domaine de génération et d'application des tests des protocoles a été fait.

L'accent a été mis sur les protocoles IPv6 et surtout les protocoles assurant la sécurité dans les réseaux fonctionnant sous IPv6.

Chaque de deux partie du rapport commence par les généralités (théorie, approches, méthodes) et se termine par l'analyse des algorithmes ou programmes concrets.

On suppose continuer ce travail par des recherches sur les tests d'interopérabilité pour les protocoles assurants la sécurité et la mobilité dans IPv6 et spécialement dans le cas multicast.

REFERENCES

[BOYD91]

Boyd S., Ural H. The synchronisation problem in protocol testing and its complexity. Information Processing Letters, vol. 40, pp. 131-136, novembre 1991.

[CAS94]

Castanet R. Test de protocoles de communication. Réseaux de Communication et Techniques Formelles. Paris – septembre 1994.

[CHOW78]

Chow T.S. Testing software design modelled by finit state mashines. IEEE Software Engineering, Vol. 4, No. 3. 1978.

[CIZ98]

Cizault G. IPv6. Editions O'Reilly, Paris, 1998.

[EST89]

Estelle A. Formal Description Technique Based on an Extended State Transition Model. ISO9074. 1989.

[HARK98]

Harkins D., Carrel D. The Internet Key Exchange (IKE). Internet-Draft, Jun 1998.

[ISO9646]

ISO/IEC 9646. Information technology, Open system interconnection, OSI conformance testing metodology and framework. 9646 Paris 1-5 .

[KENTS98]

Kent S., Atkinson R. Security Architecture for the Internet Protocol. Internet-Draft, May 1998.

[KENTW98]

Kent W., Atkinson R. IP Encapsulating Security Playload (ESP). Internet-Draft, May 1998.

[LOT89]

LOTOS. A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour. ISO8807. 1989.

[MAU98]

Maughan D. , Schertler M., Schneider M ., Turner J. Internet Security Association and Key Management Protocol (ISAKMP). Internet-Draft, Jul 1998.

[MEALY55]

Mealy G. E., A Method of Syntesizing Sequential Circuits. Bell System Technical Journal, 34, 1045-1079 (1955).

[NAITO81]

Naito S., Tsunoyoma. Fault detection for sequential machines by transition tours. IEEE Fault Tolerant Computing Conference. 1981.

[SAB88]

Sabnani K., Dahbura A. A protocol test generation procedure. Computer Networks and ISDN Systems, Vol. 15, pp. 285-297, 1988.

[SAR84]

Sarikaya B., Bohmann G. Synchronisation and specification issues protocol testing. IEEE Transactions on communications. Vol. COM-32, avril 1984.

[SDL88]

SDL Specification and Description Language, CCITT SG XI, Recomendation Z.100, 1988.

[TAHI00]

TAHI Project. Internet-Draft, February 2000.

[TASC98]

Laboratoire TASC. Université de Pau et des Pays de l'Adour. Génération de séquences de test : une étude comparative. Internet-Draft, October 29 1998.

Nancy, 22/12/2000.