SÉCURITÉ DES RÉSEAUX
RÉSEAUX PRIVÉS VIRTUELS (VPN) IPSEC

Jonathan CERTES, Benoît MORGAN

1 Environnement de travail

1.1 Travail dans une machine virtuelle

Télécharger Oracle VirtualBox :
`https://www.virtualbox.org/wiki/Downloads`
Installer VirtualBox sur sa machine personnelle :
`https://www.virtualbox.org/manual/UserManual.html#installation`
Télécharger la machine virtuelle fournie par l'enseignant.
Importer la machine virtuelle dans VirtualBox :
`https://www.virtualbox.org/manual/UserManual.html#ovf-import-appliance`
Démarrer la machine virtuelle :
`https://www.virtualbox.org/manual/UserManual.html#intro-starting`
Tout le TP sera réalisé dans la machine virtuelle.

1.2 Environnement

La machine virtuelle contient un conteneur `lxc` par machine du réseau virtuel pour ce TP (hôte ou routeur). Ces conteneurs ne possèdent pas d'interface graphique. Il s'agit de lancer un terminal dans la machine virtuelle et de s'y attacher autant de fois que nécessaire, pour tous les conteneurs.

Le lancement des conteneurs est automatisé par un script présent dans la machine virtuelle. Pour procéder à leur lancement, ouvrir un terminal et exécuter les commandes suivantes :

```
debian@myhostname:~$ cd tp-ipsec/
debian@myhostname:~/tp-ipsec$ ./start.sh
```

Le mot de passe pour le compte utilisateur est : `debian`.

1.3 S'associer aux conteneurs

Dans 4 terminaux, répéter ces actions pour les 4 conteneurs `host-a`, `router-a`, `router-b` et `host-b` :

```
debian@myhostname:~$ sudo xhost +
debian@myhostname:~$ sudo lxc-attach host-a
root@host-a:/$
```

1.4 Ouvrir Wireshark sur la machine virtuelle

La dernière étape de préparation de ce TP est le lancement de Wireshark sur un bridge (connexions virtuelles) de notre réseau. Il est nécessaire d'observer le trafic sur le bridge `br-router`.

Dans 1 terminal :

```
debian@myhostname:~$ sudo wireshark
```

Réduire le terminal **sans le fermer**. Sélectionner la capture sur le bridge `br-router`. Une fois la capture lancée, appliquer le filtre `tcp||icmp`.

2 Objectifs

L'objectif de ces séances est de comprendre en quoi le protocole IPsec permet de créer des réseaux privés virtuels (VPN) et d'assurer la sécurité des communications entre deux entités via un canal non-sécurisé.

C'est parti !

3 Introduction

IPsec (*Internet Protocol Security*), défini par l'IETF comme un cadre de standards ouverts pour assurer des communications privées et protégées sur des réseaux IP, par l'utilisation des services de sécurité cryptographiques, est un ensemble de protocoles utilisant des algorithmes permettant le transport de données sécurisées sur un réseau IP. IPsec se différencie des standards de sécurité antérieurs en n'étant pas limité à une seule méthode d'authentification ou d'algorithme et c'est la raison pour laquelle il est considéré comme un cadre de standards ouverts.

Source : `https://fr.wikipedia.org/wiki/Ipsec`

Le protocole IKE (*Internet Key Exchange*) est chargé de négocier la connexion. Avant qu'une transmission IPsec puisse être possible, IKE est utilisé pour authentifier les deux extrémités d'un tunnel sécurisé en échangeant des clés partagées. Ce protocole permet deux types d'authentifications, PSK (secret prépartagé ou secret partagé) pour la génération de clefs de sessions RSA ou à l'aide de certificats.

Source : `https://fr.wikipedia.org/wiki/Ipsec`

Le protocole AH (*Authentication Header*) fournit l'intégrité et l'authentification. AH authentifie les paquets en les signant, ce qui assure l'intégrité de l'information. Une signature unique est créée pour chaque paquet envoyé et empêche que l'information soit modifiée.

Source : `https://fr.wikipedia.org/wiki/Ipsec`

Le protocole ESP (*Encapsulating Security Payload*), en plus de l'authentification et l'intégrité, fournit également la confidentialité par l'entremise de la cryptographie.

Source : `https://fr.wikipedia.org/wiki/Ipsec`

Wireshark est un analyseur de paquets libre et gratuit. Il est utilisé dans le dépannage et l'analyse de réseaux informatiques, le développement de protocoles, l'éducation et la rétro-ingénierie.

Wireshark utilise la bibliothèque logicielle Qt pour l'implémentation de son interface utilisateur et pcap pour la capture des paquets ; il fonctionne sur de nombreux environnements compatibles UNIX comme GNU/Linux, FreeBSD, NetBSD, OpenBSD ou Mac OSX, mais également sur Microsoft Windows. Il existe aussi entre autres une version en ligne de commande nommé TShark. Ces programmes sont distribués gratuitement sous la licence GNU General Public License.

Wireshark reconnaît 1 515 protocoles.

Source : `https://fr.wikipedia.org/wiki/Wireshark`

4 Configuration réseau

Procéder au lancement des conteneurs comme décrit dans la section 1.2. L'architecture du réseau virtuel créée est représentée sur la figure 1. Chaque machine possède, sur ce réseau, une adresse IPv4 et un nom (`host-a`, `router-a`, `router-b` ou `host-b`).

Dans ce TP, nous incarnons l'administrateur système et réseau d'une entreprise. Cette entreprise est située sur deux sites géographiques distincts. Afin de réduire le coût des communications entre les sites, l'entreprise a opté pour l'utilisation du réseau Internet publique. L'objectif de l'administrateur système et réseau est donc de sécuriser les connexions entre les machines des deux sites géographiques.

À ces fins, le réseau de l'entreprise est séparé en trois zones :

— le sous-réseau **`host-a`**, qui contient des machines hébergeant des données sensibles/confidentielles sur le site géographique **A**.

— le sous-réseau **`host-b`**, qui contient des machines hébergeant des données sensibles/confidentielles sur le site géographique **B**.

— le réseau Internet sur lequel les deux sites sont joignables.

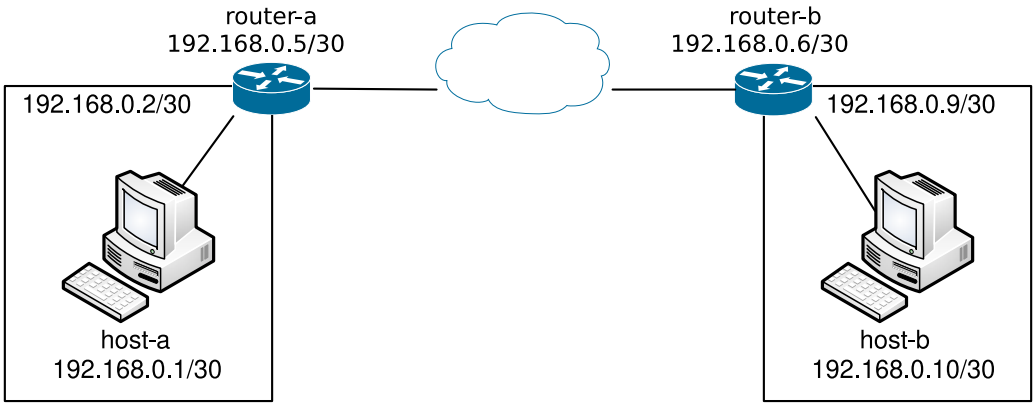

FIGURE 1 – Architecture du réseau

Les sous-réseaux `host-a` et `host-b` sont respectivement connectés par deux routeurs : **`router-a`** et **`router-b`**, qui sont nos *gateway* pour communiquer avec l'autre site géographique. Durant ce TP, nous allons donc configurer IPsec sur ces routeurs, à l'aide de la bibliothèque Libreswan et de la commande `ipsec`. Nous pouvons prendre le contrôle des routeurs avec les commandes suivantes :

```
debian@myhostname:~$ sudo lxc-attach router-a
```

```
debian@myhostname:~$ sudo lxc-attach router-b
```

Question 1 Vérifier le fonctionnement du réseau décrit sur la figure 1. Pour chacune des 2 machines qui simule les sous-réseaux, (`host-a`, et `host-b`), tenter d'envoyer une requête *ping*.
Vérifier qu'une réponse est reçue. Vérifier que les deux paquets transitent bien sur Internet à l'aide de *wireshark*.

```
debian@myhostname:~$ sudo lxc-attach host-a
root@host-a:/$ ping -c 1 host-b
```

Répéter cette opération depuis la machine `host-b` (vers `host-a`).
À chaque opération, redémarrer la capture sur *wireshark* pour bien observer les nouveaux paquets.

Contactez l'enseignant si vous n'observez pas de réponse sur une de ces communications !

5 Services ayant recours à IP

IPsec se différencie des standards de sécurité antérieurs en n'étant pas limité à une seule méthode d'authentification ou d'algorithme et c'est la raison pour laquelle il est considéré comme un cadre de standards ouverts. De plus, IPsec opère à la couche réseau (couche 3 du modèle OSI) contrairement aux standards antérieurs qui opéraient à la couche application (couche 7 du modèle OSI), ce qui le rend indépendant des applications, et veut dire que les utilisateurs n'ont pas besoin de configurer chaque application aux standards IPsec.

Réalisée dans le but de fonctionner avec le protocole IPv6, la suite de protocoles IPsec a été adaptée pour l'actuel protocole IPv4. Son objectif est d'authentifier et de chiffrer les données : le flux ne pourra être compréhensible que par le destinataire final (confidentialité) et la modification des données par des intermédiaires ne pourra être possible (intégrité). IPsec est souvent un composant de VPN, il est à l'origine de son aspect sécurité (canal sécurisé ou *tunneling*).

Source : `https://fr.wikipedia.org/wiki/Ipsec`

Un des avantages d'IPsec est qu'il permet donc de sécuriser toute communication ayant recours à IP. Pour tester la configuration du réseau privé virtuel, au moins un service ayant recours à IP doit être installé sur les sous-réseaux des deux sites : serveur HTTP ou FTP, SMTP, DNS, etc. La mise en place et la configuration d'un de ces services est fastidieuse et peu pertinente pour notre besoin. Nous allons donc simuler un serveur à l'aide de *netcat*.

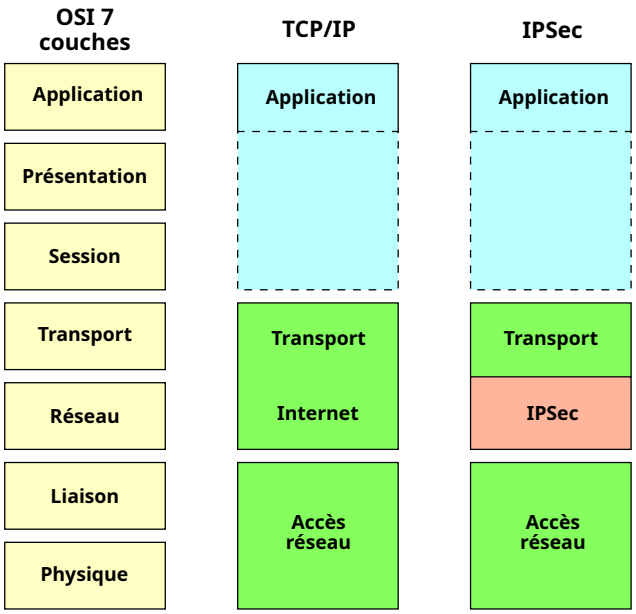

FIGURE 2 – Positionnement du protocole IPsec dans le modèle OSI

La commande suivante permet de lancer un serveur qui écoute (option `-l` de *listen*) sur un port (`-p`) en TCP, pour une adresse IP source (`-s`) donnée. L'option `-v` (*verbose*) indique d'afficher plus d'informations que normalement.

```
netcat -v -l -s <ip> -p <port>
```

Penser à utiliser l'adresse IP de l'interface qui nous intéresse. Par exemple, si nous choisissons `127.0.0.1`, l'écoute ne se fera que sur l'interface `lo` (en *loopback*).

De la même manière, la commande suivante permet de lancer un serveur qui écoute sur un port en UDP (option `-u`), toujours pour une adresse IP source et un port donnés.

```
netcat -v -l -u -s <ip> -p <port>
```

5.1 Tester et observer les paquets

Lorsque *netcat* est en écoute sur un port pour une adresse IP donnée, nous pouvons lui envoyer de la donnée à travers le réseau. Si le protocole utilisé est TCP, *netcat* se terminera après la fin de session.

Pour transmettre de la donnée à *netcat* depuis une autre machine et fermer la session, nous pouvons utiliser *netcat* et lui demander de quitter après envoi. La commande suivante établit une session TCP vers une adresse IP sur un port donné et la termine.

```
echo "quit" | netcat -q 0 <ip> <port>
```

Question 2 Simuler un serveur HTTP sur `host-a` à l'aide d'une écoute de *netcat* en TCP sur le port 80.

```
root@host-a:/$ netcat -v -l -s 192.168.0.1 -p 80
listening on [192.168.0.1] 80 ...
```

Question 3 Depuis `host-b`, simuler une communication avec le serveur HTTP de `host-a`, en TCP sur le port 80. Vérifier que *netcat* se termine sur `host-a`.

```
root@host-b:/$ echo "quit" | netcat -q 0 192.168.0.1 80
```

Question 4 À l'aide de *wireshark*, observer le bridge `br-router` et vérifier que la communication TCP a bien transité dessus. Appliquer le filtre `tcp.port == 80` pour ne visualiser que les paquets TCP que nous venons d'émettre.

5.2 Détails des paquets

À l'aide de *wireshark*, nous pouvons observer le détail des paquets. En l'occurrence, ce sont des trames Ethernet qui sont transmises entre les deux machines.

Chaque trame Ethernet débute par un entête, qui contient les adresses MAC de destination et de source dans ses deux premiers champs. L'entête est suivi par la charge utile à transmettre, qui contient elle-même les entêtes des protocoles de couche plus haute, par exemple le protocole Internet.

Dans ce TP, **nous nous intéressons à la sécurité de la charge utile** : c'est-à-dire de la couche réseau, couche 3 du modèle OSI (dans ce cas au protocole Internet), et de tout ce qu'elle transporte, couche 4 du modèle OSI (dans ce cas de l'entête et de la donnée suivant le protocole TCP). La figure 3 schématise la charge utile que nous souhaitons sécuriser.

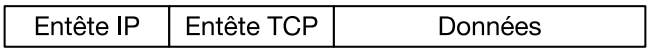

FIGURE 3 – Charge utile d'une trame Ethernet

Question 5 Utiliser *wireshark* pour observer le paquet dans lequel la donnée a été transmise, c'est-à-dire le paquet dont l'entête TCP contient le *flag* `PSH`.
Ce paquet doit ressembler à ceci (l'index du paquet, l'instant de transmission et le port source peuvent être différents) :

```
4   0.007s  192.168.0.10  192.168.0.1 TCP 72  40108 -> 80 [PSH, ACK] Seq=1 Ack=1
```

Question 6 Toujours à l'aide de *wireshark*, observer le détail du paquet et déterminer quelles sont les différentes parties de sa structure (entête Ethernet, entête IP, entête TCP et la donnée).

Voici un exemple d'identification, ici aussi le contenu peut être différent.
Entête Ethernet :

```
Ethernet II, Src: ca:fe:be:ef:00:06 (ca:fe:be:ef:00:06), Dst: ca:fe:be:ef:00:05 (ca:fe:be:ef:00:05)

0000   ca fe be ef 00 05 ca fe be ef 00 06 08 00         ..............
```

Entête IP :

```
Internet Protocol Version 4, Src: 192.168.0.10, Dst: 192.168.0.1

0000                                           45 10                 E.
0010   00 3a e0 d3 40 00 3f 06 d9 7e c0 a8 00 0a c0 a8   .:..@.?..~......
0020   00 01                                           ..
```

Entête TCP :

```
Transmission Control Protocol, Src Port: 40108, Dst Port: 80, Seq: 1, Ack: 1, Len: 6

0020         9c ac 00 50 32 72 15 c5 a9 30 0a 08 80 18   .....P2r...0....
0030   01 f6 81 88 00 00 01 01 08 0a 66 c5 79 e2 39 95   ..........f.y.9.
0040   21 d3                                           !.
```

Donnée :

```
0040         71 75 69 74 0a                                  quit.
```

Ici, notre donnée est bien égale à ce que nous avons transmis via *netcat*, à savoir le mot-clef `quit` suivi d'un saut de ligne (`\n`). Ceci peut être vérifié à l'aide d'un éditeur hexadécimal :

```
debian@myhostname:~$ echo "quit" | xxd
00000000: 7175 6974 0a                             quit.
```

Dans la suite de ce TP, nous allons configurer IPsec pour ajouter un entête d'authentification (AH) et encapsuler la charge utile (ESP). À chaque nouvelle configuration, nous allons utiliser *wireshark* pour observer la structure du paquet et constater les changements.

6 Sécurité de la couche transport

Notre objectif est de sécuriser les connexions entre les machines de deux sites géographiques. Dans un premier temps, nous nous intéressons à la communication de seulement deux machines : la *gateway* du site A (`router-a`) et la *gateway* du site B (`router-b`). La figure 4 représente le réseau tel qu'il nous intéresse dans ce cas.

FIGURE 4 – Architecture simplifiée du réseau

IPsec utilise trois protocoles pour fournir la sécurité de la couche transport :
— **Internet Key Exchange** (**IKE**) fournit le mécanisme d'échange de clés.
— **Authentication Header** (**AH**) fournit l'intégrité et l'authentification de l'origine des données sur l'ensemble du paquet.
— **Encapsulating Security Payload** (**ESP**) fournit la confidentialité à l'aide du chiffrement ainsi qu'une authentification des données, à l'exception de l'entête IP.

Ces protocoles peuvent être combinés ou utilisés seuls. Par exemple, les protocoles AH et ESP peuvent être utilisés conjointement afin de créer une communication sécurisée entre différentes entités. Le protocole IKE est utilisé pour simplifier la distribution des clés servant au chiffrement et à l'authentification.

Afin de sécuriser une communication, il est impératif de connaître quelles sont les adresses IP concernées, dans quel sens la communication doit être sécurisée et avec quel algorithme de signature/chiffrement. IPsec appelle cela une ***Security Association*** (**SA**). Il est important de noter que chaque SA est unidirectionnelle et que pour une communication bidirectionnelle, il faut spécifier deux SA.

Ensuite, il faut établir quel(s) entête(s) nous souhaitons ajouter/encapsuler, c'est-à-dire dans quels cas (pour quel protocole, quelles adresses IP, etc.) il faut utiliser les SA pour sécuriser les messages. IPsec appelle cela la ***Security Security Policy*** (**SP**).

En résumé : une *Security Security Policy* définit quand et quelles propriétés de sécurité nous souhaitons obtenir ; une *Security Association* définit avec qui et comment nous souhaitons les obtenir.

Afin de configurer IPsec, nous utilisons le paquet Debian `libreswan`, qui contient tout le nécessaire. Celui-ci inclut le programme `ipsec` qui manipule le démon *Internet Key Exchange* chargé de négocier les SA. Un fichier de configuration d'IPsec est disponible sur les deux routeurs : `/etc/ipsec.conf`. Les secrets partagés entre les deux routeurs sont également disponibles dans un fichier : `/etc/ipsec.secrets`.
La documentation du programme `ipsec` est disponible dans son manuel :

```
debian@myhostname:~$ man ipsec
```

La documentation du fichier de configuration `/etc/ipsec.conf` est disponible dans son manuel :

```
debian@myhostname:~$ man ipsec.conf
```

6.1 Phase 1 : échange des clés

Toute la sécurité fournie par IPsec tient du secret que sont les clés. Dès qu'un attaquant les obtient, il peut non seulement usurper l'identité d'un des hôtes mais également déchiffrer les données capturées lors d'une communication entre deux hôtes légitimes. L'idée est donc d'utiliser des SA à durée de vie limitée et de négocier de nouvelles SA après expiration.

Dans un premier temps, nous allons configurer le protocole *Internet Key Exchange* (IKE) qui permet de négocier dynamiquement les SA à partir d'un secret partagé, appelé *pre-shared key* (PSK). L'avantage d'une négociation dynamique des SA est qu'une clé de session, différente du secret partagé, est utilisée pour signer/chiffrer les paquets. Ainsi, si un adversaire s'empare du secret partagé, il ne pourra pas déchiffrer les communications s'il n'a pas intercepté la négociation.

Cependant, il pourra toujours usurper l'identité d'un hôte en demandant une nouvelle négociation et ainsi réaliser une attaque de l'homme du milieu.

IPsec appelle l'échange des clés : la **phase 1** ; et la communication authentifiée/confidentielle : la **phase 2**.

Question 7 La première étape consiste donc à générer un secret partagé. Nous utilisons le générateur de nombres aléatoires `/dev/random` pour générer une PSK de 64 octets.
Prendre le contrôle du `router-a` et générer notre PSK.

```
root@router-a:/$ dd if=/dev/random count=1 bs=64 2> /dev/null | base64 --wrap=88
<notre_psk>
```

Question 8 Ensuite, il faut indiquer à IPsec d'utiliser ce secret partagé lors d'une communication entre les deux routeurs.
Sur le `router-a`, observer le contenu du fichier `/etc/ipsec.secrets`. En particulier, observer la dernière ligne.

```
root@router-a:/$ tail -n 1 /etc/ipsec.secrets
include /etc/ipsec.d/*.secrets
```

Pour indiquer à nos deux routeurs d'utiliser le secret partagé, nous pouvons écrire un fichier avec l'extension `.secrets` dans le dossier `/etc/ipsec.d/`. Celui-ci sera inclus dans `/etc/ipsec.secrets` au démarrage d'IPsec.

Question 9 Notre secret partagé sera utilisé pour réaliser un échange de clés entre les deux hôtes accessibles aux adresses IP `192.168.0.5` et `192.168.0.6`.
Sur les deux routeurs, écrire un fichier `/etc/ipsec.d/tp.secrets` (de manière identique) pour indiquer le PSK à utiliser.

```
root@router-a:/$ nano /etc/ipsec.d/tp.secrets
root@router-a:/$ cat  /etc/ipsec.d/tp.secrets
192.168.0.5 192.168.0.6 : PSK "<notre_psk>"
```

Désormais, nous devons demander à IPsec d'utiliser le protocole IKE pour réaliser un échange de clés.
De la même manière que pour le ficher `/etc/ipsec.secrets`, une inclusion de plusieurs fichiers est réalisée dans `/etc/ipsec.conf`. Nous pouvons donc configurer IPsec dans un fichier de configuration dédié.

Question 10 Vérifier que les fichiers `*.conf` contenus dans le dossier `/etc/ipsec.d/` sont bien inclus dans le fichier `/etc/ipsec.conf`.

```
root@router-a:/$ grep "^include" /etc/ipsec.conf
```

Question 11 **Sur les deux routeurs**, écrire un fichier de configuration `/etc/ipsec.d/tp.conf` (de manière identique) pour demander à IPsec de sécuriser les connexions entre eux. Demander à IPsec d'automatiser l'échange de clés à l'ajout d'une nouvelle connexion (étape manuelle). L'authentification se fait par secret partagé.
Attention ! Dans le fichier, l'indentation est importante.

```
root@router-a:/$ nano /etc/ipsec.d/tp.conf
root@router-a:/$ cat  /etc/ipsec.d/tp.conf
conn myvpn
  left=192.168.0.5
  right=192.168.0.6
  auto=add
  authby=secret
```

À répéter sur le `router-b`.

Question 12 **Sur les deux routeurs**, redémarrer le démon IPsec et ajouter la connexion que nous venons de définir.

```
root@router-a:/$ ipsec setup restart
Redirecting to: systemctl restart ipsec.service
root@router-a:/$ ipsec auto --add myvpn
002 "myvpn": terminating SAs using this connection
002 "myvpn": added IKEv2 connection
```

À répéter sur le `router-b`.

Nous allons désormais procéder à l'échange de clés. IPsec appelle cela la **phase 1**. La phase 1 possère deux modes de fonctionnement : le mode *main* ou le mode *aggressive*. Nous détaillons ici le mode *main*.

IKE utilise l'échange de clés Diffie-Hellman [1] pour mettre en place un secret partagé d'où les clefs de chiffrement sont dérivées. Ceci signifie que les deux parties doivent d'abord s'entendre sur un choix d'algorithmes de chiffrement et d'authentification. Ensuite les deux parties procèdent à un échange de clés. Finalement, une vérification des identités est réalisée.

Chaque partie doit donc transmettre 3 messages à l'autre partie. Soit un total de 6 messages. La figure 5 résume le fonctionnement de l'échange de clés selon le protocole IKE.

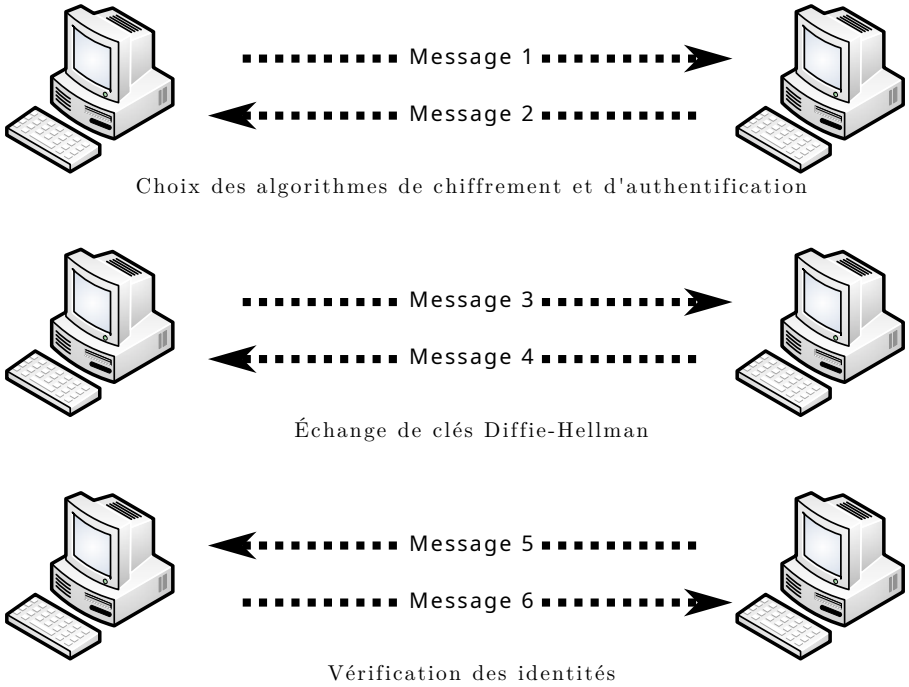

FIGURE 5 – Protocole *Internet Key Exchange* (IKE)

Lorsque la phase 1 se déroule en mode *aggressive*, la vérification des identités est omise. Seulement 4 messages sont alors nécessaires à l'échange de clés. Dans cette section, nous utilisons le mode *main* de la phase 1.

Question 13 À l'aide de *wireshark*, observer le bridge `br-router` et appliquer le filtre `isakmp`.

Question 14 Sur le `router-a`, démarrer la connexion que nous avons définie pour IPsec. Un échange de clés est alors initié par le `router-a`.

```
root@router-a:/$ ipsec auto --up myvpn
```

Si tout se passe bien, le `router-b` est authentifié lors de la phase 1. Par défaut, l'algorithme de chiffrement utilisé est `AES_GCM` avec une clé de 256 bits et l'authentification est réalisée à l'aide d'un calcul de `HMAC` via la somme de hachage `SHA2`.

1. https://fr.wikipedia.org/wiki/%C3%89change_de_cl%C3%A9s_Diffie-Hellman

```
181 "myvpn" #1: initiating IKEv2 connection
181 "myvpn" #1: sent IKE_SA_INIT request to 192.168.0.6:500
182 "myvpn" #1: sent IKE_AUTH request {cipher=AES_GCM_16_256 integ=n/a prf=HMAC_SHA2_512 group=MODP2048}
003 "myvpn" #1: initiator established IKE SA; authenticated peer using authby=secret and ID_IPV4_ADDR '192.168.0.6'
```

À l'aide de *wireshark*, nous pouvons observer 4 messages. Ces messages sont le résultat de deux échanges entre les routeurs pour, respectivement, le choix des algorithmes et l'échange de clés. Les deux derniers messages de la phase 1 ne sont pas transmis car la connexion que nous avons définie pour IPsec sur `router-b` n'a pas encore été démarrée.

```
No. Time    Source       Destination  Protocol Length Info
1   0.000   192.168.0.5 192.168.0.6  ISAKMP   870    IKE_SA_INIT MID=00 Initiator Request
2   0.091   192.168.0.6 192.168.0.5  ISAKMP   482    IKE_SA_INIT MID=00 Responder Response
3   0.094   192.168.0.5 192.168.0.6  ISAKMP   427    IKE_AUTH MID=01 Initiator Request
4   0.242   192.168.0.6 192.168.0.5  ISAKMP   267    IKE_AUTH MID=01 Responder Response
```

Question 15 À l'aide de *wireshark*, observer le détail du premier des 4 paquets. En particulier, observer la section *Internet Security Association and Key Management Protocol* (`isakmp`) et les champs `Initiator SPI` et `Responder SPI`. Quelles valeurs transmet le `router-a` pour ces deux champs ?

Question 16 Toujours dans la section *Internet Security Association and Key Management Protocol*, observer le champs `Payload: Security Association`. Combien de champs `Payload: Proposal` sont présent à l'intérieur ? Selon vous, que décrivent ces champs ?

Dans le protocole IKE, le premier échange consiste à choisir les algorithmes de chiffrement et d'authentification. Ici, c'est le `router-a` qui initie la demande, il est donc l'*Initiator* et le `router-b` est le *Responder*.

Pour authentifier les messages lors de la phase 2, IPsec stipule l'utilisation d'un paramètre de sécurité qui est utilisé dans le calcul du test d'intégrité. Ce paramètre, c'est le **SPI** : ***Security Parameter Index***. Il est choisi lors de la phase 1 en même temps que les algorithmes de chiffrement et d'authentification. C'est un nombre aléatoire que l'*Initiator* et le *Responder* doivent choisir. Lors du premier message envoyé par le `router-a`, celui-ci a choisi son SPI et a laissé un champs vide (avec une valeur nulle) pour le SPI du *Responder*.

En ce qui concerne le choix des algorithmes, l'*Initiator* supporte plusieurs solutions. Il propose alors au *Responder* une liste d'algorithmes pour définir la SA. Parmi cette liste, le *Responder* doit alors choisir 3 algorithmes : un pour le chiffrement, un pour l'authentification et un pour l'échange de clés (utilisé pour la suite du protocole IKE).
Voici un exemple de contenu du premier des 4 paquets. Dans cet exemple, le `router-a` a choisi comme SPI la valeur `0d5bd4cf113f54a8` et propose 4 solutions pour définir la SA.

```
Internet Security Association and Key Management Protocol
    Initiator SPI: 0d5bd4cf113f54a8
    Responder SPI: 0000000000000000
[...]
    Payload: Security Association (33)
[...]
        Payload: Proposal (2) # 1
        Payload: Proposal (2) # 2
        Payload: Proposal (2) # 3
        Payload: Proposal (2) # 4
```

Question 17 Désormais, à l'aide de *wireshark*, observer le détail du second des 4 paquets. C'est-à-dire la réponse du `router-b`. En particulier, observer la section *Internet Security Association and Key Management Protocol* (`isakmp`) et les champs `Initiator SPI` et `Responder SPI`. Quelles valeurs transmet le `router-b` pour ces deux champs ?

Question 18 Toujours dans la section *Internet Security Association and Key Management Protocol*, observer le champs `Payload: Security Association`. Combien de champs `Payload: Proposal` sont présent à l'intérieur ? Que peut on observer dans le contenu ?

Le second paquet est la réponse du `router-b` au choix des algorithmes de chiffrement et d'authentification. Le *Responder* complète le champs `Responder SPI` tout en laissant intact le champs `Initiator SPI` (sa valeur est identique à celle présente dans le premier paquet, envoyé par le `router-a`). Maintenant, les deux parties ont choisi leur SPI.

En ce qui concerne le choix des algorithmes, le *Responder* choisi parmi les propositions les algorithmes qui lui conviennent. Il n'y a plus qu'une seule proposition. Cette proposition contient la définition de 3 transformations : une pour chaque algorithme choisi pour respectivement le chiffrement, l'authentification et l'échange de clés.
Voici un exemple de contenu du second des 4 paquets. Dans cet exemple, le `router-b` a choisi comme SPI la valeur `e10955c0b9eef7ef` et les algorithmes suivants :

— AES-GCM pour le chiffrement,
— un calcul de HMAC avec l'algorithme de hachage SHA2 pour l'authentification,
— le groupe des entiers modulo p sur 2048 bits pour l'échange de clés.

```
Internet Security Association and Key Management Protocol
    Initiator SPI: 0d5bd4cf113f54a8
    Responder SPI: e10955c0b9eef7ef
[...]
    Payload: Security Association (33)
[...]
        Payload: Proposal (2) # 1
[...]
            Payload: Transform (3)
                Transform ID (ENCR): AES-GCM with a 16 octet ICV (20)
            Payload: Transform (3)
                Transform ID (PRF): PRF_HMAC_SHA2_512 (7)
            Payload: Transform (3)
                Transform ID (D-H): 2048 bit MODP group (14)
```

Question 19 Désormais, à l'aide de *wireshark*, observer le détail des troisième et quatrième paquets. C'est-à-dire l'échange de clés entre le `router-a` et le `router-b`. En particulier, observer la section *Internet Security Association and Key Management Protocol* et le champs `Payload`.
Que peut on conclure de cette information ?

Les troisième et quatrième paquets représentent l'échange de clés. L'information est chiffrée et n'est pas accessible par un adversaire qui observe le trafic et ne possède pas la PSK. Nous ne pouvons rien conclure de cette information, si ce n'est qu'après cette étape, le `router-a` et le `router-b` peuvent s'authentifier et communiquer de manière chiffrée.

Question 20 Les deux derniers messages du protocole IKE n'ont pas encore été échangés, la raison est que la connexion que nous avons définie pour IPsec sur le `router-b` n'a pas encore été démarrée.
Prendre le contrôle du `router-b` et démarrer la connexion IPsec.

```
root@router-b:/$ ipsec auto --up myvpn
```

Si tout se passe bien, la vérification des identité est réalisée. Cela requiert l'envoi de données chiffrées que nous ne pouvons pas accéder via *wireshark*. Nous pouvons seulement observer leur transmission.

Question 21 À l'aide de *wireshark*, observer les deux derniers messages du protocole IKE. Notons que si la donné est trop lourde, les messages peuvent être fragmentés en plusieurs paquets. L'échange doit ressembler à ceci :

```
No. Time   Source      Destination  Protocol Length Info
5   6.627  192.168.0.6 192.168.0.5  ISAKMP   581    CREATE_CHILD_SA MID=00 Responder Request (fragment 1/2)
6   6.627  192.168.0.6 192.168.0.5  ISAKMP   201    CREATE_CHILD_SA MID=00 Responder Request (fragment 2/2)
7   6.660  192.168.0.5 192.168.0.6  ISAKMP   491    CREATE_CHILD_SA MID=00 Initiator Response
```

Libreswan possède une fonctionnalité qui permet d'automatiser l'échange de clés et de négocier toutes les SA au démarrage. Ainsi, IPsec sera toujours utilisé lorsque la configuration le stipule.

Question 22 **Sur les deux routeurs**, modifier le fichier de configuration `/etc/ipsec.d/tp.conf` (de manière identique) et remplacer l'option `auto=add` par `auto=start`.

```
root@router-a:/$ nano /etc/ipsec.d/tp.conf
root@router-a:/$ cat  /etc/ipsec.d/tp.conf
conn myvpn
  left=192.168.0.5
  right=192.168.0.6
  auto=start          # <- modifier ici
  authby=secret
```

À répéter sur le `router-b`.

Question 23 **Sur les deux routeurs**, redémarrer le démon IPsec et vérifier à l'aide de *wireshark* qu'un échange de clés est alors initié.

```
root@router-a:/$ ipsec setup restart
Redirecting to: systemctl restart ipsec.service
```

À répéter sur le `router-b`.

À ce stade de la configuration, les routeurs peuvent communiquer de manière authentifiée et chiffrée. Par défaut, *Libreswan* choisit de chiffrer les communications avec ESP en mode tunnel. Nous ne pouvons pas observer le fonctionnement d'IPsec mais pouvons vérifier que la communication fonctionne.

Question 24 Simuler un serveur HTTP sur le `router-a` à l'aide d'une écoute de *netcat* en TCP sur le port 80.

```
root@router-a:/$ netcat -v -l -s 192.168.0.5 -p 80
listening on [192.168.0.5] 80 ...
```

Question 25 Depuis le `router-b`, simuler une communication avec le serveur HTTP du `router-a` en TCP sur le port 80. Vérifier que *netcat* se termine sur le `router-a`.

```
root@router-b:/$ echo "quit" | netcat -q 0 192.168.0.5 80
```

Question 26 À l'aide de *wireshark*, observer le bridge `br-router` et vérifier que la communication TCP a bien transité dessus. Appliquer le filtre `esp` pour ne visualiser que les paquets chiffrés que nous venons d'émettre.

6.2 Phase 2 : authentification avec AH

Se baser sur une adresse IP n'est pas suffisant pour garantir l'identité d'un hôte. En effet, l'attaque *IP Spoofing* consiste à usurper l'adresse IP d'un hôte existant et potentiellement de confiance. Bien que des règles de pare-feu permettent d'éviter dans une certaine mesure cette attaque entre différents réseaux, ces règles n'y peuvent rien lors d'une connexion point à point. IPsec répond à cette problématique en fournissant l'intégrité et l'authentification de l'origine des données sur l'ensemble du paquet. Cette tâche est réalisée avec l'ajout d'un entête **Authentication Header** (**AH**).

L'objectif de cette section est de modifier la configuration d'IPsec afin d'observer l'entête AH utilisé pour vérifier l'authenticité des paquets IP. En mode transport, l'entête AH est inséré dans le paquet IP originel, entre l'entête IP et l'entête de la couche transport (TCP ou UDP). La figure 6 représente cet ajout.

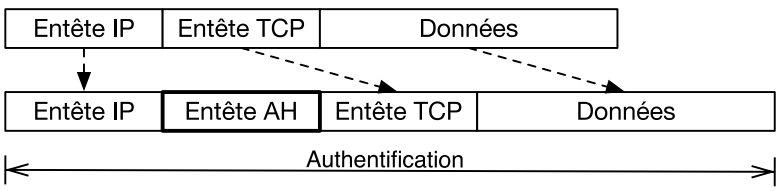

FIGURE 6 – Paquet IP avec AH en mode transport

Notons que si IPsec permet d'utiliser l'authentification (AH) sans chiffrement (ESP), il est préférable ne pas utiliser cette fonctionnalité. La configuration décrite dans cette section est donc à but éducatif. En effet, nous souhaitons observer le fonctionnement de l'authentification dans d'IPsec. Aujourd'hui, la puissance de calcul des machines et les débits obtenus sur les réseaux sont tels que la réduction des performances engendrée par des tâches de chiffrement/déchiffrement reste acceptable.

Question 27 **Sur les deux routeurs**, modifier le fichier de configuration `/etc/ipsec.d/tp.conf` (de manière identique) et ajouter deux options pour stipuler que nous souhaitons utiliser AH en mode transport :

- `phase2=ah`
- `type=transport`

```
root@router-a:/$ nano /etc/ipsec.d/tp.conf
root@router-a:/$ cat  /etc/ipsec.d/tp.conf
conn myvpn
  left=192.168.0.5
  right=192.168.0.6
  auto=start
  authby=secret
  phase2=ah           # <- ajouter ici
  type=transport      # <- ajouter ici
```

À répéter sur le `router-b`.

Question 28 À l'aide de *wireshark*, observer le bridge `br-router` et appliquer le filtre `isakmp`.

Question 29 **Sur les deux routeurs**, redémarrer le démon IPsec et observer à l'aide de *wireshark* les paramètres SPI choisis lors du dernier échange de clés. Se souvenir de ces valeurs.

```
root@router-a:/$ ipsec setup restart
Redirecting to: systemctl restart ipsec.service
```

À répéter sur le `router-b`.
Les paramètres SPI sont présents dans la section *Internet Security Association and Key Management Protocol* du dernier paquet. Par exemple :

```
Internet Security Association and Key Management Protocol
    Initiator SPI: 0d5bd4cf113f54a8
    Responder SPI: e10955c0b9eef7ef
```

Question 30 Simuler un serveur HTTP sur le `router-a` à l'aide d'une écoute de *netcat* en TCP sur le port 80.

```
root@router-a:/$ netcat -v -l -s 192.168.0.5 -p 80
listening on [192.168.0.5] 80 ...
```

Question 31 Depuis le `router-b`, simuler une communication avec le serveur HTTP du `router-a` en TCP sur le port 80. Vérifier que *netcat* se termine sur le `router-a`.

```
root@router-b:/$ echo "quit" | netcat -q 0 192.168.0.5 80
```

Question 32 À l'aide de *wireshark*, observer le bridge `br-router` et vérifier que la communication TCP a bien transité dessus. Appliquer le filtre `tcp` pour ne visualiser que les paquets que nous venons d'émettre.

Question 33 Vérifier que les paquets émis par les deux routeurs sont modifiés pour contenir un entête AH comme décrit sur la figure 6.
Quels sont les 3 champs relatifs à AH présents dans cet entête ?

Nous pouvons observer la présence de l'entête AH entre l'entête IP et l'entête TCP, exactement comme décrit par la figure 6. Voici un exemple de trame Ethernet telle qu'observée par *Wireshark* (les valeurs sont différentes à chaque émission) :

```
Ethernet II, Src: ca:fe:be:ef:00:06 (ca:fe:be:ef:00:06), Dst: ca:fe:be:ef:00:05 (ca:fe:be:ef:00:05)
Internet Protocol Version 4, Src: 192.168.0.6, Dst: 192.168.0.5
Authentication Header
    Next header: TCP (6)
    Length: 9 (44 bytes)
    Reserved: 0000
    AH SPI: 0x048a7456
    AH Sequence: 1
    AH ICV: efa5c362341cfb4d3cf7bf9164abb4e877d5bd372d6beb5e33b93061802005d9
Transmission Control Protocol, Src Port: 37922, Dst Port: 80, Seq: 0, Len: 0
```

L'entête AH contient 3 informations importantes :

- `AH SPI` : le paramètre de sécurité (*Security Parameter Index*).
- `AH Sequence` : l'index du paquet dans la séquence d'authentification, cet index est incrémenté à chaque émission de paquet.
- `AH ICV` : la valeur du test d'intégrité (*Integrity Check Value*), le résultat du calcul. Dans notre cas, c'est le `HMAC` avec la fonction de hachage `SHA2`, calculé sur toute la charge utile (voir la figure 6) et la clé de session négociée avec IKE.

Question 34 Dans tous les paquets TCP contenant l'entête AH, combien de valeurs différentes peut-on observer pour le paramètre SPI ? Le paramètre SPI présent dans l'entête AH est il identique à l'un des deux paramètres SPI présents dans l'échange de clés IKE ?
Que peut-on en conclure sur l'utilisation par IPsec du paramètre SPI ?
Établir une seconde communication TCP pour confirmer votre intuition.

Les protocoles IKE et AH requièrent tous deux un paramètre SPI pour chaque hôte authentifié. Les SPI du protocoles IKE ne sont donc pas liés aux SPI du protocole AH.

- les SPI du protocole IKE sont choisis par les hôtes lors des messages 1 et 2 du protocole (voir la figure 5) ; ils sont utilisés lors des messages 3, 4, 5 et 6 de ce même protocole.
- les SPI du protocole AH sont choisis par les hôtes une fois l'échange de clés terminé ; ils sont utilisés lors de tous les échanges jusqu'à la négociation de nouvelles clés de session.

Question 35 Sur le `router-b`, redémarrer le démon IPsec. Réaliser ensuite une nouvelle communication TCP entre les deux routeurs. Observer à l'aide de *wireshark* la modification des paramètres SPI.

```
root@router-b:/$ ipsec setup restart
Redirecting to: systemctl restart ipsec.service
```

```
root@router-a:/$ netcat -v -l -s 192.168.0.5 -p 80
listening on [192.168.0.5] 80 ...
```

```
root@router-b:/$ echo "quit" | netcat -q 0 192.168.0.5 80
```

6.2.1 Authenticité

Une des propriétés de sécurité garanties par AH est que les hôtes qui communiquent sont authentifiés. Cette authentification est réalisée lors de la phase 1, par le choix d'une clé de session. Elle n'est donc possible que si les deux routeurs partagent un secret.

Question 36 À l'aide de *wireshark*, lancer une nouvelle capture sur le bridge `br-router`. Appliquer le filtre `isakmp || tcp` pour ne visualiser que les paquets issus du protocole IKE ou d'un échange TCP.

Question 37 Sur le `router-b`, modifier le fichier `/etc/ipsec.d/tp.secrets` de telle sorte que les secrets connus entre les deux routeurs soient différents. Ne pas modifier le secret sur le `router-a`.

Question 38 Sur le `router-b`, redémarrer le démon IPsec et observer à l'aide de *wireshark* l'échange de clés.
Les messages 5 et 6 du protocole IKE (voir la figure 5) sont ils présents ?

Question 39 Tenter de nouveau d'établir une communication entre le `router-a` et le `router-b`. Que se passe-t-il si `router-a` écoute via le protocole TCP et que `router-b` initie une connexion ?
Confirmer votre intuition en observant les paquets sur `br-router` à l'aide de *Wireshark*. Quelle est la réaction du `router-a` ?

```
root@router-a:/$ netcat -v -l -s 192.168.0.5 -p 80
listening on [192.168.0.5] 80 ...
```

```
root@router-b:/$ echo "quit" | netcat -q 0 192.168.0.5 80
```

Question 40 Que se passe-t-il si `router-b` écoute via le protocole TCP et que `router-a` initie une connexion ? Quelle est la réaction du `router-a` ? Pourquoi ?
Confirmer votre intuition en observant les paquets sur `br-router` à l'aide de *Wireshark*.

```
root@router-b:/$ netcat -v -l -s 192.168.0.6 -p 80
listening on [192.168.0.6] 80 ...
```

```
root@router-a:/$ echo "quit" | netcat -q 0 192.168.0.6 80
```

Dans le cas où le `router-b` ne possède pas le secret attendu par le `router-a`, la négociation des clés de session à l'aide du protocole IKE ne peut pas aboutir. Nous voyons seulement transiter les 4 premiers messages du protocole et ce, à chaque tentative.

Dans le cas où le `router-b` initie une connexion, alors le paquet transmis ne contient pas d'entête AH. En effet, nous venons de redémarrer le démon IPsec et l'échange de clé n'a pas fonctionné ; il n'est donc pas possible de s'authentifier.

Le `router-a`, auquel nous n'avons pas redémarré le démon IPsec, considère qu'il se trouve face à un hôte malveillant ; pour lui, un échange de clé avait précédemment abouti. Dans ce cas, il abandonne les paquets TCP reçus depuis cet hôte et demande une nouvelle authentification via le protocole IKE.

Le comportement d'IPsec sur le `router-a` est le même que celui d'un pare-feu qui abandonne (`DROP`) les paquets qui sont destinés à la machine sur laquelle il fonctionne (`INPUT`). Ici, c'est la cryptographie qui est utilisée pour authentifier les hôtes légitimes.

Dans le cas où le `router-a` initie une connexion, alors aucun paquet TCP n'est transmis. Les paquets TCP à émettre sont abandonnés : à la place, une demande de nouvelle authentification via le protocole IKE est transmise. Ici aussi, le `router-a`, auquel nous n'avons pas redémarré le démon IPsec, considère qu'il se trouve face à un hôte malveillant.

Le comportement d'IPsec sur le `router-a` est le même que celui d'un pare-feu qui abandonne les paquets que la machine émet (`OUTPUT`).

6.2.2 Intégrité

Une autre propriété de sécurité garantie par AH est que les paquets transmis sont abandonnés s'ils ont été modifiés. En plus de l'authenticité, AH vérifie donc l'intégrité des paquets.

Lors de l'émission du paquet, IPsec calcule une somme de contrôle sur le contenu du paquet et la clé de session. Dans notre cas, c'est un `HMAC` avec la fonction de hachage `SHA2` qui est réalisé. Le résultat du calcul est placé dans l'entête AH : c'est le champs `AH ICV` (*Integrity Check Value*).

Lors de la réception du paquet, IPsec calcule également une somme de contrôle sur le contenu du paquet et la clé de session. Il compare ensuite le résultat obtenu avec la valeur placée dans dans l'entête AH qu'il a reçu.

- si le résultat est identique, alors le paquet est considéré authentifié et intègre. IPsec accepte ce paquet.
- si le résultat est différent, alors le paquet est considéré modifié (le contenu du paquet a changé et c'est la raison d'un résultat différent) ou émis par un hôte malveillant (l'émetteur ne connait pas la clé de session). IPsec abandonne ce paquet (`DROP`).

Nous pouvons observer le résultat du calcul du test d'intégrité, à chaque paquet transmis, dans le champs `AH ICV` de l'entête AH.

Question 41 À l'aide de *wireshark*, lancer une nouvelle capture sur le bridge `br-router`. Appliquer le filtre `isakmp || tcp` pour ne visualiser que les paquets issus du protocole IKE ou d'un échange TCP.

Question 42 Sur le `router-b`, modifier le fichier `/etc/ipsec.d/tp.secrets` afin de rétablir le secret partagé avec le `router-a`. Après cette tâche, les deux secrets doivent être identiques sur les deux routeurs.

Question 43 Sur le `router-b`, redémarrer le démon IPsec et observer à l'aide de *wireshark* l'échange de clés.

Question 44 Simuler un serveur HTTP sur le `router-a` à l'aide d'une écoute de *netcat* en TCP sur le port 80.

```
root@router-a:/$ netcat -v -l -s 192.168.0.5 -p 80
listening on [192.168.0.5] 80 ...
```

Question 45 Depuis le `router-b`, simuler une communication avec le serveur HTTP du `router-a` en TCP sur le port 80. Vérifier que *netcat* se termine sur le `router-a`.

```
root@router-b:/$ echo "quit" | netcat -q 0 192.168.0.5 80
```

Question 46 À l'aide de *wireshark*, observer le bridge `br-router` et vérifier que la communication TCP a bien transité dessus. Pour chaque paquet TCP différent, vérifier que le résultat du calcul dans le champs `AH ICV` de l'entête AH est différent.

L'outil `nping` (distribué avec `nmap`) permet de forger facilement des paquets IP et de choisir le contenu avec précision. Nous pouvons utiliser `nping` pour transmettre un paquet TCP/IP et choisir son *identification* IP et ses paramètres TCP : ports source et destination, numéro de séquence et *flags*.

De ce fait, nous pouvons transmettre plusieurs fois des paquets identiques et observer le comportement d'IPsec sur l'entête AH. Plus particulièrement, nous pouvons observer le résultat du calcul du test d'intégrité à chaque retransmission.

Question 47 À l'aide de *wireshark*, observer le bridge `br-router` et appliquer le filtre suivant :

```
(isakmp||tcp) && ip.src == 192.168.0.6
```

Question 48 Depuis le `router-a`, utiliser `nping` pour forger un paquet TCP/IP à transmettre au `router-b`, dont les paramètres sont les suivants :

- *identification* IP : `0x1000`
- port source : 4444
- port destination : 80
- numéro de séquence : `0x100`
- *flags* : SYN

```
root@router-a:/$ nping -c 1 --id 0x1000 --tcp --source-port 4444 --dest-port 80 \
  --seq 0x100 --flags SYN 192.168.0.6
```

Transmettre deux fois ce paquet.

Question 49 À l'aide de *wireshark*, comparer les contenus des deux paquets.

- si on ignore l'entête AH, le reste du paquet est il identique ?
- le résultat du calcul placé dans le champs `AH ICV` est il identique ?

Selon vous, qu'est-ce qui peut justifier un tel comportement ?

Si on ignore l'entête AH, alors à chaque retransmission, le contenu du paquet TCP/IP est identique. Cependant, à chaque paquet transmis, le résultat du calcul placé dans le champs `AH ICV` est différent.

Cela s'explique par le fait que le calcul du test d'intégrité est réalisé sur sur toute la charge utile (voir la figure 6), y compris l'entête AH (à l'exception du résultat lui-même). Or, dans l'entête AH est présent un index du paquet dans la séquence d'authentification (`AH Sequence`). IPsec incrémente sa valeur à chaque émission de paquet, même si c'est une retransmission en provenance de la même application.

De ce fait, le résultat du calcul du test d'intégrité est différent à chaque émission, cette fonctionnalité est une protection contre le re-jeu. Si l'index `AH Sequence` est différent de celui attendu par IPsec, alors le paquet est abandonné. De même si le résultat du calcul du test d'intégrité n'est pas correct vis-à-vis de cet index.

6.3 Phase 2 : authentification et confidentialité avec ESP

Le vol de données sensibles est l'une des plus grandes inquiétudes des entreprises. L'isolement total des systèmes informatique est aujourd'hui presque impossible et l'écoute passive de canaux de communication plutôt facile à mettre en œuvre. Le protocole **Encapsulating Security Payload** (**ESP**) d'IPSec permet de chiffrer une communication entre deux hôtes et ainsi garantir la confidentialité des données échangées.

L'objectif de cette section est de modifier la configuration d'IPsec afin d'observer l'entête ESP utilisé pour vérifier l'authenticité des paquets IP et transmettre les données de chiffrement. Le paquet ESP est créé en insérant l'entête ESP dans le paquet IP originel et en plaçant la terminaison ESP et les données d'authentification à la fin du paquet, tel que présenté dans la figure 7.

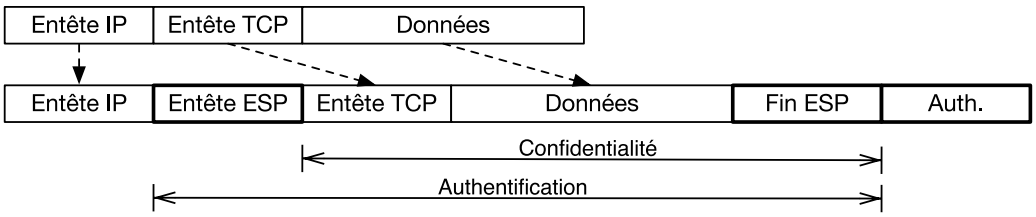

FIGURE 7 – Paquet IP avec ESP en mode transport

Notons que, contrairement à AH, l'authentification des donnée contenues dans les paquets n'inclut pas l'entête IP dans le calcul. ESP s'appuie sur la clé de session échangée à l'aide de IKE lors de la phase 1. Si la vérification des données peut être réalisée conformément à cette clé de session, alors ESP s'appuie sur IKE pour vérifier la correction des adresses IP source et destination.

Question 50 **Sur les deux routeurs**, modifier le fichier de configuration `/etc/ipsec.d/tp.conf` (de manière identique) pour stipuler que nous souhaitons utiliser ESP en mode transport : modifier la valeur de l'option `phase2=ah` par `phase2=esp`.

```
root@router-a:/$ nano /etc/ipsec.d/tp.conf
root@router-a:/$ cat  /etc/ipsec.d/tp.conf
conn myvpn
  left=192.168.0.5
  right=192.168.0.6
  auto=start
  authby=secret
  phase2=esp          # <- modifier ici
  type=transport
```

À répéter sur le `router-b`.

Question 51 À l'aide de *wireshark*, observer le bridge `br-router` et appliquer le filtre `isakmp||tcp||esp`.

Question 52 **Sur les deux routeurs**, redémarrer le démon IPsec pour appliquer les changements.

```
root@router-a:/$ ipsec setup restart
```

À répéter sur le `router-b`.

Question 53 Simuler un serveur HTTP sur le `router-a` à l'aide d'une écoute de *netcat* en TCP sur le port 80.

```
root@router-a:/$ netcat -v -l -s 192.168.0.5 -p 80
listening on [192.168.0.5] 80 ...
```

Question 54 Depuis le `router-b`, simuler une communication avec le serveur HTTP du `router-a` en TCP sur le port 80. Vérifier que *netcat* se termine sur le `router-a`.

```
root@router-b:/$ echo "quit" | netcat -q 0 192.168.0.5 80
```

Question 55 À l'aide de *wireshark*, observer le bridge `br-router` et vérifier que la communication a bien transité dessus en suivant le protocole ESP.

Question 56 Vérifier que les paquets émis par les deux routeurs sont modifiés pour contenir un entête ESP comme décrit sur la figure 7.
Quels sont les champs relatifs à ESP présents dans cet entête que *wireshark* nous indique ?

Nous pouvons observer la présence de l'entête ESP après l'entête IP, comme décrit par la figure 6. Cependant, nous ne pouvons pas identifier quelles sont les données confidentielles (entête TCP, données et terminaison ESP) et quelles sont les données d'authentification à la fin du paquet. Voici un exemple de trame Ethernet telle qu'observée par *wireshark* (les valeurs sont différentes à chaque émission) :

```
Frame 1: 110 bytes on wire (880 bits), 110 bytes captured (880 bits) on interface br-router, id 0
Ethernet II, Src: ca:fe:be:ef:00:06 (ca:fe:be:ef:00:06), Dst: ca:fe:be:ef:00:05 (ca:fe:be:ef:00:05)
Internet Protocol Version 4, Src: 192.168.0.6, Dst: 192.168.0.5
Encapsulating Security Payload
    ESP SPI: 0x67eeffb7 (1743716279)
    ESP Sequence: 1
```

L'entête ESP contient 2 informations importantes :

- `ESP SPI` : le paramètre de sécurité (*Security Parameter Index*).
- `ESP Sequence` : l'index du paquet dans la séquence d'authentification, cet index est incrémenté à chaque émission de paquet.

Exactement comme pour AH, le paramètre de sécurité (SPI) est choisi par les hôtes une fois l'échange de clés terminé. Il est utilisé lors de tous les échanges jusqu'à la négociation de nouvelles clés de session.

Ici aussi, comme pour AH, IPsec incrémente la valeur de l'index du paquet dans la séquence (`ESP Sequence`). Cela permet d'obtenir un résultat de calcul du test d'intégrité différent à chaque émission.

6.3.1 Authenticité et intégrité

Nous avons vu que nous ne pouvons pas identifier quelles sont les données confidentielles (entête TCP, données et terminaison ESP) et quelles sont les données d'authentification à la fin du paquet. Cela s'explique par le fait que les données confidentielles sont chiffrées. Les données d'authentification, à la fin du paquet, sont le résultat du calcul du test d'intégrité, équivalentes au champs `AH ICV` dans l'entête AH : c'est un résultat en clair.

Le protocole ESP supporte la désactivation de l'authentification, la présence des données d'authentification est facultative ; de plus, la taille du résultat peut varier en fonction de l'algorithme choisi lors de la phase 1 (négociation par IKE). Or, comme le résultat du calcul est un nombre avec une forte entropie, dont la taille est inconnue, alors *wireshark* est dans l'incapacité de distinguer la donnée chiffrée du résultat du calcul. C'est la raison pour laquelle ce champs n'apparait pas explicitement dans notre mesure. Pourtant, il est bien présent.

IPsec autorise le choix de l'algorithme ou sa désactivation pour le chiffrement et l'authentification. Nous allons désormais désactiver le chiffrement et conserver l'authentification afin d'observer la présence de la terminaison ESP et les données d'authentification dans les paquets.

Question 57 Désactiver le chiffrement et conserver le même algorithme d'authentification que précédemment.
Sur les deux routeurs, modifier le fichier de configuration `/etc/ipsec.d/tp.conf` (de manière identique) pour désactiver le chiffrement : ajouter une option `esp=null-sha2`.

```
root@router-a:/$ nano /etc/ipsec.d/tp.conf
root@router-a:/$ cat  /etc/ipsec.d/tp.conf
conn myvpn
  left=192.168.0.5
  right=192.168.0.6
  auto=start
  authby=secret
  phase2=esp
  type=transport
  esp=null-sha2       # <- ajouter ici
```

À répéter sur le `router-b`.

Question 58 **Sur les deux routeurs**, redémarrer le démon IPsec pour appliquer les changements.

```
root@router-a:/$ ipsec setup restart
```

À répéter sur le `router-b`.

Question 59 À l'aide de *wireshark*, observer le bridge `br-router` et appliquer le filtre `isakmp||tcp||esp`.

L'outil `nping` (distribué avec `nmap`) permet de forger facilement des paquets IP et de choisir la *payload*. Nous pouvons donc utiliser `nping` pour choisir de la donnée lisible que nous pouvons identifier sur le réseau une fois le chiffrement désactivé. En conséquence, nous pouvons considérer que la donnée qui suit ce que nous avons choisi est alors la terminaison ESP et les données d'authentification.

Question 60 Depuis le `router-a`, utiliser `nping` pour forger un paquet TCP/IP à transmettre au `router-b`, dont les paramètres sont les suivants :
- *identification* IP : `0x1000`
- port source : 8738 (valeur décimale de `0x2222`)
- port destination : 17476 (valeur décimale de `0x4444`)
- numéro de séquence : `0x66668888`
- *flags* : SYN
- *payload* au format texte : "`hello world`"

```
root@router-a:/$ nping -c 1 --id 0x1000 --tcp --flags SYN --seq 0x66668888 \
  --source-port 8738 --dest-port 17476 --data-string "hello world" 192.168.0.6
```

Question 61 À l'aide de *wireshark*, observer le contenu du paquet émis par le `router-a`. Notons que *wireshark* n'est toujours pas capable de déterminer si la donnée est chiffrée ou en clair : il faut donc observer le contenu au format hexadécimal et son *dump* en ASCII.
Combien d'octets suivent la *payload* que nous avons choisie ?

La norme RFC-4303 décrit l'architecture des paquets ESP. Chaque paquet est construit comme suit :
- L'entête ESP est constituée de 8 octets : les 4 premiers octets contiennent le paramètre de sécurité (`SPI`); les 4 octets suivants contiennent l'index du paquet dans la séquence d'authentification (`ESP Sequence`).
- Les données et la terminaison ESP sont alignées sur 4 octets. Des octets de remplissage (*padding*) sont ajoutés en début de terminaison pour assurer cet alignement.
  La terminaison ESP est ensuite composée de 2 octets :
  - le premier octet indique le nombre d'octets de remplissage ajoutés.
  - le second octet est un identifiant unique qui stipule le type de donnée contenu dans la *payload*. Par exemple, l'identifiant `0x06` stipule que la *payload* est de type TCP.
- La résultat du calcul du test d'intégrité (`ICV` : *Integrity Check Value*) est placé à la fin du paquet, en clair.

Question 62 Combien de caractères composent notre *payload*, la chaîne de caractères "`hello world`" ?
Ceci est donc le nombre d'octets présents dans notre *payload*.

Question 63 Si l'on ajoute les 2 octets de la terminaison ESP, combien d'octets d'alignement sont nécessaires pour que les données et la terminaison ESP soient alignées sur 4 octets ?

Question 64 Confirmer votre intuition en transmettant de nouveau un paquet forgé à l'aide de `nping`. Choisir la *payload* de sorte que l'octet qui la suit indique zéro : soit qu'aucun octet de remplissage ne soit ajouté.

```
root@router-a:/$ nping -c 1 --id 0x1000 --tcp --flags SYN --seq 0x66668888 \
  --source-port 8738 --dest-port 17476 --data-string "hello worldddd" 192.168.0.6
```

Vérifier la valeur de l'octet suivant à l'aide de *wireshark*.

Question 65 Connaissant la taille de notre *payload* et de notre terminaison ESP, observer dans *wireshark* à quel octet se terminent nos données et déduire le nombre d'octets dans l'`ICV` : le résultat du calcul du test d'intégrité placé à la fin du paquet ESP.
Aurions nous pu connaitre cette valeur sans avoir à faire tous ces calculs ?

La chaîne de caractères "`hello world`" est composée de 11 caractères, soit 11 octets. Si nous ajoutons les 2 octets obligatoires de la terminaison ESP, nous obtenons 13 octets, qui n'est pas multiple de 4. 3 octets d'alignement sont donc nécessaires.

Nous pouvons reconnaitre notre *payload* dans les données en regardant le *dump* en ASCII fourni par *wireshark* ou directement le contenu au format hexadécimal :

```
debian@myhostname:~$ echo -n "hello world" | xxd -groupsize 1
00000000: 68 65 6c 6c 6f 20 77 6f 72 6c 64                hello world
```

Voici un exemple de paquet ESP, ici le SPI est égal à `0xc303a537` et le numéro de séquence à `0x01`.

```
0000   c3 03 a5 37 00 00 00 01                           ...7....           <- SPI + ESP Sequence
                            22 22 44 44 66 66 88 88           ""DDff..    | entete TCP
0010   00 00 00 00 50 02 05 c8 41 90 00 00               ....P...A...       |

                                        68 65 6c 6c             hell       | payload
0020   6f 20 77 6f 72 6c 64                           o world            |

                         01 02 03 03 06                       .....       <- terminaison ESP
                                        32 7f c6 fe             2...       | ICV
0030   54 59 c0 17 d6 64 65 f0 43 02 8f b8               TY...de.C...       |
```

Dans cet exemple, la terminaison ESP est composée de 3 octets d'alignement : `0x01`, `0x02` et `0x03`. Le nombre d'octets de remplissage ajoutés est de 3 : `0x03`. Finalement, l'identifiant unique stipule que la *payload* est de type TCP : `0x06`.

Nous pouvons confirmer cette intuition en ajoutant 3 octets à notre *payload*. Par exemple, si nous transmettons la chaîne de caractères "`hello worldddd`", voici un résultat possible pour le nouveau paquet ESP :

```
0000   c3 03 a5 37 00 00 00 02                           ...7....           <- SPI + ESP Sequence
                            22 22 44 44 66 66 88 88           ""DDff..    | entete TCP
0010   00 00 00 00 50 02 05 c8 dc c4 00 00               ....P.......       |

                                        68 65 6c 6c             hell       | payload
0020   6f 20 77 6f 72 6c 64 64 64 64                     o worlddd          |

                                  00 06                       ..           <- terminaison ESP
                                        1a 3b ed 3d             .;.=       | ICV
0030   f2 61 e7 09 04 ac 83 90 14 9d 3e 6e               .a........>n       |
```

Dans les deux exemples précédents, nous observons un résultat du calcul du test d'intégrité (`ICV`), placé à la fin du paquet ESP, dont la valeur est codée sur 16 octets.

Nous aurions pu connaitre la taille de l'`ICV` sans réaliser ces calculs si nous avions observé le détail de l'échange de clés avec IKE. En effet, les deux hôtes qui s'authentifient (`router-a` et `router-b`) choisissent la taille de l'`ICV` en même temps que les algorithmes.

Question 66 Sur le `router-a`, redémarrer le démon IPsec pour provoquer un nouvel échange de clés.

```
root@router-a:/$ ipsec setup restart
```

Question 67 À l'aide de *wireshark*, observer le détail du dernier paquet de type `IKE_SA_INIT`. C'est-à-dire celui qui contient les choix des algorithmes. En particulier, observer la section *Internet Security Association and Key Management Protocol* (`isakmp`) et le champs `Payload: Security Association`.
Ce champs ne doit contenir qu'un seul `Payload: Proposal` (le choix des algorithmes) avec 3 fois `Payload: Transform`. La première transformation choisie est celle destinée au protocole ESP.
Que nous dit l'`ID` de cette transformation à propos de l'`ICV` ?

```
Internet Security Association and Key Management Protocol
    Initiator SPI: 6965eae568b80e91
    Responder SPI: 049219eb35a1a5bd
[...]
    Payload: Security Association (33)
[...]
        Payload: Proposal (2) # 1
[...]
            Payload: Transform (3)
                Transform ID (ENCR): AES-GCM with a 16 octet ICV (20)
```

6.3.2 Confidentialité

Si *wireshark* ne détecte pas automatiquement l'`ICV` dans les paquets ESP, nous sommes désormais en mesure de l'identifier. Nous pouvons donc demander à IPsec d'activer le chiffrement des données confidentielles et nous pourrons distinguer les données chiffrées du résultat du calcul du test d'intégrité.

Même si le protocole ESP supporte la désactivation de l'authentification, nous conservons celle-ci afin de garantir l'authenticité et intégrité des messages échangés. En effet, la confidentialité seule n'apporte aucune garantie que nous communiquons avec un hôte légitime et/ou que les paquets n'ont pas été modifiés.

Question 68 Réactiver le chiffrement des messages échangés entre les deux routeurs. Ne pas désactiver la fonctionnalité d'authentification.
Sur les deux routeurs, modifier le fichier de configuration `/etc/ipsec.d/tp.conf` (de manière identique) et supprimer l'option qui permet de désactiver le chiffrement : l'option `esp=null-sha2`.

```
root@router-a:/$ nano /etc/ipsec.d/tp.conf
root@router-a:/$ cat  /etc/ipsec.d/tp.conf
conn myvpn
  left=192.168.0.5
  right=192.168.0.6
  auto=start
  authby=secret
  phase2=esp
  type=transport
```

À répéter sur le `router-b`.

Question 69 **Sur les deux routeurs**, redémarrer le démon IPsec pour appliquer les changements.

```
root@router-a:/$ ipsec setup restart
```

À répéter sur le `router-b`.

Question 70 À l'aide de *wireshark*, observer le bridge `br-router` et appliquer le filtre `isakmp||tcp||esp`.

Question 71 Depuis le `router-a`, utiliser `nping` pour forger un paquet TCP/IP à transmettre au `router-b`, dont les paramètres sont les suivants :

- *identification* IP : `0x1000`
- port source : 8738 (valeur décimale de `0x2222`)
- port destination : 17476 (valeur décimale de `0x4444`)
- numéro de séquence : `0x66668888`
- *flags* : SYN
- *payload* au format texte : "`hello world`"

```
root@router-a:/$ nping -c 1 --id 0x1000 --tcp --flags SYN --seq 0x66668888 \
  --source-port 8738 --dest-port 17476 --data-string "hello world" 192.168.0.6
```

Question 72 À l'aide de *wireshark*, observer le contenu du paquet émis par le `router-a`. Notons que *wireshark* n'est toujours pas capable de déterminer si la donnée est chiffrée ou en clair mais nous savons que l'`ICV` contient 16 octets.
Peut on observer la *payload* que nous avons choisie ?
Peut on observer l'entête TCP du paquet que nous avons forgé (ports source et destination, index de séquence) ?

Question 73 Dans le même paquet, nous savons que des champs sont pourtant en clair. Observer tout le contenu de la trame sans l'entête Ethernet (entête IP + données ESP) à l'aide de *wireshark*.
En s'aidant de la figure 7, identifier quelles sont les champs qui sont en clair dans cette trame. Quelles informations intéressantes peut on en extraire ?

Comme indiqué sur la figure 7, il y a seulement 3 champs qui sont chiffrés : l'entête TCP, la *payload* et la terminaison ESP. Cela signifie que 3 autres champs sont en clair : l'entête IP, l'entête ESP et l'`ICV`.

Une information importante que nous pouvons extraire de l'entête IP est que nous connaissons les adresses IP source et destination du paquet. Dans notre cas, respectivement 192.168.0.5 et 192.168.0.6 :

```
Internet Protocol Version 4, Src: 192.168.0.5, Dst: 192.168.0.6
    Source Address: 192.168.0.5
    Destination Address: 192.168.0.6

0000                                            45 00                E.
0010   00 58 10 00 00 00 40 32 e9 18 c0 a8 00 05 c0 a8   .X....@2........
0020   00 06                                            ..
```

Dans l'entête ESP, tout comme *wireshark*, nous pouvons lire les valeurs du SPI et de l'index de la séquence ESP. Nous pouvons également identifier l'`ICV`, qui est constitué des 16 derniers octets du message. Toutes ces informations ne sont utiles que si nous possédons la clé de session (choisie lors de la phase 1 avec IKE), pour vérifier l'authenticité et l'intégrité du message.

Voici un exemple de trame ESP, ici le SPI est égal à `0xc97802b9` et l'index de séquence est égal à `0x00000001`. L'`ICV` est égal à `0x7fb06f96cf850751240b6162afc2966f`.

```
Encapsulating Security Payload
    ESP SPI: 0xc97802b9 (3380085433)
    ESP Sequence: 1

0020         c9 78 02 b9 00 00 00 01 f9 78 0c 1e af d9     .x.......x....
0030   f9 39 b9 1a f2 8b 5d 1f b5 5e c5 c9 7f d8 3e 29   .9....]..^....>)
0040   ab 70 2c bd 81 ef 55 33 b4 a7 6c 52 84 d8 49 34   .p,...U3..lR..I4
0050   68 61 56 89 a8 c9 7f b0 6f 96 cf 85 07 51 24 0b   haV.....o....Q$.
0060   61 62 af c2 96 6f                                 ab...o
```

En admettant qu'un adversaire intercepte ce message sur le réseau public et qu'il ne connaisse pas la clé de session (car, pour cela, il faut connaitre le secret partagé PSK et intercepter l'échange de clés IKE), alors nous avons obtenu pour nos communications : l'authenticité, l'intégrité et la confidentialité.

Chiffrer ses données est inutile si le correspondant ne parvient pas à déchiffrer le message qui lui est destiné. La dernière étape consiste donc à vérifier que les deux hôtes réussissent à communiquer correctement. En effet, nous avons forgé un paquet qui n'attendait pas de réponse depuis le `router-a` ; si le `router-b` répond, c'est probablement pour demander une `RST` de la communication.

Question 74 Simuler un serveur HTTP sur le `router-a` à l'aide d'une écoute de *netcat* en TCP sur le port 80.

```
root@router-a:/$ netcat -v -l -s 192.168.0.5 -p 80
listening on [192.168.0.5] 80 ...
```

Question 75 Depuis le `router-b`, simuler une communication avec le serveur HTTP du `router-a` en TCP sur le port 80. Vérifier que *netcat* se termine sur le `router-a`.

```
root@router-b:/$ echo "quit" | netcat -q 0 192.168.0.5 80
```

7 Sécurité des paquets en mode tunnel

Dans la section précédente, nous avons utilisé IPsec en *mode transport*. Dans le mode transport, ce sont uniquement les données transférées (la partie *payload* du paquet IP) qui sont chiffrées et/ou authentifiées. L'entête du paquet IP est inchangé et, de ce fait, le routage des paquets n'est pas modifié.

Les adresses IP ne pouvant pas être modifiées par le NAT sans corrompre le *hash* de l'en-tête AH généré par IPsec, AH ne peut pas être utilisé dans un environnement nécessitant ces modifications d'en-tête (en revanche, il est possible d'avoir recours à l'encapsulation NAT-T pour utiliser ESP : ceci n'est pas détaillé dans ce TP). Le mode transport est donc utilisé pour les communications dites point à point (*host-to-host*). La figure 8 résume l'utilisation d'IPsec en mode transport.

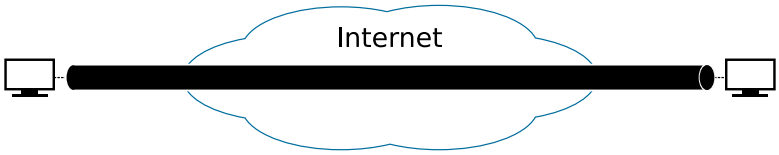

FIGURE 8 – IPsec en mode transport

En mode tunnel, c'est la totalité du paquet IP qui est chiffré et/ou authentifié. Le paquet est ensuite encapsulé dans un nouveau paquet IP avec un nouvel en-tête IP. Au contraire du mode transport, ce mode supporte donc la traversée de NAT quand le protocole AH ou ESP est utilisé.

Le mode tunnel est utilisé pour créer des réseaux privés virtuels (VPN) permettant la communication de réseau à réseau (c'est-à-dire entre deux sites distants), d'hôte à réseau (accès à distance d'un seul utilisateur) ou bien d'hôte à hôte (messagerie privée, utilisation identique à celle de la figure 8). La figure 9 résume l'utilisation d'IPsec en mode tunnel pour une communication de réseau à réseau.

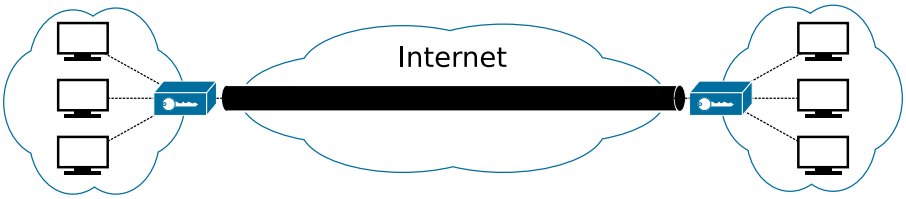

FIGURE 9 – IPsec en mode tunnel

Passons à un cas d'utilisation plus réel. Le schéma de la figure 10 représente le cas d'une entreprise dont les bureaux sont situés sur deux sites distants. Afin de réduire le coût des communications entre les sites, l'entreprise a opté pour la mise en place d'un tunnel sécurisé sur le réseau Internet publique. Elle vous demande de vous assurer que le trafic entre les deux sous-réseaux privés soit sécurisé par un tunnel.

Avant de commencer, nous allons observer la sécurité que nous obtenons avec IPsec configuré en mode transport.

Question 76 À l'aide de *wireshark*, observer le bridge `br-router` et appliquer le filtre `isakmp||esp||tcp`.

Question 77 Simuler un serveur HTTP sur le `router-a` à l'aide d'une écoute de *netcat* en TCP sur le port 80.

```
root@router-a:/$ netcat -v -l -s 192.168.0.5 -p 80
listening on [192.168.0.5] 80 ...
```

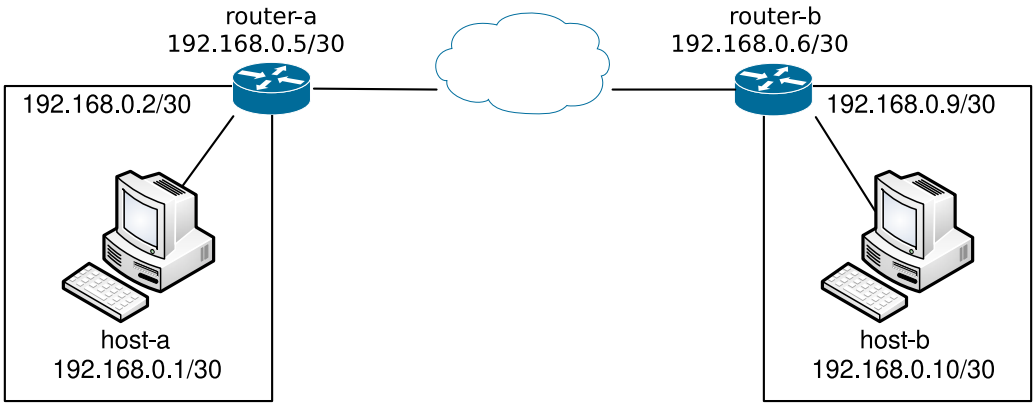

FIGURE 10 – Architecture du réseau

Question 78 Depuis le `router-b`, simuler une communication avec le serveur HTTP du `router-a` en TCP sur le port 80. Vérifier que *netcat* se termine sur le `router-a`.

```
root@router-b:/$ echo "quit" | netcat -q 0 192.168.0.5 80
```

Question 79 À l'aide de *wireshark*, observer les paquets émis par les deux routeurs. Ces paquets sont-ils confidentiels/authentifiés ?

Question 80 Simuler un serveur HTTP sur `host-a` à l'aide d'une écoute de *netcat* en TCP sur le port 80.

```
root@host-a:/$ netcat -v -l -s 192.168.0.1 -p 80
listening on [192.168.0.1] 80 ...
```

Question 81 Depuis `host-b`, simuler une communication avec le serveur HTTP de `host-a` en TCP sur le port 80. Vérifier que *netcat* se termine sur `host-a`.

```
root@router-b:/$ echo "quit" | netcat -q 0 192.168.0.1 80
```

Question 82 À l'aide de *wireshark*, observer les paquets émis par les deux hôtes. Ces paquets sont-ils confidentiels/authentifiés ?

En mode transport, seules les communications point à point sont confidentielles/authentifiées. C'est à dire que la sécurité ne s'applique que pour les communications entre deux hôtes, comme montré sur la figure 8. En effet, les `router-a` et `router-b` communiquent de manière sécurisé, tandis que toute autre machine (`host-a` ou `host-b`) communique sans utiliser IPsec.

L'avantage d'une configuration en mode transport est qu'un hôte peut partager un secret différent avec chacun de ses correspondants. Il peut également choisir des algorithmes de chiffrement et d'authentification différents.

L'inconvénient est qu'il est nécessaire de configurer IPsec sur toutes les machines qui doivent communiquer, et de choisir un mode d'authentification et un secret pour chaque paire de machines qui sera amenée à communiquer. L'alternative est donc de ne configurer IPsec que sur les deux routeurs et de s'assurer qu'il soit utilisé pour tous les paquets transmis (`FORWARD`) entre les sous-réseaux qu'ils relient. C'est-à-dire configurer IPsec en mode tunnel, comme représenté sur la figure 9.

Question 83 **Sur les deux routeurs**, modifier le fichier de configuration `/etc/ipsec.d/tp.conf` (de manière identique) et activer le mode tunnel. Remplacer l'option `type=transport` par `type=tunnel`

```
root@router-a:/$ nano /etc/ipsec.d/tp.conf
root@router-a:/$ cat  /etc/ipsec.d/tp.conf
conn myvpn
  left=192.168.0.5
```

```
  right=192.168.0.6
  auto=start
  authby=secret
  phase2=esp
  type=tunnel          # <- modifier ici
```

À répéter sur le `router-b`.

Question 84 **Sur les deux routeurs**, redémarrer le démon IPsec pour appliquer les changements.

```
root@router-a:/$ ipsec setup restart
```

À répéter sur le `router-b`.

Question 85 Simuler un serveur HTTP sur `host-a` à l'aide d'une écoute de *netcat* en TCP sur le port 80.

```
root@host-a:/$ netcat -v -l -s 192.168.0.1 -p 80
listening on [192.168.0.1] 80 ...
```

Question 86 Depuis `host-b`, simuler une communication avec le serveur HTTP de `host-a` en TCP sur le port 80. Vérifier que *netcat* se termine sur `host-a`.

```
root@host-b:/$ echo "quit" | netcat -q 0 192.168.0.1 80
```

Question 87 À l'aide de *wireshark*, observer les paquets émis par les deux hôtes. Ces paquets sont-ils confidentiels/authentifiés ?

Les paquets échangés entre `host-a` et `host-b` ne sont toujours pas confidentiels. La raison à cela est que le comportement par défaut d'IPsec est de ne pas sécuriser les connexions pour les sous-réseaux qui n'ont pas été identifiés comme étant les extrémités du tunnel. En effet, un routeur peut faire communiquer ses sous réseaux avec l'extérieur et le chiffrement des communications peut poser des problèmes de disponibilité.

Afin de rendre les communications confidentielles, il est nécessaire de spécifier à IPsec quels sont les sous-réseaux pour lesquels la sécurité doit être appliquée.

Question 88 **Sur les deux routeurs**, modifier le fichier de configuration `/etc/ipsec.d/tp.conf` (de manière identique) et ajouter une nouvelle connexion pour spécifier les deux sorties du tunnel.
Utiliser l'option `also=<nom_du_vpn>` pour réutiliser tous les paramètres de la connexion que nous avons précédemment définie. Ajouter les options `leftsubnet=<mask>` et `rightsubnet=<mask>` pour définir les deux masques de sous-réseau.

```
root@router-a:/$ nano /etc/ipsec.d/tp.conf
root@router-a:/$ cat  /etc/ipsec.d/tp.conf
conn myvpn
  left=192.168.0.5
  right=192.168.0.6
  auto=start
  authby=secret
  phase2=esp
  type=tunnel

# ajouter ici:
conn mysubnet
  also=myvpn
  leftsubnet=192.168.0.1/30
  rightsubnet=192.168.0.10/30
```

À répéter sur le `router-b`.

Question 89 **Sur les deux routeurs**, redémarrer le démon IPsec pour appliquer les changements.

```
root@router-a:/$ ipsec setup restart
```

À répéter sur le `router-b`.

Question 90 Dans 3 terminaux, lancer 3 fois *wireshark* (2 fois de plus si le premier n'a pas été fermé).
À l'aide de ces 3 *wireshark*, observer respectivement les bridges `br-router`, `br-a` et `br-b`.

Question 91 Simuler un serveur HTTP sur `host-a` à l'aide d'une écoute de *netcat* en TCP sur le port 80.

```
root@host-a:/$ netcat -v -l -s 192.168.0.1 -p 80
listening on [192.168.0.1] 80 ...
```

Question 92 Depuis `host-b`, simuler une communication avec le serveur HTTP de `host-a` en TCP sur le port 80. Vérifier que *netcat* se termine sur `host-a`.

```
root@host-b:/$ echo "quit" | netcat -q 0 192.168.0.1 80
```

Question 93 À l'aide de *wireshark*, observer sur le bridge `br-router` les paquets émis par les deux hôtes. Vérifier que les paquets sont confidentiels/authentifiés.
Quelles sont les adresses IP source et destination présentes dans l'entête IP de ces paquets ? Sont elles les adresses IP respectives des `host-a` et `host-b` ?

Question 94 À l'aide de *wireshark*, observer sur les bridges `br-a` et `br-b` les mêmes paquets émis par les deux hôtes. Ces paquets sont-ils confidentiels/authentifiés ?
Quelles sont les adresses IP source et destination présentes dans l'entête IP de ces paquets ? Sont elles identiques à celles observées sur le bridge `br-router` ?

L'architecture que nous avons construite est celle représentée sur la figure 9. Les paquets sont confidentiels et authentifiés seulement dans le tunnel : c'est-à-dire entre les `router-a` et `router-b`. C'est ce que nous observons sur le bridge `br-router`. Une fois sorti du tunnel, sur les sous-réseaux, les paquets sont en clair. C'est ce que nous observons sur les bridges `host-a` et `host-b`.

Les adresses IP sont différentes selon si le paquet se trouve dans le tunnel ou hors de celui-ci. Afin que les paquets soient routés correctement, ceux-ci contiennent donc deux entêtes IP : une pour aller d'une extrémité du tunnel à l'autre ; une pour être routés jusqu'à leur destination une fois qu'ils sont hors du tunnel. Nous verrons comment un nouvel entête IP est ajouté par IPsec dans les sections 7.2 et 7.3.

7.1 Phase 1 : échange des clés

Exactement comme en mode transport, la sécurité de la communication en mode tunnel repose sur la négociation de clés de session lors de la phase 1 : l'échange des clés avec IKE. Egalement, les connexions doivent être activées pour que l'échange des clés soit réalisé et que les machines puissent les utiliser.

Exactement comme en mode transport, nous allons donc modifier la configuration d'IPsec pour ne pas automatiser les échanges de clés et activation des connexions. Ensuite, nous allons observer le comportement d'IPsec.

Question 95 **Sur les deux routeurs**, modifier le fichier de configuration `/etc/ipsec.d/tp.conf` (de manière identique) pour demander à IPsec de sécuriser les connexions dans le tunnel. Demander à IPsec d'automatiser l'échange de clés à l'ajout d'une nouvelle connexion (étape manuelle) : modifier l'option `auto=start` par `auto=add`.

```
conn myvpn
  left=192.168.0.5
  right=192.168.0.6
  auto=add            # <- modifier ici
  authby=secret
  phase2=esp
  type=tunnel
```

```
conn mysubnet
  also=myvpn
  leftsubnet=192.168.0.1/30
  rightsubnet=192.168.0.10/30
```

À répéter sur le `router-b`.

Question 96 **Sur les deux routeurs**, redémarrer le démon IPsec pour appliquer les changements.

```
root@router-a:/$ ipsec setup restart
```

À répéter sur le `router-b`.

Question 97 À l'aide de *wireshark*, observer le bridge `br-router` et appliquer le filtre `isakmp||esp||tcp`.

Question 98 Tenter d'établir une communication entre `host-a` et `host-b`. Que peut on observer à l'aide de *wireshark* sur le bridge `br-router` ? La communication est-elle sécurisée ? Pourquoi ?

```
root@host-a:/$ netcat -v -l -s 192.168.0.1 -p 80
```

```
root@host-b:/$ echo "quit" | netcat -q 0 192.168.0.1 80
```

La communication entre les deux sous-réseaux n'est pas sécurisée car nous n'avons pas encore procédé à l'échange de clés avec IKE. En effet, nous n'avons pas vu passer de paquets *Internet Security Association and Key Management Protocol* (`isakmp`).

Question 99 **Sur les deux routeurs**, démarrer la connexion que nous avons définie pour IPsec et ainsi procéder à un échange de clés avec IKE.

```
root@router-a:/$ ipsec auto --up myvpn
```

À répéter sur le `router-b`.

Question 100 À l'aide de *wireshark*, observer le bridge `br-router` et vérifier que l'échange de clés a bien été réalisé.

Si nous tentons d'établir une communication entre `host-a` et `host-b` maintenant, celle-ci ne sera toujours pas sécurisée. Cela s'explique par le fait que la connexion que nous avons définie pour les sous-réseaux n'est pas activée. En revanche, nous venons d'activer la connexion configurée entre les deux routeurs ; ils peuvent donc communiquer de manière sécurisée comme en mode transport.

Pour la connexion configurée entre les sous-réseaux, l'option `also=myvpn` permet d'hériter les paramètres de la connexion configurée entre les deux routeurs. Or le paramètre `auto` de celle-ci est réglé sur `add`. Il faut donc activer manuellement la seconde connexion.

Question 101 **Sur les deux routeurs**, démarrer la connexion que nous avons définie pour IPsec pour les deux sous-réseaux.

```
root@router-a:/$ ipsec auto --up mysubnet
```

À répéter sur le `router-b`.

Question 102 À l'aide de *wireshark*, observer le bridge `br-router`. Quels nouveaux paquets ont été échangés ?

À chaque nouvelle association (SA), le protocole IKE en mode *main* procède à une vérification des identités : les messages 5 et 6 de la figure 5. Lorsque nous avons activé IPsec, 6 messages ont été échangés, où les messages 5 et 6 ont permis aux `router-a` et `router-b` de vérifier leur SA.

En activant la connexion des sous-réseaux, deux nouvelles SA sont nécessaires :

— une entre le `router-a` et le sous-réseaux de `host-b`

— une entre le `router-b` et le sous-réseaux de `host-a`

IPsec procède alors à la spécification de ces SA et IKE procède à une vérification en émettant à chaque fois les messages 5 et 6. Notons que si la donné est trop lourde, les messages peuvent être fragmentés en plusieurs paquets. L'échange doit donc ressembler à ceci :

```
No. Time   Source      Destination  Protocol Length Info
1   7.777  192.168.0.5 192.168.0.6  ISAKMP   581  CREATE_CHILD_SA MID=02 Initiator Request (fragment 1/2)
2   7.778  192.168.0.5 192.168.0.6  ISAKMP   201  CREATE_CHILD_SA MID=02 Initiator Request (fragment 2/2)
3   7.847  192.168.0.6 192.168.0.5  ISAKMP   491  CREATE_CHILD_SA MID=02 Responder Response
4   9.041  192.168.0.6 192.168.0.5  ISAKMP   581  CREATE_CHILD_SA MID=01 Responder Request (fragment 1/2)
5   9.041  192.168.0.6 192.168.0.5  ISAKMP   201  CREATE_CHILD_SA MID=01 Responder Request (fragment 2/2)
6   9.058  192.168.0.5 192.168.0.6  ISAKMP   491  CREATE_CHILD_SA MID=01 Initiator Response
```

Question 103 Tenter de nouveau d'établir une communication entre `host-a` et `host-b`. Observer à l'aide de *wireshark* le bridge `br-router` et vérifier que la communication est sécurisée.

```
root@host-a:/$ netcat -v -l -s 192.168.0.1 -p 80
```

```
root@host-b:/$ echo "quit" | netcat -q 0 192.168.0.1 80
```

Exactement comme en mode transport, nous allons désormais utiliser la fonctionnalité de *Libreswan* pour automatiser l'échange de clés et de négocier toutes les SA au démarrage. Ainsi, IPsec sera toujours utilisé lorsque la configuration le stipule.

Question 104 **Sur les deux routeurs**, modifier le fichier de configuration `/etc/ipsec.d/tp.conf` (de manière identique) et remplacer l'option `auto=add` par `auto=start`.

```
conn myvpn
  left=192.168.0.5
  right=192.168.0.6
  auto=start           # <- modifier ici
  authby=secret
  phase2=esp
  type=tunnel

conn mysubnet
  also=myvpn
  leftsubnet=192.168.0.1/30
  rightsubnet=192.168.0.10/30
```

À répéter sur le `router-b`.

Question 105 **Sur les deux routeurs**, redémarrer le démon IPsec pour appliquer les changements.

```
root@router-a:/$ ipsec setup restart
```

À répéter sur le `router-b`.

7.2 Phase 2 : authentification avec AH

Comme nous l'avons vu précédemment, les adresses IP des mêmes paquets sont différentes selon si le paquet se trouve dans le tunnel ou hors de celui-ci. Afin que les paquets soient routés correctement, ceux-ci contiennent donc deux entêtes IP. L'objectif de cette section est de modifier la configuration d'IPsec afin d'observer l'ajout d'une nouvelle entête IP et de l'entête **Authentication Header** (**AH**), utilisé pour vérifier l'authenticité des paquets. Le contenu de l'entête AH est identique à celui décrit dans la section 6.2, nous n'étudions pas son contenu dans cette section.

Le mode tunnel nécessite la création d'un nouvel entête IP, après lequel sont placés l'entête AH puis le paquet IP originel, tel que présenté sur la figure 11.

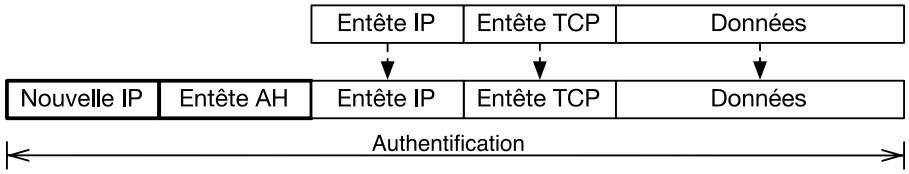

FIGURE 11 – Paquet IP avec AH en mode tunnel

Notons ici aussi que si IPsec permet d'utiliser l'authentification (AH) sans chiffrement (ESP), il est préférable ne pas utiliser cette fonctionnalité. La configuration décrite dans cette section est donc à but éducatif. En effet, aujourd'hui, la puissance de calcul des machines et les débits obtenus sur les réseaux sont tels que la réduction des performances engendrée par des tâches de chiffrement/déchiffrement reste acceptable.

Question 106 **Sur les deux routeurs**, modifier le fichier de configuration `/etc/ipsec.d/tp.conf` (de manière identique) pour stipuler que nous souhaitons utiliser AH en mode tunnel : modifier la valeur de l'option `phase2=esp` par `phase2=ah`.

```
root@router-a:/$ nano /etc/ipsec.d/tp.conf
root@router-a:/$ cat  /etc/ipsec.d/tp.conf
conn myvpn
  left=192.168.0.5
  right=192.168.0.6
  auto=start
  authby=secret
  phase2=ah           # <- modifier ici
  type=tunnel

conn mysubnet
  also=myvpn
  leftsubnet=192.168.0.1/30
  rightsubnet=192.168.0.10/30
```

À répéter sur le `router-b`.

Question 107 **Sur les deux routeurs**, redémarrer le démon IPsec pour appliquer les changements.

```
root@router-a:/$ ipsec setup restart
```

À répéter sur le `router-b`.

Question 108 À l'aide de 3 *wireshark*, observer respectivement les bridges `br-router`, `br-a` et `br-b`. Appliquer le filtre `isakmp||tcp`.

Question 109 Depuis le `host-a`, forger un paquet TCP/IP à l'aide de `nping` et vérifier que la communication passe bien par le tunnel. Observer à l'aide de *wireshark* le bridge `br-router`.
Combien d'entêtes IP sont présents dans le paquet ? À quelles positions sont ils présents par rapport à l'entête AH ? À quoi correspondent les adresses IP source et destination dans les différents entêtes IP.

```
root@host-a:/$ nping -c 1 --id 0x1000 --tcp --flags SYN --seq 0x6666 \
  --source-port 8738 --dest-port 17476 --data-string "hello world" 192.168.0.10
```

Deux entêtes IP sont présents dans le paquet :

- le premier entête, placé avant l'entête AH, est ajouté par IPsec et contient des adresses IP source et destination qui correspondent aux entrée et sortie du tunnel.
  Tant que le paquet se trouve dans le tunnel, c'est cet entête qui est utilisé par les routeurs sur le réseau publique.
  Une fois que le paquet atteint la sortie du tunnel, IPsec retire cet entête ainsi que l'entête AH. C'est le second entête qui est utilisé pour terminer le routage.
- le second entête, placé après l'entête AH, est l'entête créé par la machine qui a initié la communication. Il est laissé intact par IPsec et placé après l'entête AH : il est considéré comme de la donnée.

La figure 11 représente un paquet IP avec AH en mode tunnel. Le premier entête contient les nouvelles adresses IP, ajoutées par IPsec ; le second entête contient adresses IP telles que l'initiateur du paquet l'a créé.

Question 110 À l'aide de *wireshark*, observer respectivement les bridges `br-a` et `br-b`. L'entête AH doit être absent du paquet à cet endroit (nous sommes hors du tunnel).
Vérifier que les adresses dans l'entête IP correspondent aux adresses présentes dans le second entête IP lorsque le paquet se trouve dans le tunnel.

7.3 Phase 2 : authentification et confidentialité avec ESP

En mode tunnel, les adresses IP des mêmes paquets sont différentes selon si le paquet se trouve dans le tunnel ou hors de celui-ci. Exactement comme pour AH, afin que les paquets soient routés correctement, ceux-ci contiennent donc deux entêtes IP. L'objectif de cette section est de modifier la configuration d'IPsec afin d'observer l'ajout d'un nouvel entête IP et de l'entête **Encapsulating Security Payload** (**ESP**), utilisé pour vérifier l'authenticité et garantir la confidentialité des paquets. Les contenus de l'entête ESP et de la terminaison ESP sont identiques à ceux décrit dans la section 6.3, nous n'étudions pas ces contenu dans cette section.

En mode tunnel, le paquet IP originel est entièrement encapsulé dans un nouveau paquet. Un nouvel entête IP est créé, suivi de l'entête ESP et du paquet originel. Puis la terminaison ESP et les données d'authentification y sont annexés, tel que représenté sur la figure 12.

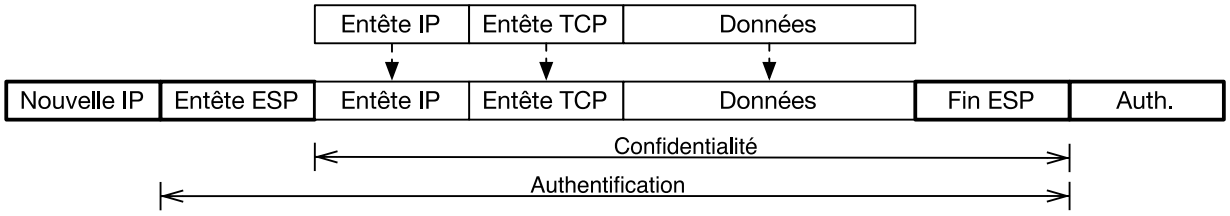

FIGURE 12 – Paquet IP avec ESP en mode tunnel

En mode tunnel, l'entête IP du paquet est originel est considéré comme de la donné, son contenu est donc lui aussi confidentiel. ESP en mode tunnel permet donc de masquer les adresses IP source et destination d'un paquet tant que celui-ci se trouve dans le tunnel.

Exactement comme en mode transport, l'authentification est facultative et *wireshark* ne peut pas déterminer si le résultat du calcul du test d'intégrité fait partie de la donnée chiffrée ou non. Gardons en tête que, dans notre cas, l'`ICV` placé à la fin du paquet ESP a une valeur codée sur 16 octets.

Comme dans la section 6.3, pour observer la donnée présente dans le paquet ESP, nous commençons par désactiver le chiffrement et conserver l'authentification afin d'observer la présence de l'entête IP dans la donnée.

Question 111 **Sur les deux routeurs**, modifier le fichier de configuration `/etc/ipsec.d/tp.conf` (de manière identique) pour stipuler que nous souhaitons utiliser ESP en mode tunnel : modifier la valeur de l'option `phase2=ah` par `phase2=esp`.
Egalement, désactiver le chiffrement : ajouter une option `esp=null-sha2`.

```
root@router-a:/$ nano /etc/ipsec.d/tp.conf
root@router-a:/$ cat  /etc/ipsec.d/tp.conf
conn myvpn
  left=192.168.0.5
  right=192.168.0.6
  auto=start
  authby=secret
  phase2=esp          # <- modifier ici
  type=tunnel
  esp=null-sha2       # <- ajouter ici

conn mysubnet
  also=myvpn
  leftsubnet=192.168.0.1/30
  rightsubnet=192.168.0.10/30
```

À répéter sur le `router-b`.

Question 112 **Sur les deux routeurs**, redémarrer le démon IPsec pour appliquer les changements.

```
root@router-a:/$ ipsec setup restart
```

À répéter sur le `router-b`.

Question 113 À l'aide de 3 *wireshark*, observer respectivement les bridges `br-router`, `br-a` et `br-b`. Appliquer le filtre `isakmp||esp||tcp`.

Question 114 Depuis le `host-a`, forger un paquet TCP/IP contenant la *payload* "`hello world`" à l'aide de `nping` et vérifier que la communication passe bien par le tunnel. Observer à l'aide de *wireshark* le bridge `br-router`.
Vérifier que la donnée est en clair en identifiant la présence de la *payload*.

```
root@host-a:/$ nping -c 1 --id 0x1000 --tcp --flags SYN --seq 0x6666 \
  --source-port 8738 --dest-port 17476 --data-string "hello world" 192.168.0.10
```

Question 115 Nous savons que *wireshark* ne peut pas identifier l'entête IP présente dans la donnée. Mais nous connaissons les adresses IP source et destination du paquet transmis par le `host-a`. Nous pouvons donc identifier l'entête IP si nous reconnaissons ces adresses.
Utiliser Python pour identifier les valeurs hexadécimales des adresses IP de `host-a` et `host-b`.

```
debian@myhostname:~$ python3
>>> [hex(192), hex(168), hex(0), hex(1)]
>>> [hex(192), hex(168), hex(0), hex(10)]
```

Question 116 Dans le contenu du paquet observé à l'aide *wireshark* le bridge `br-router`, vérifier que ces deux adresses sont présentes pour identifier l'entête IP.
Dans l'entête IP, l'adresse source est placée en premier et l'adresse destination en second. Si le paquet a été forgé depuis le `host-a`, alors chercher la donnée :

```
c0 a8 00 01 c0 a8 00 0a
```

Dans le contenu du paquet, nous pouvons également observé la terminaison ESP à la fin du paquet, en ignorant les 16 derniers octets (`ICV`). Avec la *payload* "`hello world`", nous avons 3 octets de remplissage et deux octets : respectivement le nombre d'octets de remplissage ajoutés et l'identifiant unique qui stipule le type de donnée.

```
01 02 03 03 04
```

Ici, le dernier octet indique 4, c'est l'identifiant unique pour stipuler du contenu IPv4. L'identifiant 41 stipule du contenu IPv6 et l'identifiant 6 stipule du contenu TCP (voir la norme RFC-4303).

Question 117 **Sur les deux routeurs**, modifier le fichier de configuration `/etc/ipsec.d/tp.conf` (de manière identique) pour réactiver le chiffrement : supprimer l'option `esp=null-sha2`.

```
root@router-a:/$ nano /etc/ipsec.d/tp.conf
root@router-a:/$ cat  /etc/ipsec.d/tp.conf
conn myvpn
  left=192.168.0.5
  right=192.168.0.6
  auto=start
  authby=secret
  phase2=esp
  type=tunnel

conn mysubnet
  also=myvpn
  leftsubnet=192.168.0.1/30
  rightsubnet=192.168.0.10/30
```

À répéter sur le `router-b`.

Question 118 **Sur les deux routeurs**, redémarrer le démon IPsec pour appliquer les changements.

```
root@router-a:/$ ipsec setup restart
```

À répéter sur le `router-b`.

Question 119 À l'aide de 3 *wireshark*, observer respectivement les bridges `br-router`, `br-a` et `br-b`. Appliquer le filtre `isakmp||esp||tcp`.

Question 120 Depuis le `host-a`, forger un paquet TCP/IP contenant la *payload* "`hello world`" à l'aide de `nping` et vérifier que la communication passe bien par le tunnel. Observer à l'aide de *wireshark* le bridge `br-router`.
Peut on observer l'entête IP et la *payload* TCP dans le contenu ESP ?
Quelles sont les seules adresses IP source et destination que nous pouvons observer ?

```
root@host-a:/$ nping -c 1 --id 0x1000 --tcp --flags SYN --seq 0x6666 \
  --source-port 8738 --dest-port 17476 --data-string "hello world" 192.168.0.10
```

Lorsque le chiffrement est activé, nous ne pouvons plus observer l'entête IP et la *payload* TCP dans le contenu ESP. Exactement comme en mode transport, nous ne pouvons observer que le paramètre de sécurité (`SPI`) et l'index de séquence. Nous savons également que les 16 derniers octets sont en clair mais ils représentent le résultat du test d'intégrité : pas de la donnée.

Les seules adresses IP source et destination que nous pouvons observer sont contenues dans le nouvel entête IP, ajouté par IPsec : ce sont respectivement les adresses de l'entrée et de la sortie du tunnel.

Si nous observons le bridge `br-b`, nous pouvons voir un paquet TCP/IP, dont les adresses IP source et destination correspondent respectivement à celles de `host-a` et `host-b`. Ceci est donc la preuve que l'entête ESP a bien été déchiffrée à la sortie du tunnel `router-b`.

Notons qu'avant d'encapsuler le paquet TCP/IP dans la *payload* ESP ou AH, les entrée et sortie du tunnel peuvent également réaliser un NAT en modifiant les adresses IPs source et destination par leurs propres adresses. De ce fait, le destinataire du paquet (`host-b`) ne connaitra pas non-plus l'adresse de son correspondant (`host-a`) : il ne connaitra que l'adresse de la sortie du tunnel (`router-b`).

Nous ne traitons pas le cas du NAT dans ce TP, cette tâche peut être réalisée à l'aide du logiciel `iptables`, conjointement avec l'utilisation d'IPsec. Des applications dédiées au VPN automatisent ces tâches (par exemple `wireguard`) ; dans ce cas, l'encapsulation est réalisée dans la *payload* TCP (ou UDP) et le protocole est positionné dans la couche 7 du modèle OSI : la couche application (voir la figure 2).

8 Mesures de sécurité supplémentaires

La sécurité des communications que nous avons mise en place pour nos sous-réseaux est basée sur le partage d'un secret (PSK). Jusqu'à maintenant nous avons fait passer le secret partagé d'un ordinateur à l'autre par nos propres moyens : ce qui est ni pratique ni sécurisé. De plus, on pourrait arguer que l'utilisation d'un secret partagé pose, par construction, un problème de sécurité. En effet, il "suffirait" d'avoir un accès privilégié à l'une des *gateway* du VPN (une des extrémités du tunnel) pour connaître le secret de toutes les *gateways*. Nous allons voir comment régler ce problème avec l'utilisation de la cryptographie asymétrique.

8.1 Cryptographie asymétrique

L'objectif de cette section est de s'assurer que nos routeurs ne possèdent pas le même secret, de sorte que la corruption d'un des routeurs n'engendre pas une fuite du secret de l'autre routeur. Également, l'objectif est que les secrets sur chaque routeur n'aient jamais besoin de transiter sur le réseau.

Nous allons donc utiliser la cryptographie asymétrique pour que la seule information à transmettre au second routeur soit une signature (de la donnée publique). La clé privée, générée sur chaque routeur, ne sera jamais transmise. Nous nous basons sur RSA comme système de cryptographie asymétrique.

Question 121 À ce stade, nous pouvons supprimer le secret partagé des deux routeurs car nous ne l'utiliserons plus. Supprimer le fichier `/etc/ipsec.d/tp.secrets` des deux routeurs.

```
root@router-a:/$ rm -f /etc/ipsec.d/tp.secrets
```

À répéter sur le `router-b`.

Question 122 **Sur les deux routeurs**, modifier le fichier de configuration `/etc/ipsec.d/tp.conf` (de manière identique) pour demander à IPsec d'authentifier les extrémités du tunnel par signature RSA et pas par secret partagé.

```
root@router-a:/$ nano /etc/ipsec.d/tp.conf
root@router-a:/$ cat  /etc/ipsec.d/tp.conf
conn myvpn
  left=192.168.0.5
  right=192.168.0.6
  auto=start
  authby=rsasig       # <- modifier ici
  phase2=esp
  type=tunnel

conn mysubnet
  also=myvpn
  leftsubnet=192.168.0.1/30
  rightsubnet=192.168.0.10/30
```

À répéter sur le `router-b`.

Question 123 Sur le `router-a`, générer une paire de clés RSA de 4096 bits.
Noter la valeur de l'identifiant CKAID qui nous permettra de faire référence à cette clé lors d'une utilisation future.

```
root@router-a:/$ ipsec newhostkey --keytype rsa --bits 4096
Generated RSA key pair with CKAID <valeur_du_ckaid> was stored in the NSS database
```

Question 124 Toujours sur le `router-a`, afficher la signature de la clé à partir de son identifiant CKAID.
Le résultat obtenu est du texte qui peut directement être copié dans la configuration de la connexion, définie dans le fichier `/etc/ipsec.d/tp.conf`.
C'est ce résultat que nous pouvons faire transiter sur le réseau sans crainte.

```
root@router-a:/$ ipsec showhostkey --left --ckaid <valeur_du_ckaid>
```

Question 125 Sur le `router-b`, générer une paire de clés RSA de 4096 bits.
Ici aussi, noter la valeur de l'identifiant CKAID qui nous permettra de faire référence à cette clé lors d'une utilisation future.

```
root@router-b:/$ ipsec newhostkey --keytype rsa --bits 4096
Generated RSA key pair with CKAID <valeur_du_ckaid> was stored in the NSS databas
```

Question 126 Toujours sur le `router-b`, afficher la signature de la clé à partir de son identifiant CKAID.
Cette fois-ci, utiliser l'option `--right` à la place de `--left`.

```
root@router-b:/$ ipsec showhostkey --right --ckaid <valeur_du_ckaid>
```

Question 127 **Sur les deux routeurs**, modifier le fichier de configuration `/etc/ipsec.d/tp.conf` (de manière identique) pour renseigner les signatures des deux extrémités du tunnel.

```
root@router-a:/$ nano /etc/ipsec.d/tp.conf
root@router-a:/$ cat  /etc/ipsec.d/tp.conf
conn myvpn
  left=192.168.0.5
  right=192.168.0.6
  auto=start
```

```
  authby=rsasig
  phase2=esp
  type=tunnel
  leftrsasigkey=<signature_du_router-a>   # <- ajouter ici
  rightrsasigkey=<signature_du_router-b>  # <- ajouter ici

conn mysubnet
  also=myvpn
  leftsubnet=192.168.0.1/30
  rightsubnet=192.168.0.10/30
```

À répéter sur le `router-b`.

Question 128 À l'aide de 3 *wireshark*, observer respectivement les bridges `br-router`, `br-a` et `br-b`. Appliquer le filtre `isakmp||esp||tcp`.

Question 129 **Sur les deux routeurs**, redémarrer le démon IPsec pour appliquer les changements.

```
root@router-a:/$ ipsec setup restart
```

À répéter sur le `router-b`.

Question 130 À l'aide de *wireshark*, vérifier qu'un échange de clés avec IKE a lieu sur le bridge `br-router`.

Question 131 Tenter d'établir une communication entre `host-a` et `host-b`. Vérifier que les deux hôtes réussissent à communiquer.
Observer à l'aide de *wireshark*, le bridge `br-router` et vérifier que la communication est sécurisée.
Observer les paquets en clair sur les bridges `br-a` et `br-b`.

```
root@host-a:/$ netcat -v -l -s 192.168.0.1 -p 80
```

```
root@host-b:/$ echo "quit" | netcat -q 0 192.168.0.1 80
```

Avec cette stratégie, la configuration de notre tunnel VPN ne nécessite pas la transmission de secret sur le réseau publique : seulement des signatures de clés RSA.

Néanmoins, le vol de secret (d'une clé RSA privée) sur l'un des deux routeurs permettra toujours l'usurpation d'identité et/ou une attaque de l'homme du milieu. En effet, l'adversaire pourra toujours négocier de nouvelles clés de session à l'aide du protocole IKE.

8.2 Autorité de certification (CA)

Afin de sécuriser davantage la négociation de clés de session, il est possible de construire un tunnel IPSec en se basant sur des certificats. Un certificat électronique est **un ensemble de données** contenant :

- au moins une clé publique (celle du routeur : de l'extrémité du tunnel).
- des informations d'identification, par exemple : nom, localisation, adresse électronique.
- au moins une signature, construite à partir de la clé privée d'un tiers de confiance ; lorsqu'il n'y en a qu'une, l'entité signataire est alors la seule autorité permettant de prêter confiance (ou non) à l'exactitude des informations du certificat.

D'autres données peuvent être contenues dans le certificat : signature du routeur, clé publique de l'autorité, etc.

En termes d'utilisation, la création et l'utilisation de certificat se déroule comme suit :

1. Chaque routeur génère, de la même manière que dans la section précédente, une paire de clés d'un système de chiffrement asymétrique (comme RSA) puis remplit une demande de certificat. Remplir une demande de certificat revient à créer un certificat sans signature.

   Cette demande est accompagnée de la clé publique précédemment produite.

2. Chaque routeur soumet sa demande de certificat à une autorité de certification (*Certificate Authority*, CA). Le choix de l'autorité de certification est tel que le second routeur ait confiance en cette autorité.

3. Si l'autorité de certification considère la demande légitime, elle signe les informations et la clé publique du routeur à partir de sa clé privée. Elle transmet ensuite le certificat signé au routeur qui en a fait la demande.

L'idée est que les deux routeurs établissent une confiance avec l'autorité de certification, pas avec le second routeur. Lors de la création de clés de session avec IKE, c'est à dire de la phase 1 :

5. Les deux routeurs transmettent leur certificat, contenant leur clé publique (signée par l'autorité) et vérifient respectivement que celle du pair est elle aussi signée par l'autorité de certification.
6. Si la clé publique du pair est signée, alors le routeur l'utilise pour une authentification par cryptographie asymétrique (exactement comme dans la section précédente) et des clés de session sont négociées (comme dans la section 6.1).

Le fonctionnement de la phase 2 ne change pas : sont appliqués les mesures de sécurité négociées lors de la phase 1.

Dans cette section, l'objectif est de voir comment nous créons les certificats, comment nous les signons et les transférons d'une machine à une autre, puis comment nous configurons IPsec pour les utiliser lors de la phase 1.

Afin de limiter les échanges, nous allons créer les demandes de certificats directement depuis l'autorité et les signer au même instant. De ce fait, ce sera dans notre cas l'autorité qui choisit les clés privée/publique pour chacun des routeurs et les intègre au certificat. Ceci est une faiblesse car la clé privée du routeur va transiter sur le réseau. Dans un cas d'utilisation réelle, la clé privée ne doit jamais quitter le routeur.

L'autorité de certification doit être une machine accessible par les deux routeurs sans avoir à passer par le second routeur. Dans notre exemple, nous n'avons que 4 machines (voir la figure 1) : nous utiliserons donc la machine `host-a` pour faire figure d'autorité. Ici aussi, ceci est une faiblesse car le `router-a` peut alors procéder à une attaque de l'homme du milieu entre `router-b` et `host-a`.

8.2.1 Création et signature des certificats

Question 132 Sur le `host-a`, qui est notre autorité de certification, les certificats sont stockés dans une base de donnée SQL. Créer une nouvelle base de donnée, à l'aide du programme `certutil`, dans le dossier `/root/tmpdb/`.

```
root@host-a:/$ mkdir -p "/root/tmpdb/"
root@host-a:/$ certutil --empty-password -N -d "/root/tmpdb/"
```

Détails de la commande :
- la base de donnée peut être protégée par un mot de passe, l'option `--empty-password` permet de laisser vide.
- l'option `-N` stipule la création d'une nouvelle base de donnée.
- l'option `-d` permet de choisir le dossier dans lequel la base de donnée est créée.

Question 133 Toujours sur le `host-a`, créer le certificat de l'autorité, qui contient une clé privée et une clé publique. Donner à ce certificat le surnom "`host-a`" et le signer. La clé privé contenue dans ce certificat sera utilisée pour signer les clés publiques des certificats des deux routeurs.
Voici les détails du certificat à créer :
- propriétaire : `host-a`
- nom : `host-a CA`
- attributs de confiance : `Trusted CA` et `Trusted CA for client authentication`
- paire de clés : type RSA de taille 4096 bits

Note : `certutil` demande de taper des caractères aléatoires au clavier pour ajouter de l'entropie.

```
root@host-a:/$ certutil -S -x -n "host-a" -s "O=host-a, CN=host-a CA" -t "CT,," \
  -k rsa -g 4096 -d "/root/tmpdb/"
```

Détails de la commande :
- l'option `-S` stipule la création d'un certificat à ajouter à la base de donnée.
- l'option `-x` stipule que le certificat doit être signé.
- l'option `-n` permet de choisir le surnom (*nickname*) du certificat, nous utiliserons ce surnom pour le manipuler.
- l'option `-s` permet de renseigner les différentes informations d'identification, par exemple : `O` signifie *Owner* ; `CN` signifie *Certificate Name*.
- l'option `-t` permet de renseigner les attributs de confiance (*trust*), par exemple : `C` signifie *Trusted CA* (c'est un certificat d'autorité) ; `T` signifie *Trusted CA for client authentication* (l'autorité peut authentifier des clients : ici les routeurs).

— l'option `-k` permet de choisir le type de clés à générer.
— l'option `-g` permet de choisir la taille des clés à générer.
— l'option `-d` permet de choisir le dossier dans lequel la base de donnée se trouve.

Question 134 Toujours sur le `host-a`, lister les certificats présents dans la base de donnée et vérifier qu'un certificat avec le surnom `host-a` et les attributs `C` et `T` est présent.

```
root@host-a:/$ certutil -L -d "/root/tmpdb/"
```

L'étape suivante consiste à créer les certificats pour les `router-a` et `router-b`. Encore une fois, ceci devrait être réalisé sur les *gateway* respectives et seulement signé sur l'autorité `host-a`, pour éviter de fournir la clé privée à l'autorité et faire transiter celle-ci sur le réseau. Ici nous brûlons les étapes pour gagner du temps et cela introduit une faiblesse.

Question 135 Toujours sur le `host-a`, créer deux certificats qui ne sont pas des certificats d'autorité. Ils contiennent une clé privée et une clé publique pour chaque routeur et sont utilisés pour signer numériquement et chiffrer les données, les routeurs sont utilisés à la fois comme serveur et client. Donner à ces certificats les surnoms "`router-a`" et "`router-b`" puis les signer à l'aide du certificat d'autorité.

```
root@host-a:/$ certutil -S -c "host-a" -n "router-a" \
  -s "O=host-a, CN=router-a-cert" -t ",," -k rsa -g 4096 -d "/root/tmpdb/" \
  --keyUsage    digitalSignature,keyEncipherment \
  --extKeyUsage serverAuth,clientAuth
#
root@host-a:/$ certutil -S -c "host-a" -n "router-b" \
  -s "O=host-a, CN=router-b-cert" -t ",," -k rsa -g 4096 -d "/root/tmpdb/" \
  --keyUsage    digitalSignature,keyEncipherment \
  --extKeyUsage serverAuth,clientAuth
```

Détails des commandes :
— l'option `-S` stipule la création d'un certificat à ajouter à la base de donnée.
— l'option `-c` permet de choisir le certificat qui sera utilisé pour authentifier celui que nous créons.
— l'option `-n` permet de choisir le surnom (*nickname*) du certificat, nous utiliserons ce surnom pour le manipuler.
— l'option `-s` permet de renseigner les différentes informations d'identification, par exemple : `O` signifie *Owner* ; `CN` signifie *Certificate Name*.
— l'option `-t` permet de renseigner les attributs de confiance (*trust*) ; ici, pour l'autorité, aucun.
— l'option `-k` permet de choisir le type de clés à générer.
— l'option `-g` permet de choisir la taille des clés à générer.
— l'option `-d` permet de choisir le dossier dans lequel la base de donnée se trouve.
— l'option `--keyUsage` permet de choisir dans quelles situations le certificat sera utilisé, par exemple : `digitalSignature` pour signer numériquement ou `keyEncipherment` pour chiffrer.
— l'option `--extKeyUsage` permet de choisir de manière étendue dans quelles situations le certificat sera utilisé, par exemple : `serverAuth` pour s'authentifier en tant que serveur ou `clientAuth` pour s'authentifier en tant que client.

Question 136 Toujours sur le `host-a`, lister les certificats présents dans la base de donnée et vérifier que 3 certificats sont présents, où seulement le certificat avec le surnom `host-a` possède les attributs `C` et `T`.

```
root@host-a:/$ certutil -L -d "/root/tmpdb/"
```

Question 137 Toujours sur le `host-a`, exporter dans deux fichiers, nommés respectivement `/root/router-a.p12` et `/root/router-b.p12`, les certificats signés (et contenant une paire de clés RSA) des `router-a` et `router-b`. Nous pouvons choisir un mot de passe pour protéger les certificats ou laisser vide pour ne pas les chiffrer.

```
root@host-a:/$ pk12util -o "/root/router-a.p12" -n "router-a" -d "/root/tmpdb/"
root@host-a:/$ pk12util -o "/root/router-b.p12" -n "router-b" -d "/root/tmpdb/"
```

8.2.2 Configuration d'IPsec

Le rôle de l'autorité de certificat est terminé : les certificats ont été signés à partir de sa clé privée. Désormais, les routeurs peuvent s'appuyer sur la clé publique de l'autorité de certificat pour vérifier la signature lors de l'import de nouveaux certificats. L'idée est la suivante : chaque routeur possède sa clé privée et 3 certificats, contenant seulement des clés publiques.

- le certificat de l'autre routeur, pour pouvoir vérifier sa signature lors de la phase 1 (avec IKE) : il utilisera la clé publique présente à l'intérieur.
- son propre certificat, signé par la CA, pour savoir à quelle paire de clés il fait référence et quelle clé privée il doit utiliser pour s'authentifier lors de la phase 1.
- le certificat de la CA, utilisé pour vérifier la signature de la clé publique contenue dans le certificat de l'autre routeur.

Dans notre exemple, c'est la CA qui a choisi les clés privées des deux routeurs (pour rappel, c'est une faiblesse). Il suffit donc de copier seulement les certificats des deux routeurs sur chacun des deux routeurs.

Question 138 Utiliser `scp` pour copier à travers le réseau les certificats des `router-a` et `router-b` sur les deux machines, dans le dossier `/root/`.
Sur les deux machines `router-a` et `router-b`, le mot de passe root est "`debian`".

```
root@host-a:/$ scp /root/router-a.p12 /root/router-b.p12 root@192.168.0.2:/root/
root@host-a:/$ scp /root/router-a.p12 /root/router-b.p12 root@192.168.0.6:/root/
```

Question 139 Sur chacun des deux routeurs, vérifier que les fichiers ont bien été copiés.

```
root@router-a:/$ ls /root/router-a.p12 /root/router-b.p12
```

À répéter sur le `router-b`.

Question 140 Toujours sur chacun des deux routeurs, importer les certificats signés dans IPsec pour qu'ils puissent être utilisés pour l'authentification.
Si un mot de passe a été choisi pour protéger les certificats, il est nécessaire de le taper pour l'import.

```
root@router-a:/$ ipsec import /root/router-a.p12
root@router-a:/$ ipsec import /root/router-b.p12
```

À répéter sur le `router-b`.

Question 141 **Sur les deux routeurs**, modifier le fichier de configuration `/etc/ipsec.d/tp.conf` (de manière identique) pour renseigner les surnoms des certificats et indiquer que les clés publiques se trouvent dans ceux-ci.
Le mot-clé "`%cert`" comme valeur de `leftrsasigkey` et `rightrsasigkey` permet de stipuler à IPsec d'utiliser ces clés publiques.

```
root@router-a:/$ nano /etc/ipsec.d/tp.conf
root@router-a:/$ cat  /etc/ipsec.d/tp.conf
conn myvpn
  left=192.168.0.5
  right=192.168.0.6
  auto=start
  authby=rsasig
  phase2=esp
  type=tunnel
  leftcert=router-a      # <- ajouter ici
  leftrsasigkey=%cert    # <- modifier ici
  rightcert=router-b     # <- ajouter ici
  rightrsasigkey=%cert   # <- modifier ici

conn mysubnet
  also=myvpn
  leftsubnet=192.168.0.1/30
  rightsubnet=192.168.0.10/30
```

À répéter sur le `router-b`.

Question 142 À l'aide de 3 *wireshark*, observer respectivement les bridges `br-router`, `br-a` et `br-b`. Appliquer le filtre `isakmp||esp||tcp`.

Question 143 **Sur les deux routeurs**, redémarrer le démon IPsec pour appliquer les changements.

```
root@router-a:/$ ipsec setup restart
```

À répéter sur le `router-b`.

Question 144 À l'aide de *wireshark*, vérifier qu'un échange de clés avec IKE a lieu sur le bridge `br-router`.

Question 145 Tenter d'établir une communication entre `host-a` et `host-b`. Vérifier que les deux hôtes réussissent à communiquer.
Observer à l'aide de *wireshark*, le bridge `br-router` et vérifier que la communication est sécurisée.
Observer les paquets en clair sur les bridges `br-a` et `br-b`.

```
root@host-a:/$ netcat -v -l -s 192.168.0.1 -p 80
```

```
root@host-b:/$ echo "quit" | netcat -q 0 192.168.0.1 80
```

La configuration de notre réseau privé virtuel d'entreprise est terminée. Avec cette stratégie, la configuration de notre tunnel VPN ne nécessite pas la transmission de secret sur le réseau publique : seulement des signatures de clés RSA.

De plus, si un secret est volé (une clé RSA privée) depuis l'un des deux routeurs, alors celui-ci peut demander à la CA de révoquer le certificat et d'en fabriquer un nouveau, à partir d'un nouveau secret. De ce fait, les certificats ne sont pas copiés sur les routeurs comme nous l'avons fait mais sont stockés sur des serveurs de clés (qui peuvent aussi faire office d'autorité d'enregistrement et de certification). Ces serveurs recensent et contrôlent les certificats. Ils possèdent souvent une liste des certificats révoqués. À chaque nouvelle négociation, les extrémités du tunnel contactent ces serveurs, puis la validité du certificat et la signature de la CA sont vérifiées.