UNIVERSITE BADJI MOKHTAR – ANNABA - FACULTE SCIENCE DE L'INGENIORAT -
DEPARTEMENT D'INFORMATIQUE

Cours les protocoles sécurisés

Master 1 RSI

Dr. Amira Bendjeddou

2020/2021

Amira bendjeddou , LRS laboratory,amira.bendjeddou@univ-annaba.org

Table des matières

Objectifs du cours

1. L'objectif est de permettre aux étudiants de maitriser les protocoles de communication ayant des outils de sécurité intégrés
2. Acquérir des connaissances sur les protocoles sécurisés
3. Maitriser les outils nécessaires pour mettre en place un protocole de sécurité.
4. Manipulation et création d'un certificat numérique.
5. Configuration d'une autorité de certification

De ces trois objectifs découlent six grands chapitres

Chapitre 1 : **Les principes théoriques d'un protocole sécurisé**.
Chapitre 2 : **Les certificats numériques**
Chapitre 3 : **L'Annuaire LDAP**
Chapitre 4 : **Le protocole SSL/TLS**
Chapitre 5 : **Le protocole IPsec**
Chapitre 6 : **Le CRC dans les protocoles**

CHAPITRE 1

LES PRINCIPES THÉORIQUES D'UN PROTOCOLE SÉCURISÉ

Chapitre 1

Les principes théoriques d'un protocole sécurisé

1.1. Introduction

La communication par internet permet des échanges d'informations sous des formats différents (textes, sons, vidéos, images) au travers d'outils spécifiques. Ces outils de communication permettent la création de liens et favorisent la communication instantanée quelque soit l'heure, le lieu. Toutes les communications Internet utilisent le protocole de transmission TCP/IP (*Transmission Control Protocol/Internet Protocol*).

Le TCP/IP permet d'envoyer des informations d'un ordinateur à un autre au travers d'une grande variété d'ordinateurs intermédiaires et de réseaux distincts avant qu'elles atteignent leurs destinations. La grande flexibilité de TCP/IP à conduit à son adoption mondiale en tant que protocole de base pour les communications Internet et Intranet. Le fait que TCP/IP permettent aux informations de transiter par de nombreux ordinateurs intermédiaires rend possible à une tierce-partie d'interférer avec les communications des façons suivantes :

- **Ecoute clandestine :**

Les informations ne seront pas modifiées, elles restent intactes, mais leur confidentialité est compromise. Par exemple, quelqu'un peut intercepter votre numéro de carte de crédit, enregistrer des conversations sensibles ou intercepter des informations confidentielles.

- **Altération de données** :

Les informations en transit sont modifiées ou remplacées par d'autres informations puis envoyées au destinataire.

- **Usurpation d'identité (vole d'identité) :**

Il s'agit de se faire passer pour quelqu'un d'autre et de commettre des délits via Internet. Pour faire face à ce genre de problèmes, il a été nécessaire de créer un ensemble de mécanismes pour assurer les 4 fonctions de sécurité sur les documents et les nouveaux modes de communication électroniques. Ceux-ci peuvent reposer sur les protocoles sécurisés.

La cryptographie à clé publique ou à clé secrète est à la base de toute solution sécurisant le web.

1.2. Chiffrement

1.2.1. Principe: Le chiffrement est un processus de transformation des informations de façon à les rendre inintelligibles à toute personne autre que le destinataire. Elle consiste à appliquer une fonction mathématique (en fait c'est un ensemble de fonctions) avec des caractéristiques très particulières sur les informations. Cette fonction utilise une variable, la **clé de chiffrement**, qui est une suite de bits quelconque. Une fois les informations chiffrées, ils sont illisibles. Pour obtenir la version lisible, il faut les déchiffrer, c'est à dire appliquer une autre fonction mathématique, compatible avec la première, avec une autre variable, la **clé de déchiffrement**.

Ces 2 fonctions mathématiques sont appelées **algorithmes de chiffrement**. La valeur de la clé de déchiffrement dépend évidemment de la valeur de la clé de chiffrement et uniquement le possesseur de la clé de déchiffrement peut déchiffrer les informations transmises.

Lorsque l'on désire transmettre un document confidentiel à un correspondant à travers le réseau, on chiffre le document sur son poste de travail avec une clé de chiffrement et on envoie la version chiffrée. Le destinataire déchiffre le document sur son poste de travail avec la clé de déchiffrement, qu'il est le seul à connaître. Si une troisième personne intercepte le document durant le transfert, il ne pourra pas le déchiffrer car il ne connaîtra pas la valeur de la clé de déchiffrement.

Il y a deux grandes familles d'algorithmes de chiffrement : symétriques et asymétriques.

1.2.2. Chiffrement par clef symétrique

Avec le chiffrement par clef symétrique, le chiffrement peut être calculé avec la clef de déchiffrement et vice versa. Avec la plupart des algorithmes symétrique, la même clef est utilisée pour le chiffrement et le déchiffrement. De ce fait, pour que le texte chiffré ne soit lisible que par le destinataire, la valeur de cette clé doit être un secret partagé entre l'émetteur et le destinataire uniquement. Ceci explique le qualificatif de « clé secrète ». **DES** est l'algorithme symétrique historiquement le plus connu. La figure 1.1 illustre le principe du chiffrement symétrique.[1]

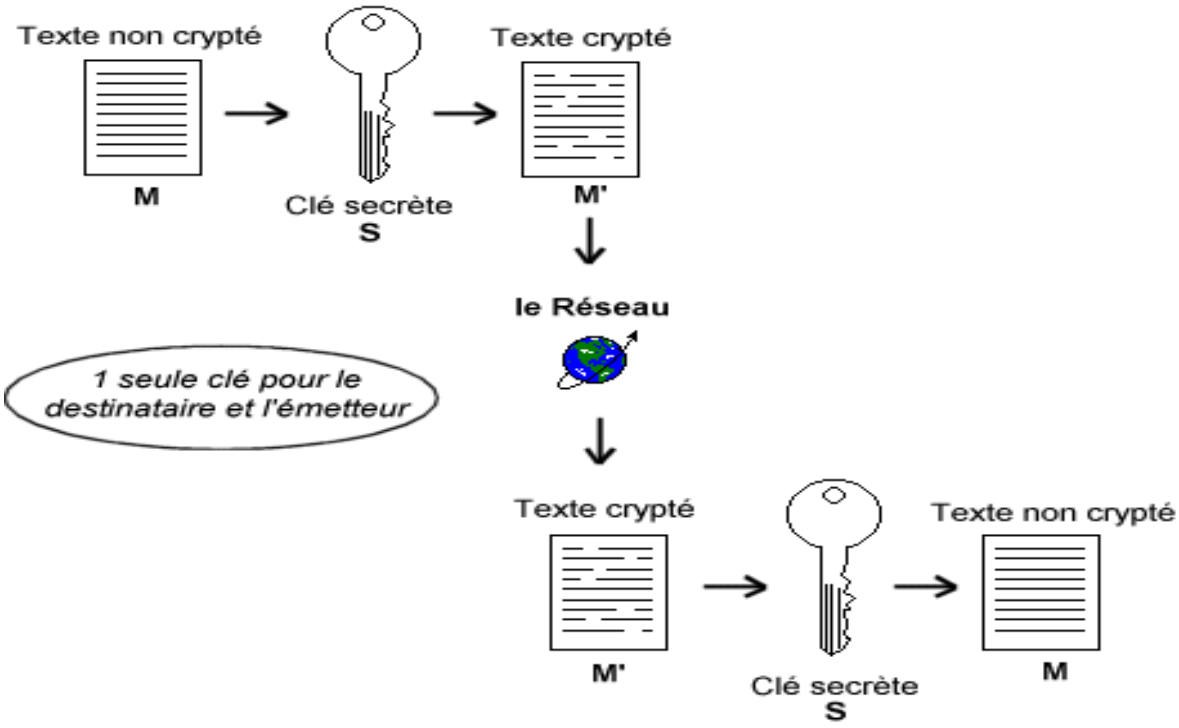


Figure 1.1 : Le chiffrement symétrique [2]

Les premiers algorithmes symétriques datent de la fin des années 70 et utilisent des fonctions mathématiques «simples ». L'avantage est que **les opérations de chiffrement et de déchiffrement sont rapides** à exécuter sur des ordinateurs classiques. Par contre, **le problème est la gestion des clés**. En effet, chaque clé que l'on utilise avec un correspondant doit être secrète et unique. On a donc autant de clés que de correspondants et il faut trouver un moyen d'échanger chaque clé secrète avec chaque correspondant de manière sûre (pas question d'utiliser le fax du secrétariat ou la messagerie électronique non sécurisée). Si ceci est possible entre un groupe restreint de personnes, c'est impossible à plus grande échelle, par exemple pour échanger des messages chiffrés avec tous nos correspondants sur Internet.

1.2.3. Chiffrement par clef publique

Le chiffrement à clef publique (également appelé chiffrement asymétrique) utilise une paire de clefs, l'une publique et l'autre privée, associées à une entité qui a besoin d'authentifier son identité de façon électronique ou de chiffrer des données .

La clef publique est publiée et la clef privée correspondante est conservée secrète par l'entité. **Les 2 clés (une pour chiffrer, l'autre pour déchiffrer)** sont crées ensembles avec une fonction mathématique. Elles **forment un couple**, l'une ne va pas sans l'autre, mais il est impossible avec une des clés de découvrir l'autre. Tout texte chiffré avec une des clés (de chiffrement ou de déchiffrement) peut être déchiffré avec l'autre clé (de déchiffrement ou de chiffrement) et uniquement avec celle-ci.

En pratique, pour utiliser des algorithmes de chiffrement asymétrique, il faut générer un couple de clés (l'une pour chiffrer, l'autre pour déchiffrer) pour chaque utilisateur. La personne le fera elle-même sur sa machine ou quelqu'un de confiance le fera pour elle. Elle

gardera sa clé de déchiffrement secrète, on l'appellera ainsi clé privée. A l'inverse elle rentra sa clé de chiffrement publique et la diffusera (on dit aussi la publiera) le plus largement possible. On la trouvera dans des annuaires électroniques par exemple. Ainsi **le couple de clés est formé d'une clé privée secrète pour déchiffrer et d'une clé publique pour chiffrer.**

RSA[3], du nom des 3 inventeurs Rivest, Shamir, Adleman, est l'algorithme de chiffrement asymétrique le plus répandu. Ce découplage entre clé publique et clé privée est très utile pour une utilisation « planétaire » du chiffrement.

Alors que les algorithmes symétriques obligent à échanger un secret, la clé secrète, avec chaque interlocuteur, là il suffit d'avoir un annuaire qui permette de trouver la clé publique de chaque internaute et ce système peut fonctionner entre tous les internautes. Quand un utilisateur voudra envoyer un message chiffré à un correspondant, il consultera l'annuaire qui lui indiquera la clé publique de son correspondant. Avec cette clé, il chiffrera le message. Celui-ci ne pourra être déchiffré qu'avec la clé privée du correspondant, donc que par le correspondant. Ainsi les propriétés des algorithmes asymétriques, sans intérêt au premier abord, vont permettre de s'affranchir du problème de la gestion des clés et ainsi d'envisager de déployer l'utilisation du chiffrement à très grande échelle. Mais il reste un problème.

Avec les algorithmes asymétriques, le temps pour les opérations de chiffrement et de déchiffrement est long**.** Ainsi sur un ordinateur courant, RSA (algorithme asymétrique) est de 100 à 1000 fois plus lent que le Triple DES (algorithme symétrique). La clé de session est un moyen pour atténuer ces mauvaises performances.

1.2.4. Clé de session

Pour entre autre, contourner les très mauvaises performances en temps de traitement des algorithmes asymétriques, une astuce est couramment utilisée dans les logiciels. Elle consiste à utiliser les deux types de chiffrement, asymétrique et symétrique, lors du même transmission.

Pour envoyer un message chiffré, le programme émetteur ne chiffrera pas le message complet avec un algorithme asymétrique. Il chiffrera en asymétrique, avec la clé publique du destinataire, uniquement un petit nombre aléatoire (quelques dizaines de caractères) choisi par lui. Ce nombre servira de clé secrète qui sera utilisée pour chiffrer de manière symétrique le texte. A l'arrivée, le programme du destinataire déchiffrera cette clé secrète, chiffrée « en asymétrique », avec sa clé privée. Munie de la clé sécrète ainsi déchiffrée, il déchiffrera ensuite le message, chiffré « en symétrique ». Dans ce processus, l'algorithme asymétrique n'est utilisé que sur quelques dizaines de caractères, cette opération sera donc relativement rapide. Le message, qui peut être un fichier de plusieurs gigabytes, sera lui chiffré et déchiffré avec un algorithme symétrique.

Le temps de traitement de l'ensemble sera donc du même ordre que si l'on avait utilisé que du chiffrement symétrique. D'autre part, l'introduction du chiffrement asymétrique permet de s'affranchir du problème de la gestion de la clé secrète des algorithmes symétriques. En effet cette clé secrète n'a pas besoin d'être connue avant la communication par le destinataire. Elle

est choisie par l'émetteur et le destinataire en prend connaissance en déchiffrant une partie de ce qu'il a reçu. Ce processus à deux étapes est employé par les outils sécurisés de messagerie, transfert de fichiers, navigation, … La clé secrète ne servant que pour l'opération en cours, est appelée **clé de session**. Au prochain transfert le programme de l'émetteur choisira une autre clé de session. Un intrus qui récupère le document chiffré ne pourra pas déchiffrer la clé de session car il ne possède pas la clé privée du destinataire ; et sans cette clé de session, il ne pourra pas déchiffrer le texte du message.

La figure 1.2 montre un exemple de transfert d'un texte chiffré avec utilisation d'une clé de session.

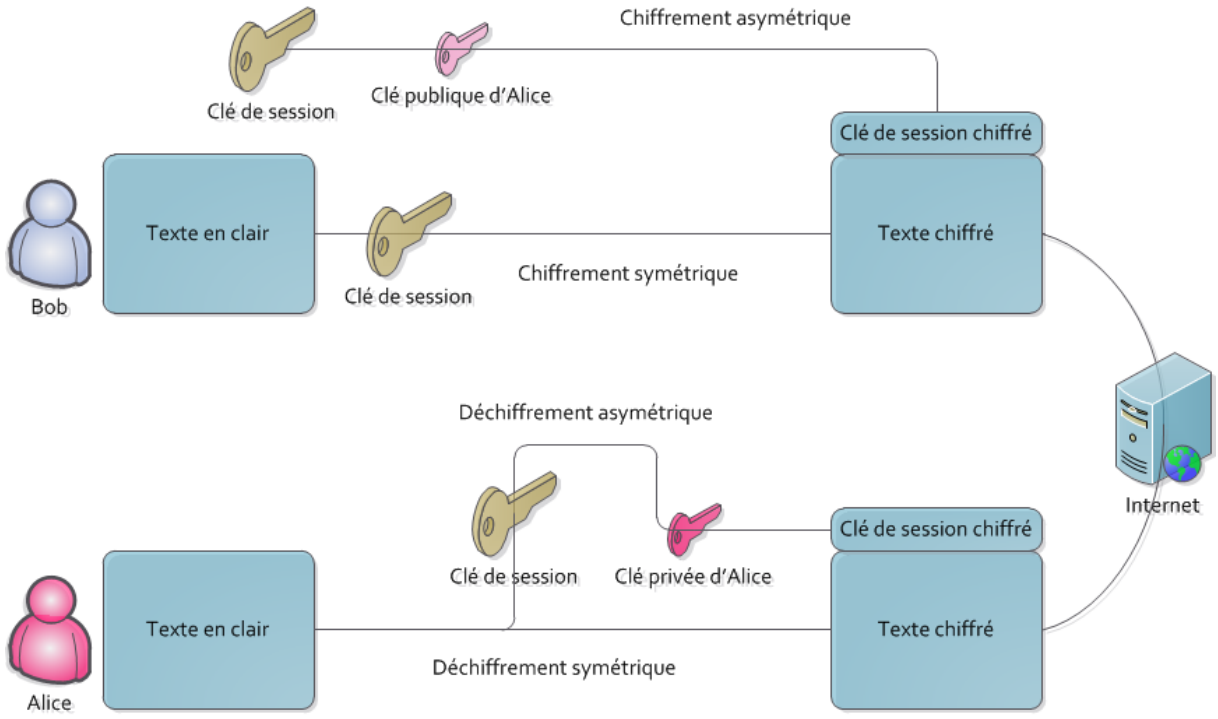


Figure 1.2. Un exemple de transfert d'un texte avec utilisation d'une clé de session.

1.3. Signature électronique

Les paragraphes précédents ont décrit comment est assurée la fonction de confidentialité avec le mécanisme de chiffrement. La signature électronique est un des mécanismes qui permettent d'assurer les fonctions d'authentification et d'intégrité. Il est utilisé en particulier dans la messagerie électronique. Pour générer une signature électronique, il faut dans un premier temps d'utiliser une **fonction de hachage**. C'est une fonction mathématique qui à partir d'un texte de n'importe quelle longueur génère un nombre, suite de bits de taille fixe, bien inférieure à la taille du texte initial. Cette fonction est telle que si un bit du texte d'origine est modifié, le résultat de la fonction sera différent. Cette suite de bits est ainsi appelée **condensé** ou **empreinte.**

MD5 (MD pour Message Digest) est une fonction de hachage très répandue, elle crée une empreinte de 128 bits. **SHA** (Secure Hash Algorithm), autre fonction, crée des empreintes de 160 bits.

Avant d'envoyer le message, l'outil logiciel émetteur calcule l'empreinte du message, résultat d'une fonction de hachage appliquée au message. Il chiffre ensuite cette empreinte par un algorithme asymétrique avec sa clé privée. Ce résultat est appelé signature électronique. Avant l'envoi, cette signature est ajoutée au message, qui devient un message signé. Le logiciel du destinataire qui reçoit l'ensemble déchiffre cette empreinte chiffrée avec la clé publique de l'émetteur. Puis il recalcule la fonction de hachage sur le message reçu et compare le résultat avec l'empreinte déchiffrée. Si les deux sont égaux, cela veut dire que le message n'a pas été modifié durant le transfert et que l'émetteur est authentifié. En effet, si le message a été modifié, les 2 empreintes seront différentes. De plus, être capable de déchiffrer, avec la clé publique d'une personne, une empreinte chiffrée, prouve que cette empreinte a obligatoirement été chiffrée avec la clé privée de la personne, clé que seul possède l'émetteur. Cela authentifie donc l'émetteur [3]. Comme montré dans la figure 1.3

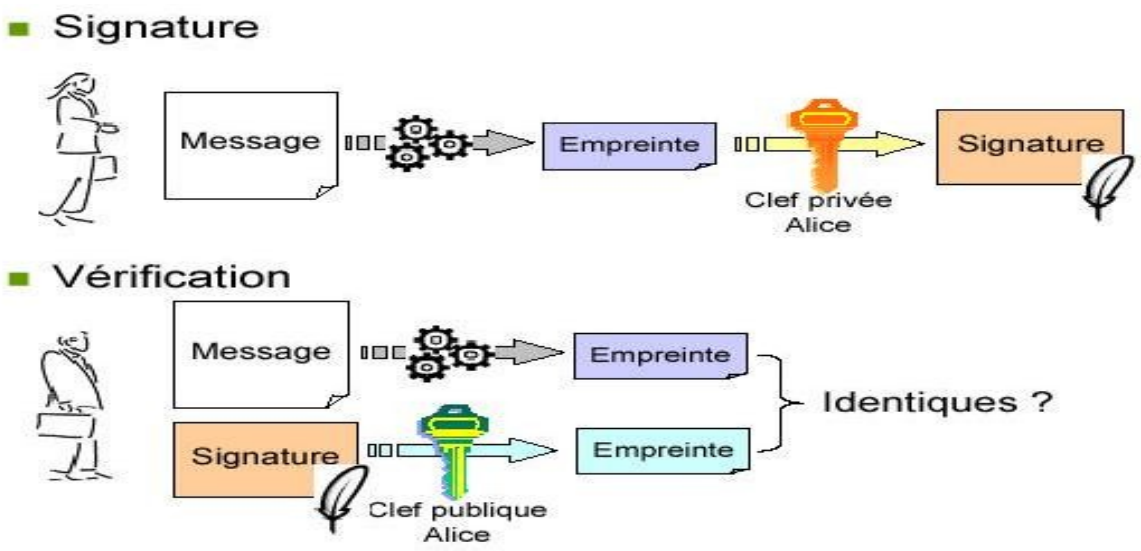


Figure 1.3 : la signature électronique [4]

CHAPITRE 2

LES CERTIFICATS NUMÉRIQUES

Chapitre 2

Les certificats numériques

Nous avons décrit dans le premiers chapitre les mécanismes qui permettent d'assurer les 3 fonctions de base de sécurité avec le couple de clés privée-publique et les algorithmes de chiffrement asymétriques. Mais il y a une énorme lacune dans les raisonnements précédents. On a considéré qu'un utilisateur connaissait la clé publique d'une personne simplement en consultant un annuaire ou un serveur Web et ainsi il la considérait comme vraie. Mais qu'est-ce qui garantit que la clé publique qu'un utilisateur a ainsi récupérée est la bonne ? Il ne faut pas oublier que tout ceci fonctionne de manière électronique, sur Internet, sans contact direct donc sans moyen visuel de reconnaissance d'une personne. C'est le rôle des **certificats.**

2.1. Définition d'un certificat

Un certificat est l'équivalent d'une carte d'identité ou d'un passeport. Un passeport contient des informations concernant son propriétaire (nom, prénom, adresse, etc.), la signature manuscrite, la date de validité, ainsi qu'un tampon et une présentation (forme, couleur, papier) qui permettent de reconnaître que ce passeport n'est pas un faux, qu'il a été délivré par une autorité bien connue. Le certificat numérique ou électronique, résultat d'un traitement fixant les relations qui existent entre une clé publique, son propriétaire et l'application pour laquelle il est émis.

- Pour une personne il assure son identité
- Pour une application, il assure que celle-ci n'as pas été détournée de ses fonctions
- Pour un site, il offre la garantie lors d'un accès vers celui –ci que l'on est bien sur le site auquel on veut accéder

Le format reconnu actuellement est le **format X509V3.** C'est un petit fichier, qui contient au moins les informations suivantes [5]:

- Le nom de l'autorité (de certification) qui a créé le certificat
- Le nom et le prénom de la personne
- Son entreprise (CNRS par exemple)
- Son service (au CNRS, le nom du laboratoire)
- Son adresse électronique
- Sa clé publique
- Les dates de validité du certificat
- Des informations optionnelles
- Une signature électronique

Cette **signature électronique** est calculée sur les informations contenues dans le certificat comme dans le cas d'un message électronique. **La signature est l'empreinte de ces informations chiffrée avec la clé privée de l'autorité de certification qui a délivré ce certificat.**

2.2. Types de certificats [7]

Le type de certificats électroniques dépend du support qui l'héberge. Ainsi, nous pouvons distinguer trois (03) types de certificats :

- le certificat serveur
- le certificat personnel ou client
- le certificat IP SEC (Internet Protocol Security) ou VPN

a. Le certificat serveur

Ce type de certificat est installé sur un serveur. Il est le garant de l'identité du serveur et la sécurité de la session établie par un utilisateur.

Le certificat serveur le plus répandu actuellement est le certificat SSL ou « Security Socket Layer ». Il permet d'assurer l'authenticité d'une URL et de garantir la sécurité des transactions effectuées par les internautes.

b. Le certificat personnel ou certificat client

Le certificat client est enregistré sur un ordinateur ou sur un conteneur, comme les clés USB ou carte à puce, appartenant à une personne physique. En analogie avec la carte d'identité d'une personne physique, il sert à identifier l'utilisateur et définir ses droits d'accès aux différentes informations partagées sur le réseau.

c. Le certificat IP SEC (Internet Protocol Security) ou VPN

Le certificat IPSEC est quant à lui hébergé sur un équipement réseau. Il a pour but de chiffrer l'ensemble des informations transmises sur les réseaux usant des protocoles internet. Comme les deux précédents certificats, il sert également d'identifiant pour le composant de réseau. Il permet ainsi de rendre privé l'ensemble des flux transitant entre deux équipements réseaux.

2.3. Autorité de certification et infrastructure de gestion de clés

2.3.1. Définition d'une autorité de certification

Une autorité de certification (AC) est un organisme reconnu comme étant compétent pour délivrer des certificats à une population auprès de laquelle elle a toute confiance et en assurer la validité. Elle s'engage sur l'identité d'une personne au travers du certificat électronique qu'elle lui remet. Une autorité de certification est responsable (vis-à-vis de ses clients, mais aussi de toute personne se fiant à un certificat électronique qu'elle a émis) de l'ensemble du

processus de certification et, par voie de conséquence, de la validité des certificats qu'elle émet. Par ailleurs, c'est elle qui définit la politique de certification et la fait appliquer.

La confiance que l'on accordera à un certificat va dépendre du sérieux de l'autorité qui l'aura délivré. De plus, on voit très bien le risque encouru par une entreprise ou un organisme dont la carte d'identité des employés aurait été créé par une autorité ni habilitée, ni contrôlée par elle-même. **Le choix de l'autorité de certification dans une organisation ou une entreprise est une décision stratégique.**

2.3.2. Infrastructure de gestion de clé [6]

Quelle que soit l'autorité de certification choisie, il faut faire d'autres choix. Comme il existe un circuit de procédures et de vérifications, des personnes habilitées, etc., pour délivrer les cartes d'identité, il faut mettre l'équivalent en place. Il faut ainsi décider qui va recueillir et vérifier les informations données par une personne lorsqu'elle va demander un certificat, suivant quelles procédures, qui va créer le certificat, qui va le lui délivrer, pour quelle durée, où va-t-il être stocké, où va-t-on pouvoir récupérer les certificats d'autres personnes, etc.

Il faut définir ce que l'on appelle une architecture de gestion des certificats. **IGC** (Infrastructure de Gestion de Clés), et **PKI** (Public Key Infrastructure) sont les deux sigles les plus connus pour la désigner. Les normes internationales décrivent les différents éléments fonctionnels d'une IGC. En simplifiant, l'architecture est constituée de :

2.3.2.1. D'autorités d'enregistrement

Ce sont les mairies ou les préfectures, c'est à dire les guichets auxquels s'adressent les utilisateurs qui désirent obtenir un certificat. L'autorité d'enregistrement vérifie l'identité du demandeur, s'assure que celui-ci possède bien un couple de clés privée-publique et récupère la clé publique du demandeur. Elle transmet ensuite ces informations (informations d'identité du demandeur ainsi que sa clé publique) à l'autorité de certification. Une autorité d'enregistrement peut être un secrétariat avec une personne habilitée qui:

- Vérifie une pièce d'identité présentée par le demandeur (action 1 dans la figure 2.1 ci après).
- Crée un couple de clés pour l'utilisateur (action 2 dans la figure 2.1). Ceci est réalisé avec un logiciel spécifique sur un ordinateur dédié et déconnecté du réseau (par prudence).
- Remet une disquette à l'utilisateur qui contient la clé privée générée (action 3).
- Garde la clé publique de l'utilisateur (action3).
- Transmet avec un message électronique signé une demande de certificat (contenant les informations d'identité et la clé publique du demandeur) à l'autorité de certification (action 4) La transmission des demandes doit se faire de manière sécurisée, personne ne doit pouvoir modifier la demande durant le transport par exemple. Pour ce faire, les autorités d'enregistrement ainsi que l'autorité de certification ont des certificats et utilisent les mécanismes d'authentification,

d'intégrité et de confidentialité pour communiquer entre eux. Ceci n'est qu'un exemple de procédures, elles peuvent être très différentes selon l'organisation que l'on met en place.

2.3.2.2. D'une autorité de certification

Celle-ci reçoit les demandes de création de certificats venant des autorités d'enregistrement. Elle vérifie la validité de la signature des messages reçus, garantie de l'intégrité de la demande et de l'authentification des émetteurs. **Elle crée les certificats et signe ces certificats en utilisant sa clé privée.** Elle envoie les certificats aux utilisateurs et en parallèle les transmet au service de publication. Une autorité de certification a donc un couple de clé privée-publique pour signer les certificats. Si la clé publique est la plus largement connue, la clé privée est au contraire ultra confidentielle et doit être très bien protégée. Car si un tiers prend connaissance de cette clé, il pourra générer des faux certificats.

Le travail de l'autorité de certification peut être assuré par une personne habilitée d'un service central qui possède la clé privée de l'autorité de certification ou plutôt le mot de passe et la clé du coffre qui permet d'accéder et d'utiliser cette clé. La génération des certificats peut être réalisée par un logiciel spécifique sur un ordinateur portable entreposé dans un coffre fort. Sur cet ordinateur est aussi stockée la clé privée de l'autorité de certification.

La personne habilitée « autorité de certification » reçoit les demandes de certificat par messagerie électronique sur un poste connecté au réseau (action 4). A chaque demande, elle vérifie la signature électronique de l'autorité d'enregistrement et transfère les informations sur une disquette. Elle ouvre le coffre, démarre l'ordinateur portable, insère la disquette (action 5), génère le certificat avec une signature en utilisant la clé privée de l'autorité de certification et transfère le certificat créé sur la disquette (action 6). Elle remet l'ordinateur dans le coffre et referme ce dernier. Avec la disquette, sur le poste connecté au réseau, elle envoie par messagerie électronique le certificat à l'utilisateur (action 7) et au service de publication (action 8).

Là encore ce n'est qu'un exemple de procédures. Le choix et la mise au point de ces procédures, dont la description est fastidieuse est néanmoins fondamental pour la fiabilité de l'ensemble. Plus que les techniques de chiffrement, ce sont ces procédures qui sont un des talons d'Achille des certificats. Car si ces procédures sont mal définies ou mal appliquées, un intrus pourra par exemple s'emparer de la clé privée de l'autorité de certification et générer des faux certificats qui feront écrouler l'ensemble.

a. Types d'autorités de certification

Les principales autorités de certification américaines et européennes sont divisées en deux catégories :

- Les autorités de certification reconnues.
- Les autorités de certification non reconnue.

Parmi les autorités reconnues, on peut citer : VeriSign, thawte, Entrust, Baltimore, Globalsign, ect. Parmi les autorités non reconnues on peut citer : Certinomis, ChamberSign, E-Trust, E-Certify.

2.3.2.3. D'un service de publication

Celui-ci rend disponible les certificats émis par l'autorité de certification (action 9). Il publie aussi la liste des certificats valides et des certificats révoqués. Concrètement ce service peut-être rendu par un annuaire électronique LDAP ou un serveur Web accessibles par l'Internet.

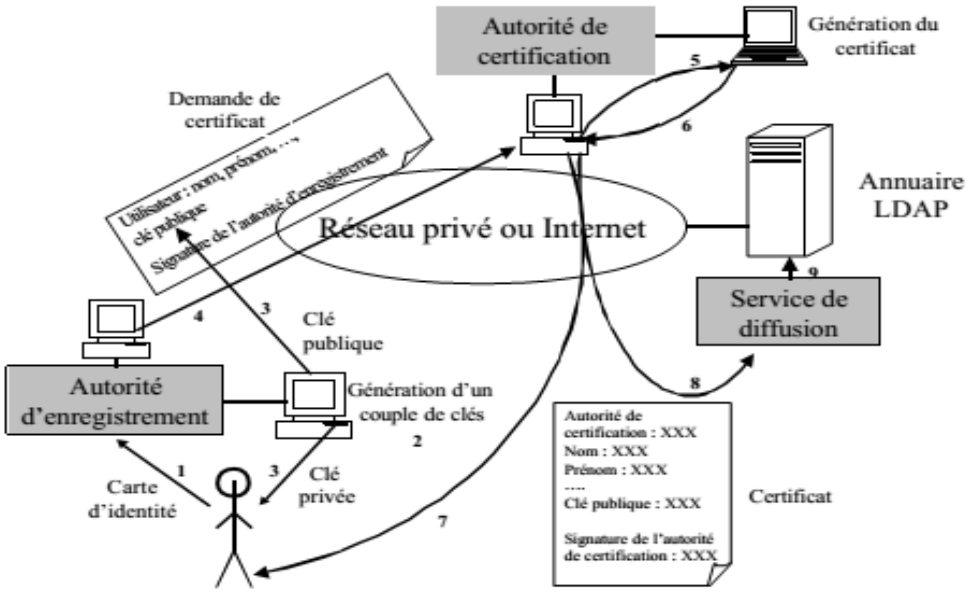


Figure 2.1 : Exemple d'étapes pour la création d'un certificat [6].

2.4. Les classes de certificats [7]

Il est courant de catégoriser les certificats par leurs classes. Cette classe permet de connaître quel niveau de validation (et donc de sécurité) qu'on peut attendre des certificats.

Classe 1 : Un certificat électronique de classe 1 ne garantit que l'existence de l'adresse e-mail de son titulaire. Il n'assure donc pas l'identité du détenteur du certificat et par ailleurs, n'effectue aucun contrôle à ce sujet.

Classe 2 : preuve de l'identité requise (photocopie de carte d'identité par exemple ou bien numéro SIRET/SIREN et nom de domaine). Le contrôle se fait à distance sur remise de justificatif.

Classe 3 : un certificat ne peut être délivré que dans le cadre d'une présentation physique du demandeur (contrôle face à face du client).

Classe 3+ : la même chose que la classe 3 avec en plus la délivrance de la clef privée et du certificat associé sur support physique.

2.5. Forme de certificats électronique

Il existe 2 solutions différentes :

2.5.1. La solution logicielle

Le certificat est téléchargé et stocké sur le disque dur de l'ordinateur. Ce n'est pas la solution la plus sécurisée car le certificat peut être copié ou supprimé par une personne mal intentionnée. Le certificat logiciel est donc fortement déconseillé.

2.5.2. La solution matérielle avec 2 formes différentes

a. **Le certificat sur clé USB :** la clé se connecte directement sur le port USB du PC. Par

contre, sur PC station, si le port USB est à l'arrière de votre machine, vous aurez des difficultés à connecter votre clé USB à chaque utilisation. Si le port USB est en face avant, vous risquez de casser la clé USB qui dépasse de votre poste. De plus, ce n'est pas un support personnalisable.

b. **Le certificat sur carte à puce** : un lecteur de cartes est nécessaire. Sur un ordinateur

portable, il est souvent inclus, par contre sur PC station, il faudra s'en procurer un. La carte à puce est personnalisée au nom du bénéficiaire et de l'organisme auquel il est rattaché. Ce qui est très utile lorsque vous êtes plusieurs à posséder des certificats électroniques au sein de votre entreprise ou si vous êtes porteur de plusieurs certificats, pour le compte de plusieurs entités. Parce que le certificat est logé sur carte à puce ou sur clé USB, il vous apporte un niveau de sécurité plus élevé que le certificat logiciel et répond à vos besoins de souplesse et de mobilité :

- Votre certificat et les éléments secrets associés sont stockés sur une puce (carte ou clé), hors de votre poste de travail ;
- ils sont protégés par un code confidentiel connu de vous seul (et personnalisable) ;
- ils vous accompagnent sur vos différents lieux de travail.

2.6. Utilisation des certificats

L'utilisation de la cryptographie à clé publique ou à clé symétrique est très répandue. Cette utilisation est faite à travers les certificats pour les raisons citées ci-dessus.

Parmi les applications et protocoles utilisant les certificats, nous pouvons citer :

- Le courrier électronique sécurisé (S/MIME).
- Les protocoles SSL/TLS

- Réseaux virtuels privés : IPsec.
- Niveau applicatifs (en remplacement d'une authentification par mot de passe de l'utilisateur).

2.7. Les Autorités de certification et les Hiérarchies de confiance [8][9][10][11]

Les autorités de certification (AC) sont des entités qui valident l'identité et l'émission des certificats. Elles peuvent être des organismes indépendants ou des entreprises qui utilisent leur propre logiciel d'émission de certificats (tel que *Red Hat Certificate System*).

Tout logiciel client ou serveur qui supporte les certificats maintient une liste des certificats d'AC de confiance. Ces certificats d'AC déterminent quels autres certificats le logiciel peut valider - en d'autres mots, dans quels émetteurs de certificats le logiciel peut avoir confiance. Dans le cas le plus simple, le logiciel ne peut valider que les certificats émis par une des AC pour lesquelles il possède le certificat. Il est également possible pour un certificat d'une AC de confiance de faire parti d'une chaîne de certificats d'AC, chacun émis par l'AC parente dans la hiérarchie de certificat

2.7.1. Hiérarchies d'AC

Dans les grandes organisations, il peut être judicieux de déléguer la responsabilité de l'émission des certificats à plusieurs autorités de certification. Par exemple, quand le nombre de certificats requis peut être trop important à maintenir par une seule AC, il est possible de déléguer la responsabilité de l'émission de certificats à des AC subordonnées.

Le standard X.509 inclus un modèle de paramétrage d'une hiérarchie d'AC comme celle montrée dans la Figure 2.2

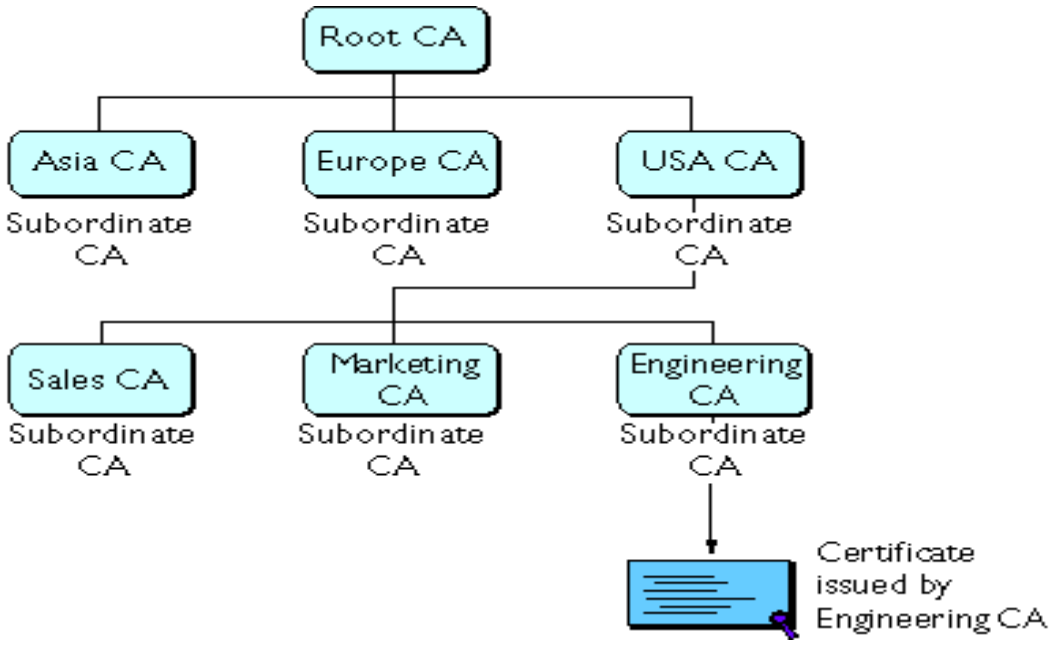


Figure 2.2 : hiérarchie d'AC[8][9][10][11]

Dans ce modèle, l'AC racine est au sommet de la hiérarchie. Le certificat de l'AC racine est un « certificat auto-signé » : c'est-à-dire que le certificat est signé numériquement par la même entité « L'AC racine ». Les AC directement subordonnées à l'AC racine on des certificats d'AC signés par l'AC racine. Les AC se trouvant sous les AC subordonnées dans la hiérarchie ont leurs certificats signés par les plus hautes AC subordonnées.

La figure 2.2 ne montre qu'un exemple d'une hiérarchie d'AC ; beaucoup d'autres hiérarchies sont possibles.

2.7.2. Chaînes de certificat

Les hiérarchies d'AC se reflètent dans les chaînes de certificats. Une **chaîne de certificats** est une série de certificats émis par des AC successives. La figure 2.3 montre un exemple d'une chaîne de certificats.

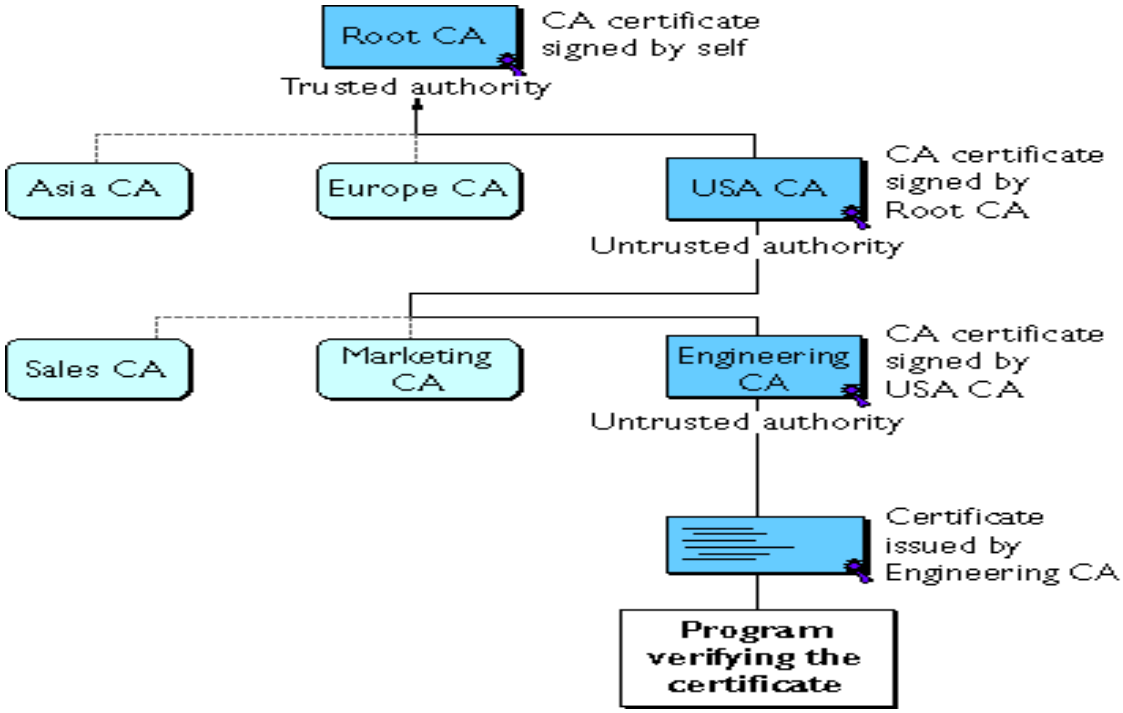


Figure 2.3. Chaîne de certificat[8][9][10][11]

Une chaîne de certificats trace le chemin des certificats d'une branche de la hiérarchie jusqu'à la racine. Dans une chaîne de certificats, on a donc :

- chaque certificat est suivi par le certificat de son émetteur.
- chaque certificat contient le nom (DN) de l'émetteur du certificat, qui est identique à celui du nom du sujet du certificat suivant dans la chaîne.

Dans la figure 2.3, Le certificat *Engineering CA* contient le DN de l'AC (qui est *USA CA*), qui a émis le certificat. Le DN de USA *CA* est également le nom du sujet du prochain certificat dans la chaîne.

- Chaque certificat est signé avec la clef privée de son émetteur. La signature peut être vérifiée avec la clef publique du certificat de l'émetteur, qui est le prochain certificat dans la chaîne.

Dans la figure 2.3, la clef publique dans le certificat de *USA CA* peut être utilisée pour vérifier la signature numérique de *USA CA* dans le certificat de *Engineering CA*.

2.7.3. Vérification d'une chaîne de certificat

La vérification de la chaîne de certificats est le processus permettant de s'assurer qu'une chaîne de certificats donnée est bien formée, proprement signée et sécurisée.
Les logiciels Red Hat utilisent la procédure suivante pour former et vérifier une chaîne de certificats, en commençant par le certificat présenté pour l'authentification :

1. La période de validité du certificat est vérifiée par rapport à la date actuelle fournie par l'horloge système du vérificateur.

1. Le certificat de l'émetteur est localisé. La source peut être une base de certificats locale du vérificateur (du client ou du serveur) ou une chaîne de certificats fournit par le sujet (par exemple, par une connexion SSL).
2. La signature du certificat est vérifiée à l'aide de la clef publique du certificat de l'émetteur.
3. Si le certificat de l'émetteur est présent dans les certificats de confiance du vérificateur, la vérification s'arrête avec succès à cette étape. Autrement, le certificat de l'émetteur est vérifié pour être certain qu'il contient les indications appropriées concernant les AC subordonnées dans l'extension du type de certificat de Red Hat, et la chaîne de vérification recommence depuis l'étape 1, mais avec le nouveau certificat. La figure 2.4 présente un exemple de ce processus.

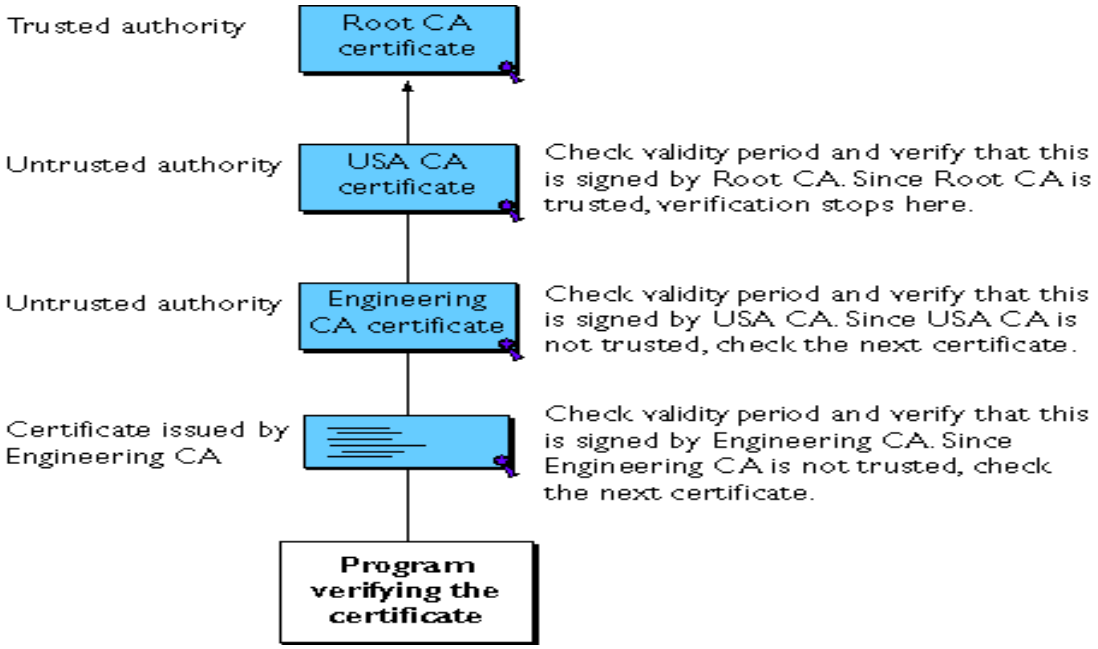


Figure 2.4 vérification d'une chaine de certificat (cas 1) [8][9][10][11]

La figure 2.4 décrit ce qui se passe lorsque seule l'AC racine est incluse dans la base locale de certificats du vérificateur. Si un certificat d'une des AC intermédiaires présente dans cette figure, telle qu'Engineering *CA*, est trouvé dans la base locale de certificats du vérificateur, la vérification s'arrête à ce certificat, comme décrit à la figure 2.5.

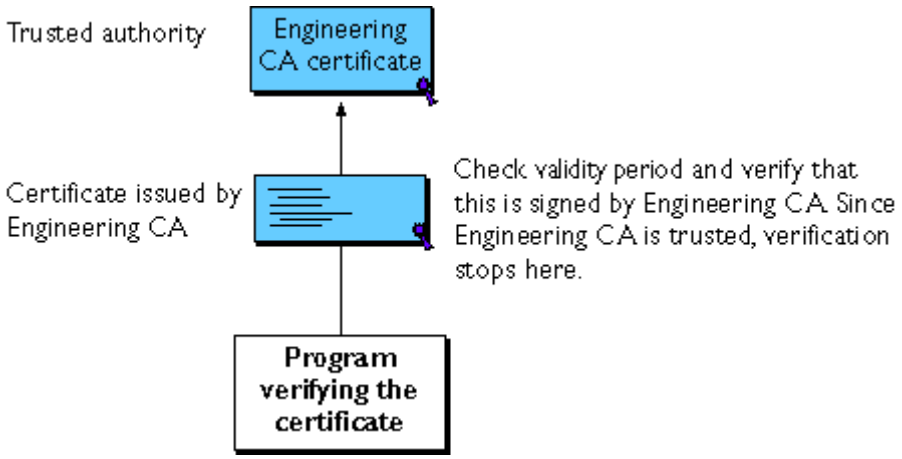


Figure 2.5 vérification d'une chaine de certificat (cas 2) [8][9][10][11]

Des dates de validité expirées, une signature invalide ou l'absence d'un certificat pour l'AC émettrice à n'importe quelle étape de la chaîne de certificats provoquera un échec de l'authentification. Par exemple, la figure 2.6 montre l'échec d'une vérification si le certificat de l'AC racine ou celui d'une AC intermédiaire ne sont pas présent dans la base locale de certificats du vérificateur.

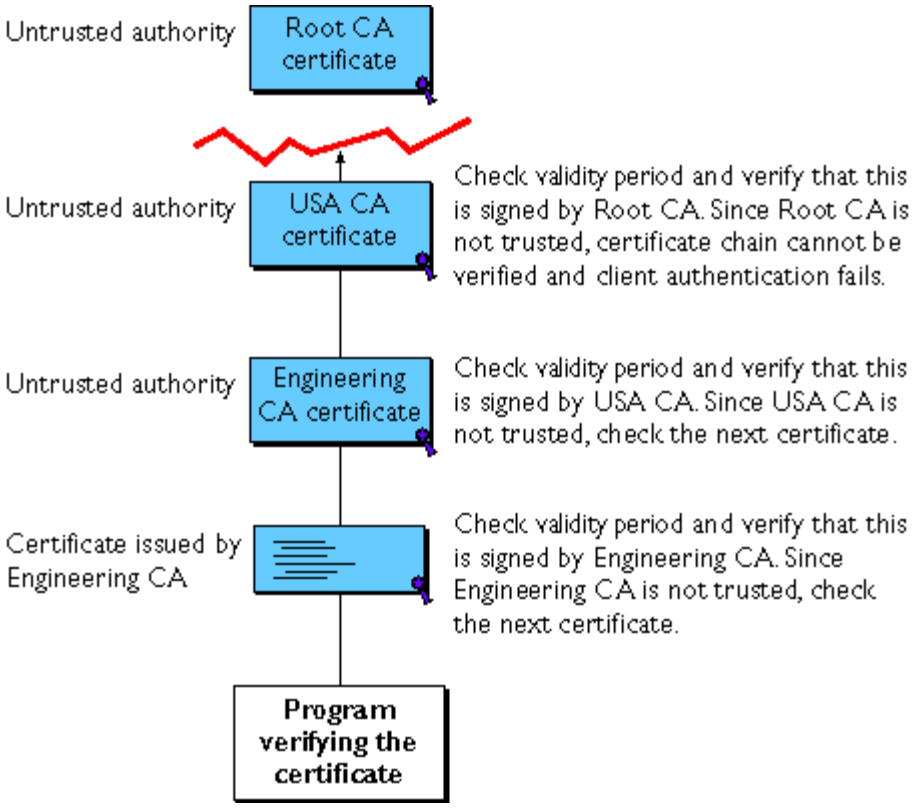


Figure 2.6 l'échec d'une vérification d'une chaine de certificats[8][9][10][11]

2.8. Gestion des certificats en cours (mise à jour, renouvellement et révocation de certificat) [12]

Un certificat peut être l'objet de plusieurs opérations : mise à jour, renouvellement et révocation :

2.8.1. Révocation d'un certificat

La révocation est la mise hors service d'un certificat avant qu'il ait atteint sa date d'expiration. Plusieurs causes peuvent être à l'origine de la révocation du certificat notamment la compromission de la clé privée associée à la clé publique du certificat. Dans ce cas, le certificat révoqué est publié afin que les autres utilisateurs en prennent connaissance lors de la validation de ce certificat.

2.8.2. Mise à jour d'un certificat

La mise à jour d'un certificat est une autre forme de révocation, qui concerne la modification (ajout/suppression) de certain champs du certificat, y compris la clé publique du certificat sans que le certificat soit arrivé à sa date d'expiration ou que le certificat soit révoqué suite à une compromission. Dans ce cas, l'autorité de certification ayant délivré le certificat, va

l'invalider en le révoquant, puis va générer un nouveau certificat contenant les modifications nécessaire, avec une nouvelle période de validité et une nouvelle signature.

2.8.3. Renouvellement d'un certificat

Le renouvellement d'un certificat est la régénération d'un même certificat une fois sa date d'expiration atteinte. Donc seule la période de validité ainsi que la signature du certificat changent. Les autres informations contenues dans le certificat sont supposées rester inchangées.

Implicitement, la mise à jour ou le renouvellement d'un certificat sous-entend en premier lieu la révocation du certificat, puis la génération d'un nouveau certificat contenant les modifications nécessaire, avec le nouveau certificat contenant obligatoirement un numéro de série différent de celui du certificat révoqué.

CHAPITRE 3

L'ANNUAIRE LDAP

Chapitre 3

L'Annuaire LDAP

3.1. Introduction

Comme nous l'avons déjà montré, il est nécessaire de connaître le certificat de son correspondant (qui renferme sa clé publique), pour communiquer de manière sécurisée avec lui. Les outils informatiques sécurisés stockent les certificats qu'ils utilisent. Ainsi ils n'ont besoin d'obtenir le certificat d'une personne qu'une seule fois. Ils intègrent aussi des mécanismes qui transmettent les certificats automatiquement. Si Philippe envoie un message signé à Nicole, son outil de messagerie sécurisé enverra en même temps le certificat de Philippe. L'outil de messagerie de Nicole le récupérera avec le message envoyé et le stockera. Donc après quelques échanges les outils de chaque utilisateur auront les certificats des principaux correspondants des utilisateurs. Néanmoins, on peut avoir besoin de communiquer avec des personnes avec lesquelles on n'a jamais échangé des messages. Pour cela il faut un annuaire.

Ce besoin d'annuaire est encore plus fondamental pour tous les serveurs réseaux (serveurs Web par exemple) qui voudront contrôler les certificats des utilisateurs qui les accèdent. Il est préférable qu'ils se réfèrent à une même base d'informations à jour qui contienne les certificats, un annuaire, au lieu que chaque serveur gère sa propre base de données locale de certificats. Le standard d'annuaire reconnu et implémenté par les principaux outils est maintenant LDAP, Light Directory Access Protocol. Il intègre en standard dans le format de ses enregistrements un champ destiné à contenir les certificats d'une personne. Il y a donc tous les éléments nécessaires pour diffuser les certificats sur l'Internet.

3.2. Définition du LDAP [13]

Le protocole LDAP est un protocole de la couche Application (7) du modèle OSI. Il est conçu pour fonctionner au-dessus de TCP, lui même au-dessus d'IP. Par conséquent, les communications avec un annuaire LDAP sont en mode connecté, et les paquets échangés ont une garantie d'intégrité. LDAP est ainsi une version allégée du protocole DAP, d'où son nom de Lightweight Directory Access Protocol.

3.3. Définition d'un annuaire électronique :

Les annuaires électroniques sont un type de base de données spécialisées permettant de stocker des informations de manière hiérarchique et offrant des mécanismes simples pour rechercher l'information, la trier, l'organiser selon un nombre limité de critères. Ainsi le but d'un annuaire électronique est approximativement le même que celui d'un

annuaire papier, si ce n'est qu'il offre une grande panoplie de possibilités que les annuaires papier ne sauraient donner.

3.4. Caractéristiques des annuaires électroniques [14]

Les annuaires électroniques possèdent un grand nombre d'avantages sur leurs "cousins de papier" :

- ils sont **dynamiques** : la mise à jour d'un annuaire électronique est beaucoup plus simple (et nettement moins coûteuse) à réaliser que celle d'un annuaire papier. Ainsi un annuaire en ligne (disponible sur le réseau) sera à jour beaucoup plus rapidement, d'autant plus que les personnes recensées dans l'annuaire peuvent elles-mêmes modifier les informations les concernant (si elles sont habilitées à le faire).
- ils sont **sûrs** : les annuaires en ligne disposent de mécanismes d'authentification des utilisateurs grâce à un mot de passe et un nom d'utilisateur ainsi que des règles d'accès permettant de définir les branches de l'annuaire auxquelles l'utilisateur peut accéder
- ils sont **souples** : ils permettent ainsi de classer l'information selon des critères multiples contrairement aux annuaires papiers, imprimés une fois pour toute pour permettre de rechercher selon un critère figé (en général l'ordre alphabétique selon le nom).

3.5. Différence avec une base de données [14]:

1. **On lit plus souvent** un annuaire qu'on ne le met à jour. Contrairement à un SGBD, un annuaire n'est pas fait pour stocker des informations constamment en mouvement. Il est logique de le structurer différemment et d'organiser les données de manière arborescente (sur un SGBD, la structuration est relationnelle).
2. un annuaire fournit **une méthode de consultation standardisée**. Le SQL est, certes, standardisé, mais chaque SGBD (Oracle, SQLServer, MySQL, PostgreSQL ...) a sa propre couche de connexion et ses propres fonctions. les annuaires doivent être compacts et reposer sur un protocole réseau léger

3. Un annuaire doit comporter des mécanismes permettant de rechercher facilement une information et d'organiser les résultats
4. les annuaires doivent pouvoir être répartis. Cela signifie qu'un serveur d'annuaire doit comporter des mécanismes permettant de coopérer, c'est-à-dire d'étendre la recherche sur des serveurs tiers si jamais aucun enregistrement n'est trouvé
5. Un annuaire doit être capable de gérer l'authentification des utilisateurs ainsi que les droits de ceux-ci pour la consultation ou la modification de données.

3.6. La normalisation des annuaires :

Ainsi un annuaire est un serveur remplissant les conditions décrites ci-dessus, mais l'implémentation peut être totalement différente d'un serveur à un autre, c'est pourquoi il a été nécessaire de définir une interface normalisée permettant d'accéder de façon standard aux différents services de l'annuaire. C'est le rôle du protocole LDAP (*Lightweight Directory Access Protocol*), dont le rôle est uniquement de fournir un moyen unique (standard ouvert) d'effectuer des requêtes sur un annuaire (compatible LDAP).

3.7. Principes et concepts [13,15]

Un serveur LDAP agit en tant qu'intermédiaire entre une source de données et un client. Le client ne verra, ni ne connaîtra l'existence du stockage des données. En effet elles peuvent être dans un fichier plat ou dans une base de données. De plus, découpler les messages du stockage permet d'avoir plusieurs serveurs et un même système de stockage.

LDAP est asynchrone, c'est-à-dire que si le client émet plusieurs requêtes successivement, elles peuvent arriver dans un ordre différent.

Les modèles LDAP représentent les services que propose le serveur au client.

Il fournit :

- un ***modèle d'information*** définissant le type de données contenues dans l'annuaire,
- un ***modèle de nommage*** définissant comment l'information est organisée et référencée,
- un ***modèle fonctionnel*** qui définit comment on accède à l'information,
- un ***modèle de sécurité*** qui définit comment données et accès sont protégés,
- un ***modèle de duplication*** qui définit comment la base est répartie entre serveurs,
- des ***APIs*** pour développer des applications clientes,
- ***LDIF***, un format d'échange de données.

Le protocole définit comment s'établit la communication *client-serveur*.
Il fournit à l'utilisateur des commandes pour se **connecter** ou se **déconnecter**, pour **rechercher**, **comparer**, **créer**, **modifier** ou **effacer** des entrées. Des mécanismes de chiffrement (SSL ou TLS) et d'authentification (SASL), couplés à des mécanismes de règles d'accès (ACL) permettent de protéger les transactions et l'accès aux données. La plupart des logiciels serveurs LDAP proposent également un protocole de communication *serveur-serveur* permettant à plusieurs serveurs d'échanger leur contenu et de le synchroniser (*replication service*) ou de créer entre eux des liens permettant ainsi de relier des annuaires les uns aux autres (*referral service*). La communication *client-serveur* est normalisée par l'IETF.

Contrairement à d'autres protocoles d'Internet, comme HTTP, SMTP ou NNTP, le *dialogue* LDAP ne se fait pas en ASCII mais utilise le format de codage *Basic Encoding Rule* (BER).

3.8. Organisation des données (modèle de nommage)[16][17]

Le modèle de nommage est la manière dont sont organisées les données dans l'annuaire. LDAP organise les données de manière hiérarchique dans l'annuaire. Ceci signifie que toutes les informations découlent d'une seule et même "racine". Voici un exemple d'arborescence LDAP (figure 3.1):

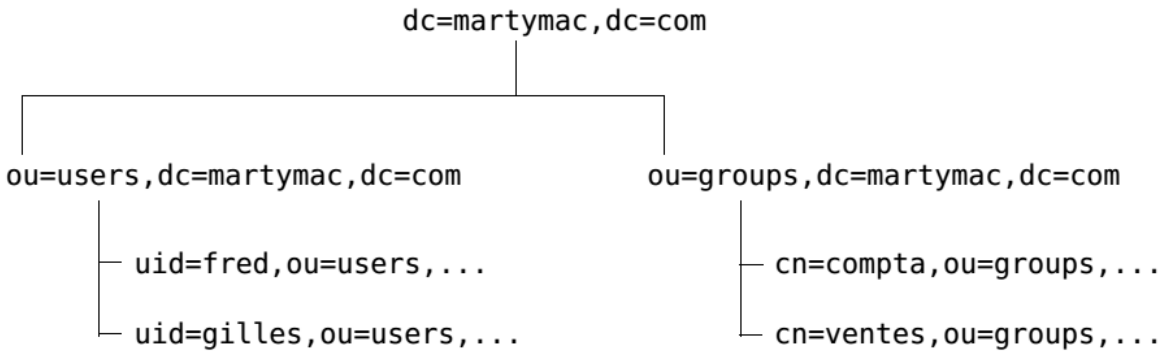


Figure 3.1 : Exemple d'arborescence LDAP[16][17]

Cette arborescence est liée au nommage de chaque élément : un élément marque son appartenance à l'élément supérieur en en reprenant le nom, qu'il complète par le sien. Ainsi, en étudiant simplement le nom de l'élément: *"cn=ventes,ou=groups,dc=martymac,dc=com"* il est possible de le situer dans la hiérarchie : il est situé sous l'élément "ou=groups" qui lui même est situé sous l'élément "*dc=martymac,dc=com*"

Termes à connaître : Voici quelques termes qui faut connaitre:

- Chaque élément est appelé une entrée (an entry). Une entrée peut être un branchement (un noeud, a node) ou un élément terminal (une feuille, a leaf).

- Chaque élément possède un DN (Distinguished Name). Le DN est le nom complet de l'élément qui permet de le positionner dans l'arborescence. Il est unique dans l'annuaire. Exemple :"cn=ventes,ou=groups,dc=martymac,dc=com".

- Chaque élément possède également un RDN (Relative Distinguished Name). Le RDN est la partie du DN de l'élément qui est relative au DN supérieur. Le RDN d'un élément ne permet pas de l'identifier de manière absolue dans l'annuaire. Exemple :"cn=ventes".

- La racine est l'élément supérieur de tous les autres, c'est la base de l'arborescence. On l'appelle root en anglais, parfois on parle de "root DN". Exemple:"dc=martymac,dc=com".

- Les DN de chaque entrée sont composés au moins d'un attribut de l'élément (par exemple "cn" ou "uid") et de sa valeur. Un attribut est l'une des caractéristiques de cet élément. Remarquez que la racine choisie ici est composée du nom du domaine où est hébergé notre serveur LDAP, martymac.com, décomposé en "dc" (Domain Components) pour obtenir dc=martymac,dc=com.

- L'arbre se découpe ensuite en deux "ou" (Organisational Units) qui constituent deux branchements : "users" et "groups", dans lesquels nous trouvons ensuite les entrées feuilles de notre arbre : les utilisateurs et les groupes.

- Chacune des entrées de notre arbre correspond à un type de donnée particulier, défini par une classe d'objet.

3.9. Le modèle fonctionnel[16][17]

Il existe plusieurs types d'opérations que l'on peut effectuer sur l'annuaire, voici les plus importantes
- Rechercher une entrée suivant certains critères.
- S'authentifier.
- Ajouter une entrée.
- Supprimer une entrée.
- Modifier une entrée.
- Renommer une entrée

Certaines de ces actions, notamment la recherche, nécessitent des outils particuliers pour nous faciliter l'accès à l'annuaire

a. La base

La base est le DN à partir duquel nous allons agir. Pour une recherche, il s'agit du nœud à partir duquel est effectuée la recherche. Il peut s'agir de la racine de l'arbre pour une recherche sur la totalité de l'arbre, par exemple "dc=martymac,dc=com".

b. La portée

La portée (scope) est le nombre de niveaux sur lesquels l'action va être effectuée. Il existe 3 niveaux différents :

• **SUB** : l'action est effectuée récursivement à partir de la base spécifiée sur la totalité de l'arborescence.
• **ONE** : l'action est effectuée sur un seul niveau inférieur par rapport à la base spécifiée (les fils directs). Si l'on effectuait une recherche avec la portée ONE à partir de "dc=martymac,dc=com", nous pourrions trouver "ou=users,dc=martymac,dc=com" et "ou=groups,dc=martymac,dc=com".
• **BASE** : l'action est effectuée uniquement sur la base spécifiée. Une recherche sur "dc=martymac,dc=com" avec la portée BASE renverrait cette entrée uniquement.

c. Les filtres

Le troisième outil à notre disposition est le filtre. Un filtre va permettre d'effectuer des tests de correspondance lors d'une recherche. Il s'agit en quelques sortes du critère de la recherche. Il existe 4 tests basiques, qui peuvent ensuite être combinés :

- Le test d'égalité : X=Y.
- Le test d'infériorité : X<=Y.
- Le test de supériorité : X>=Y.
- Le test d'approximation : X~=Y.

Les autres opérateurs (<, >) ou des tests plus complexes peuvent être mis en place par combinaison, il faut alors utiliser les parenthèses ( ) et l'un des opérateurs suivants :

- L'intersection (et) : &.
- L'union (ou) : |.
- La négation (non) : !.
- Un test d'inférioté stricte pourrait donner ceci : (&(X<=Y)(!(X=Y)))

On peut combiner plus de deux éléments : (&(X=Y)(Y=Z)(A=B)(B=C)(!(C=D)))

Ces filtres seront appliqués sur des attributs choisis pour sélectionner finement les données que nous voulons extraire de notre annuaire

d. Les URLs LDAP

Les URL est une méthode concise et simplifiée pour interroger un annuaire LDAP. Il s'agit d'un format d'URL combinant toutes les notions que nous avons étudiées. En une

seule ligne, il est possible de spécifier tous les éléments de notre requête. Voici le format de cette URL (RFC 2255, rendue obsolète par la RFC 4516) : ldap[s]://serveur[:port][/[base[?[attributs à afficher][?[portée][?[filtre][?extensions]]]]]]. L'exemple ci-dessous recherche tous les uid de notre arbre, à partir de la branche users :

ldap://localhost:389/ou=users,dc=martymac,dc=com?uid?sub

3.10. Modèle d'information (Les données contenues dans l'annuaire) [16][17]

a. Les attributs

Un attribut est une valeur contenue dans une entrée. Une entrée peut bien entendu contenir plusieurs attributs. Prenons l'exemple de l'entrée LDAP complète d'un compte utilisateur POSIX (Figure 3.2):

```
dn: uid=martymac,ou=users,dc=martymac,dc=com
objectClass: account
objectClass: posixAccount
cn: martymac
uid: martymac
uidNumber: 10001
gidNumber: 10001
homeDirectory: /home/martymac
userPassword:: e0NSWVBUfWJjT29IUk5SbG1HbC4=
loginShell: /bin/sh
gecos: martymac
description: martymac
```

Figure 3.2 : un exemple d'une entrée LDAP[16][17]

Ceci correspond à une entrée complète, extraite par une interrogation de l'annuaire. Le format

affiché est le format LDIF. Ce paragraphe présente tous les attributs, un par ligne, que comprend notre entrée. Un attribut est séparé de sa valeur par ":" suivant son type, un attribut peut avoir plusieurs valeurs : dans ce cas, il est dit "multi-valué" et apparaît sur plusieurs lignes avec des valeurs différentes.

Nous pouvons observer ici des attributs nommés "dn", "objectClass", "cn", "uid".

L'attribut "dn" qui est indiqué en première ligne est le nom unique de notre entrée dans l'arbre dont nous avons parlé précédemment. Il constitue un attribut à part entière dans notre entrée. Il est composé du dn de l'entrée supérieure, ainsi que du rdn.

Sur un annuaire LDAP la racine est toujours composée des attributs "dc" (Domain Component) associés à chacune des parties du nom de domaine où est hébergé le serveur ("dc=martymac,dc=com" pour le domaine martymac.com).

L'attribut "ou" constitue une "Organisational Unit", c'est à dire une unité organisationnelle : en quelque sorte un regroupement. Dans l'exemple y'on a deux:

"users", qui accueillera les utilisateurs et "groups", les groupes.

a. Les classes d'objets

A première vue, l'entrée présentée ci-dessus constitue un amalgame de différentes informations. Toutes ces entrées sont induites par la présence des objectClass. L'objectClass d'une entrée est un attribut qui permet de cataloguer cette entrée. Un objectClass définit un regroupement d'attributs obligatoires ou autorisés pour une entrée.

Une entrée peut posséder un ou plusieurs objectClass. Ce sont ces objectClass qui définissent

la présence de tous les autres attributs. Ici, l'objectClass "posixAccount" rend obligatoire les attributs cn, uid, uidNumber, gidNumber et homeDirectory. Il rend possible l'utilisation des 4 autres attributs userPassword, loginShell, gecos et description.

b. Les schémas

Les schémas décrivent les objectClass existant ainsi que leurs syntaxe et les attributs qu'ils contiennent.

Un annuaire LDAP a la capacité de charger en mémoire plusieurs schémas. A travers ces schémas, il est possible de définir de nouveaux attributs et de nouveaux objectClass. Cette souplesse permet de définir très finement ce qui sera stocké dans notre annuaire.

Concrètement, un schéma est un fichier qui décrit un à un les attributs disponibles (leur nom, leur type, etc...), ainsi que les objectClass qui y font appel. Au démarrage du serveur LDAP, le ou les fichiers de schéma spécifiés dans sa configuration seront chargés.

Dans l'exemple précédent, l'objectClass « posixAccount » est défini dans le fichier nis.schema. livré avec OpenLDAP et situé dans /etc/ldap/schema comme montré dans la figure 3.3:

```
# [...]
attributetype ( 1.3.6.1.1.1.1.0 NAME 'uidNumber'
        DESC 'An integer uniquely identifying a user in a domain'
        EQUALITY integerMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE )

# [...]
objectclass ( 1.3.6.1.1.1.2.0 NAME 'posixAccount' SUP top AUXILIARY
        DESC 'Abstraction of an account with POSIX attributes'
        MUST ( cn $ uid $ uidNumber $ gidNumber $ homeDirectory )
        MAY ( userPassword $ loginShell $ gecos $ description ) )
# [...]
```

Figure 3.3 : Une partie du fichier nis.schema décrivant l'objectClasse posixAccount. [16][17]

Le premier paragraphe définit l'un des attributs utilisés par le posixAccount : uidNumber. Le second, l'objectClass posixAccount. Il faut savoir que :

- A chaque définition correspond un OID (Object IDentifier), qui permet de rendre unique l'attribut spécifié. Ces OIDs sont déposés auprès de l'IANA (http://www.iana.org) et sont donc officiels.
- Un attribut définit un type d'égalité à mettre en œuvre lors d'une recherche (ici, intergerMatch) ainsi que le type de données qu'il contient (l'OID spécifié après SYNTAX).
- Un objectClass définit les attributs que l'objet doit présenter (MUST) et ceux qu'il peut posséder (MAY).
- Les schémas constituent donc une source d'information très importante. En cas de doute concernant le type ou le nom des attributs à spécifier dans une entrée, n'hésitez pas à vous y reporter.
- Enfin, sachez qu'il est tout à fait possible de créer ses propres schémas, cependant, penser à réutiliser les schémas existants : ils offrent déjà de nombreuses possibilités et il y a fort à parier qu'un schéma existe déjà pour gérer les informations que vous souhaitez.

c. Le format LDIF

Les données contenues dans l'annuaire sont présentées dans un certain format : il s'agit du format LDIF (LDAP Data Interchange Format - RFC 2849). Comme montré dans la figure 3.3. Toute interaction avec un annuaire se fait par le biais de ce format : l'ajout, la modification, la suppression d'entrées, l'interrogation de l'annuaire y compris. Dans ce format, chaque entrée constitue un paragraphe, et, au sein de chaque paragraphe, chaque ligne constitue un attribut. Voici un exemple un peu plus complet, incluant le groupe des utilisateurs (figure 3.4) :

```
# [...]
dn: cn=utilisateurs,ou=groups,dc=martymac,dc=com
objectClass: posixGroup
cn: utilisateurs
gidNumber: 10001

dn: uid=martymac,ou=users,dc=martymac,dc=com
objectClass: account
objectClass: posixAccount
cn: martymac
uid: martymac
uidNumber: 10001
gidNumber: 10001
homeDirectory: /home/martymac
userPassword:: e0NSWVBUfWJjT29IUk5SbG1HbC4=
loginShell: /bin/sh
gecos: martymac
description: martymac
# [...]
```

Figure 3.4 : le format LDIF[16][17]

3.11. La sécurité (modèle de sécurité) [16][17]

Lorsque vous mettez en place un annuaire d'entreprise, il convient de réfléchir au modèle de sécurité que vous souhaitez appliquer. LDAP fournit plusieurs mécanismes permettant de mener à bien votre projet.

a. L'authentification simple, le binding

L'annuaire met en place un mécanisme d'authentification. L'une des opérations préalables à l'interrogation de l'annuaire est cette opération dite de "binding" (dans le cas d'une authentification simple). Le client envoie alors le DN d'un compte contenu dans l'annuaire lui-même, ainsi que le mot de passe associé. On pourra par la suite appliquer des droits particuliers sur ce compte en utilisant les ACLs.

Ceci correspond, si l'on fait le parallèle avec l'annuaire téléphonique, à la fonctionnalité de liste rouge, où certaines données ne sont pas accessibles à tout le monde. Il est possible de se connecter de manière anonyme : le client envoie alors un DN vide au serveur LDAP.

b. Les ACLs

Les ACLs (Access Control Lists) interviennent après la notion de binding. Il sera possible de donner des droits de lecture, d'écriture (ou d'autres droits divers) sur des branches particulières

de l'annuaire au compte connecté. Ceci permet de gérer finement les droits d'accès aux données.

c. Le chiffrement des communications (SSL/TLS)

Le chiffrement des communications, via SSL (Secure Socket Layer, ou TLS - Transport Layer Security) est également une méthode de protection de l'information. Il est possible, avec la plupart des annuaires existants, de chiffrer le canal de communication entre l'application cliente et l'annuaire. Ceci permet de garantir, un minimum, de confidentialité des données et d'éviter qu'un tiers n'écoute les communications sur le réseau.

d. SASL

SASL (Simple Authentication and Security Layer) est un mécanisme qui permet d'ajouter des méthodes d'authentification à des protocoles orientés connexion tels que LDAP ou IMAP.

Il est défini dans la RFC 2222 ; l'implémentation la plus couramment utilisée est Cyrus-Sasl (http://asg.web.cmu.edu/sasl). SASL donne la possibilité au client et au serveur de sélectionner quelle sera la méthode d'authentification utilisée. Ces méthodes sont extensibles via des plugins. Il permet également de mettre en place une couche de connexion sécurisée telle que SSL/TLS.

3.12. Le modèle de réplication (duplication) [16][17]

Certains serveurs LDAP, dont OpenLDAP, permettent de manière native, de mettre en place un annuaire répliqué. Un annuaire dit "maître" envoie alors, par le biais du format LDIF, toutes les modifications effectuées sur un annuaire "esclave".

L'avantage d'une telle opération est double :

- permettre une meilleure montée en charge pour de gros annuaires : il est possible de rediriger le client vers l'un ou l'autre des annuaires répliqués.
- disposer d'une copie conforme du premier annuaire, utile en cas de crash (attention, toute opération est reportée de l'annuaire maître vers l'esclave, donc ceci est non valable en cas de mauvaise manipulation).

Deux types de réplication existent :

- le mode "maître-esclave", le plus courant : la réplication est unidirectionnelle, un annuaire maître envoie toutes les modifications à un annuaire esclave. Ceci n'autorise bien évidemment l'écriture que sur l'annuaire maître ; l'esclave est alors disponible uniquement. en lecture.
- le mode "maître-maître" : la réplication est bidirectionnelle, chaque annuaire peut être maître de l'autre. Ceci permet d'écrire indifféremment sur l'un ou l'autre des annuaires. Il est possible de chaîner les réplications pour obtenir plusieurs réplicats.

a. La distribution (les referrals)

La distribution est un mécanisme qui va permettre de faire pointer un lien vers un autre annuaire pour une branche particulière. Ceci va permettre de déléguer la gestion de cette branche, un peu au sens DNS lorsqu'on délègue la gestion d'un domaine. Ce mécanisme peut être représenté de la manière suivante, si l'on reprend l'exemple de notre domaine martymac.com (figure 3.5):

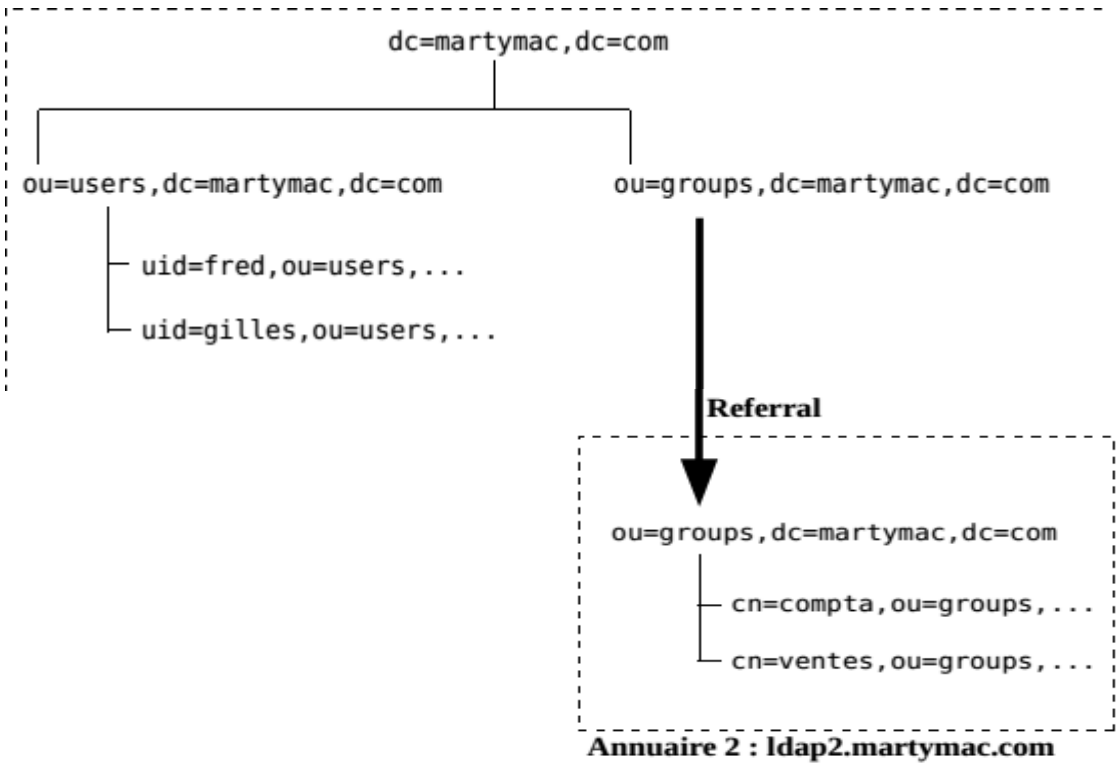


Figure 3.5 : Exemple d'utilisation de Referral[16][17]

Dans la figure, l'annuaire 1 possède un referral pour la branche ou=groups. Ce referral pointe vers l'annuaire 2. La gestion de cette branche est donc en quelques sortes "déléguée" à l'annuaire 2. Au niveau de l'annuaire 1, ceci se traduit par une entrée de la classe "referral", qui contient alors un attribut "ref" contenant l'adresse de la suite de l'arborescence comme montré dans la figure 3.6.

```
dn: ou=groups,dc=martymac,dc=com
objectClass: referral
ref: ldap://ldap2.martymac.com/ou=groups,dc=martymac,dc=com
```

Figure 3.6 : un exemple de referral avec le format LDIF[16][17]

TP1

TP sur LDAP et la sécurité

(Rapport sur le TP + validation de 10 mn)

Travail demandé

Le TP consiste à mettre en œuvre un annuaire LDAP avec gestion des droits d'accès. Il comporte deux parties :

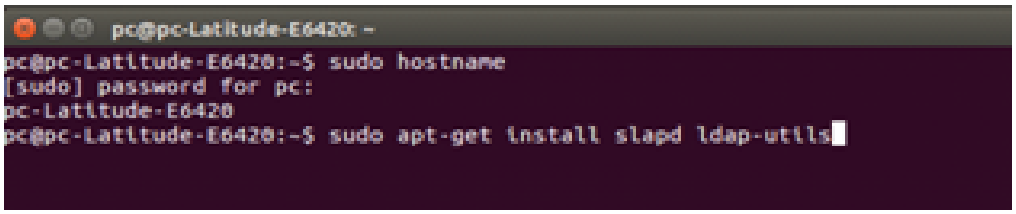
1. la gestion des listes ACL.
2. la mise en œuvre de certificats.

Corrigé TP1

LDAP et la sécurité

1 installations et configuration nécessaires

- **Installation des paquets LDAP à l'aide la commande suivante :**



- Installation du phpldapadmin

- sudo apt-get install phpldapadmin

- Modifier le fichier (/etc/ldap/ldap.conf) de tel façon qu'il soit comme figure 1

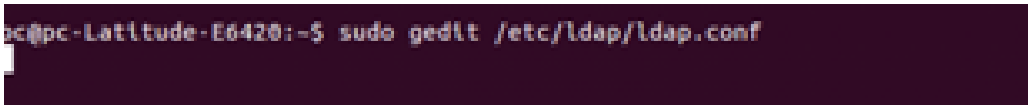


Figure 1

- **<u>Reconfigurer le LDAP de la façon suivante :</u>**

- Sudo dpkg-reconfigure slapd

Figure 2

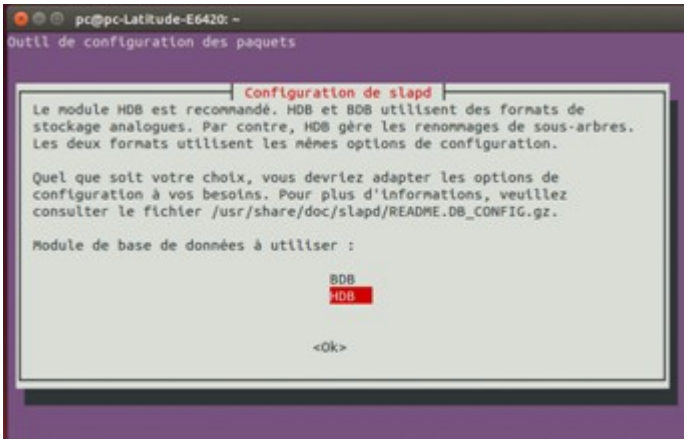


Figure 3

Figure 4

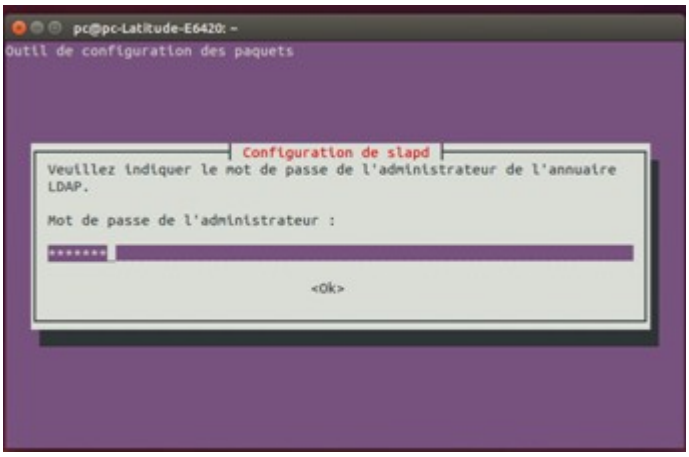


Figure 5

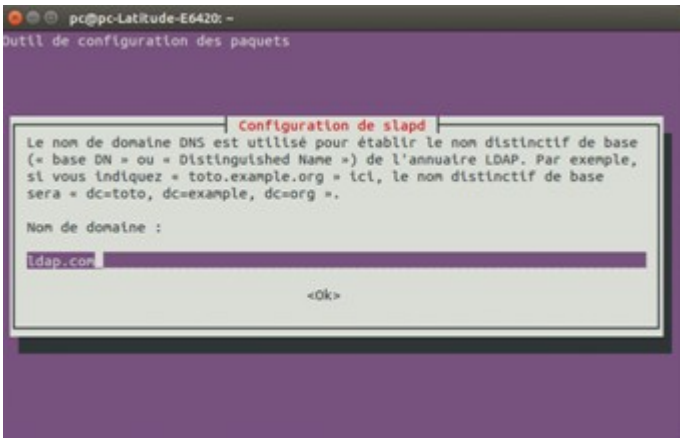


Figure 6

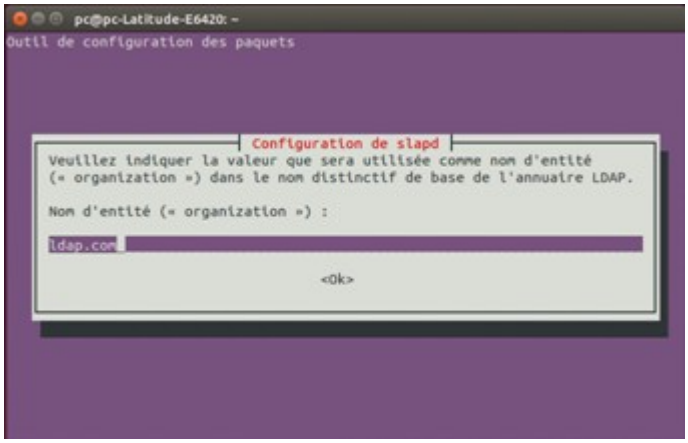


Figure 7

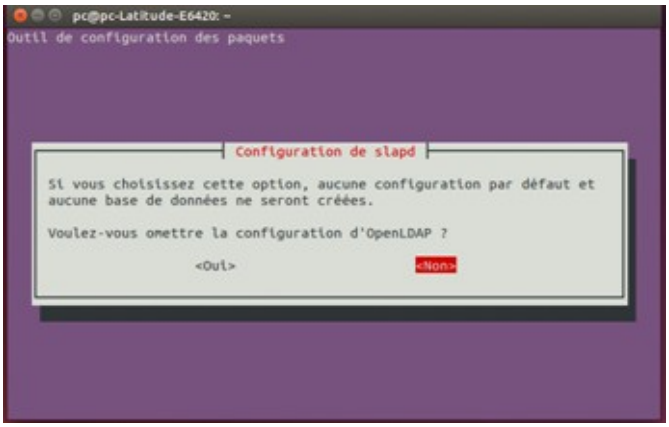


Figure 8

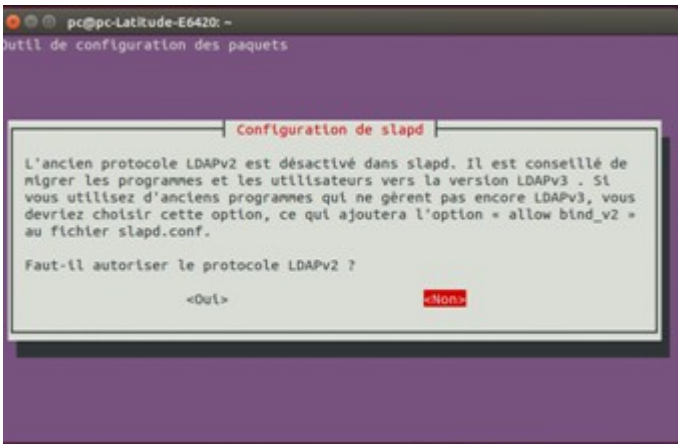


Figure 9

- **<u>Tester la configuration</u>**

Pour tester la configuration il suffit juste de taper la commande : **ldapsearch –x**

- **<u>Modification dans le fichier config.php</u>**

Accéder au fichier /etc/phpldapadmin/config.php et effectuer les modifications suivantes :

- $Servers->values('login ','auth_type','session') ;
- $Servers->values('login','blind_id','cn=admin,dc=ldap,dc=com') ;
- $Servers->values('server','host ','127.0.0.1') ;
- $Servers->values('server','base',array('dc=ldap,dc=com') ;

- **<u>Redémarrer le service SLAPD</u>**

Sudo service slapd restart

- **<u>Accéder à la plateforme Openldap</u>**

En tapant (127.0.0.1/phpldapadmin)

Figure 10

On peut entrer les informations à la base par cette commande (par format ldif) :

```
ldapadd -x -D cn=admin,dc=ldap,dc=com -W -f g.ldif
```

Voila le résultat :

```
adding new entry "cn=informatique,cn=admin,dc=ldap,dc=com"

adding new entry "ou=RSI,cn=informatique,cn=admin,dc=ldap,dc=com"

adding new entry "ou=chef_specialite,ou=RSI,cn=informatique,cn=admin,dc=ldap,dc=com"

adding new entry "ou=Enseignant,ou=RSI,cn=informatique,cn=admin,dc=ldap,dc=com"

adding new entry "uid=nafaa,ou=chef_specialite,ou=RSI,cn=informatique,cn=admin,dc=ldap,dc=com"

adding new entry "uid=ghoualmi,ou=chef_specialite,ou=RSI,cn=informatique,cn=admin,dc=ldap,dc=com"

adding new entry "uid=bendjeddou,ou=Enseignant,ou=RSI,cn=informatique,cn=admin,dc=ldap,dc=com"
```

Figure 11

2- ACL's

- **Création des fichiers LDIF**

Sudo touch acl.ldif

Sudo gedit acl.ldif

Ajouter dans le fichier acl.ldif : le contenu suivant

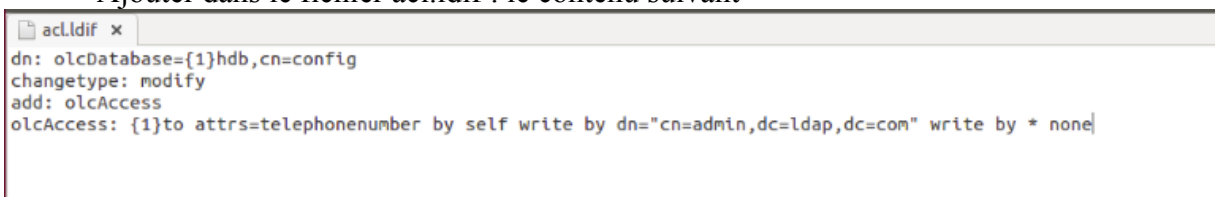


Figure 12

- **Affichage du contenu de la base**

sudo ldapsearch –Q –LLL –Y EXTERNAL –H ldapi :/// -b cn=config'(olcDatabase={1} hdb)'olcAcces

- **Inclure le fichier LDIF dans la base**

sudo ldapmodify -Y EXTERNAL -H ldapi:/// -f acl.ldif

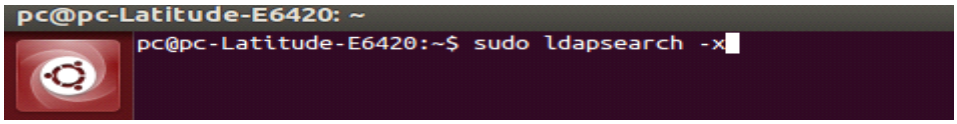
- **Affichage dans la base un utilisateur**

Pour afficher seulement le mot passe (chiffrer) pour un utilisateur prenant par ex : ghoualmi et pour les autres seront invisibles

ldapsearch -D uid="ghoualmi,ou=chef_specialite,ou=RSI,cn=informatique,cn=admin,dc=ldap,dc=com" -w naciraldap -b dc=ldap,dc=com objectclass=*

- **Affichage dans la base un utilisateur(mode anonyme)**

Pour afficher tout les informations de la base mais les attributs (password et telephone number)sont invisibles



Voila le résultat :

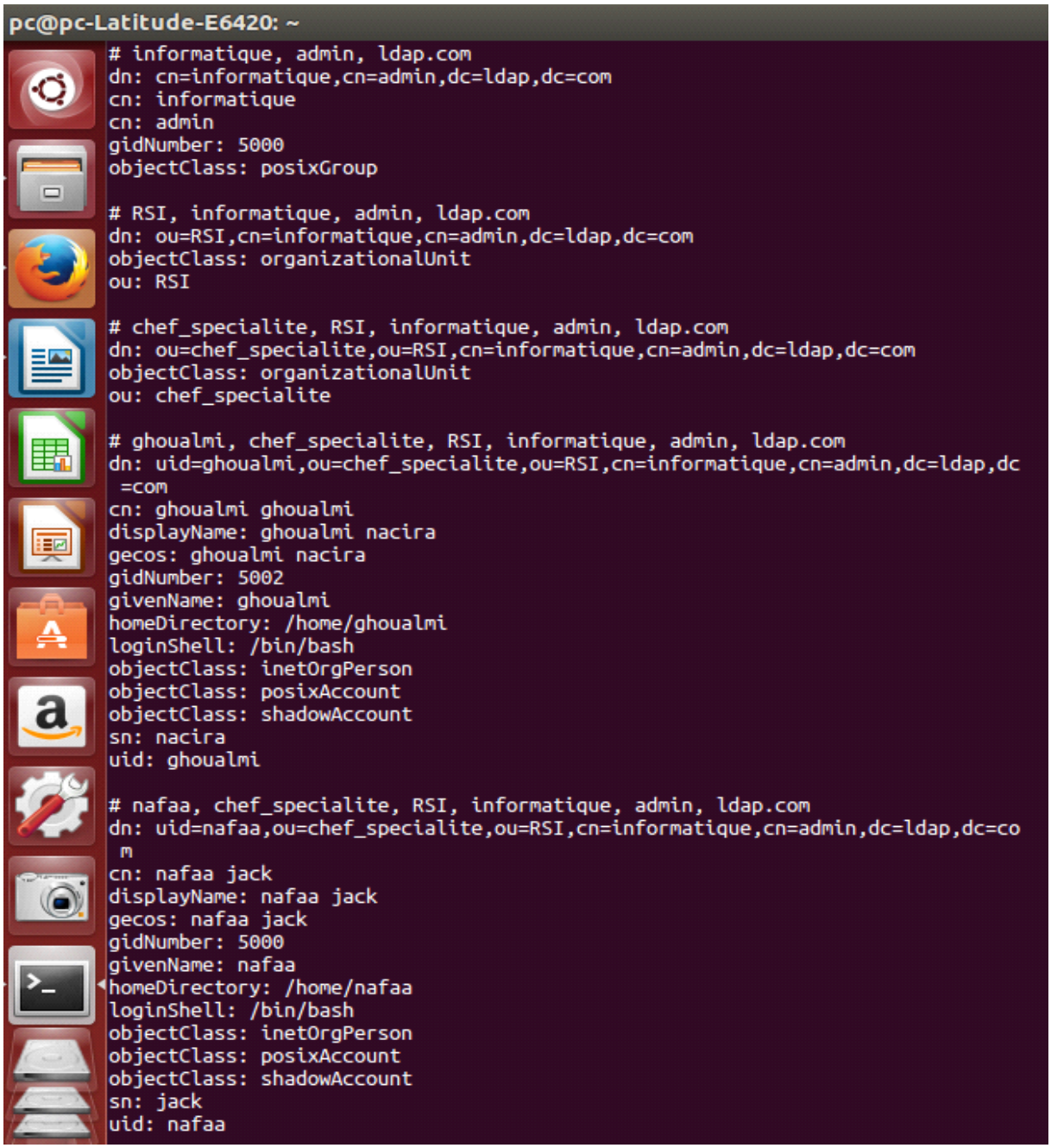


Figure 13

3- SSL/TLS

- **Installer les packages nécessaires :**

Sudo apt-get update
Sudo apt-get install gnutls-bin

- **Création des certificats**

sudo mkdir –p /root/certs/

certtool --generate-privkey –outfile /etc/ssl/private/ldap-ca-key.pem

certtool --generate-self-signed --load-privkey /etc/ssl/private/ldap-ca-key.pem --outfile /root/certs/ldap-ca-cert.pem

sudo adduser openldap ssl-cert
sudo chgrp ssl-cert /etc/ssl/private/ldap-ca-key.pem

certtool –generate-privkey –outfile /root/certs/ldap-server.key

- **Création de la clé du serveur et le certificat**

On aura une sorti d'un formulaire à remplir contenant des informations comme suit :

```
Common name: ca.edu.example.org
The certificate will expire in (days): 3650
Does the certificate belong to an authority? (y/N): y
Path length constraint (decimal, -1 for no constraint): -1
Will the certificate be used to sign other certificates? (y/N): y
```

Figure 14

- **Installation des certificats**

```
install -D -o openldap -g openldap -m 600 /root/certs/ldap-server.crt \
           /etc/ssl/certs/ldap-server.crt
install -D -o openldap -g openldap -m 600 /root/certs/ldap-server.key \
           /etc/ssl/certs/ldap-server.key
install -D -o openldap -g openldap -m 600 /root/certs/ldap-ca-cert.pem \
           /etc/ssl/certs/ldap-ca-cert.pem
```

Figure 15

- **Création du fichier LDIF**

```
~$ sudo gedit /etc/ldap/ssl.ldif
```

ssl.ldif

```
dn: cn=config
add: olcTLSCACertificateFile
olcTLSCACertificateFile: /etc/ssl/certs/ldap-ca-cert.pem
-
add: olcTLSCertificateFile
olcTLSCertificateFile: /etc/ssl/certs/ldap-server.crt
-
add: olcTLSCertificateKeyFile
olcTLSCertificateKeyFile: /etc/ssl/certs/ldap-server.key
```

Figure 16

- **Importer le fichier**

On doit avoir ce résultat

```
ldap_initialize( ldapi:///??base )
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
add olcTLSCACertificateFile:
        /etc/ssl/certs/ldap-ca-cert.pem
add olcTLSCertificateFile:
        /etc/ssl/certs/ldap-server.crt
add olcTLSCertificateKeyFile:
        /etc/ssl/certs/ldap-server.key
modifying entry "cn=config"
modify complete
```

Figure 17

- **Ecouter le port de SSL**

```
$ sudo gedit /etc/default/slapd
```

- **Trouver la ligne**

SLAPD_SERVICES="ldap:/// ldapi:///"

- **Rajouter ces informations**

```
SLAPD_SERVICES="ldap:/// ldapi:/// ldaps:///"

# If SLAPD_NO_START is set, the init script will not start or restart
# slapd (but stop will still work).  Uncomment this if you are
# starting slapd via some other means or if you don't want slapd normally
```

Figure 18

- **Redémarrer LDAP**

- **Configuration côté client**

```
-$ sudo ldapsearch -x -h 127.0.0.1 -ZZ -b dc=ldap,dc=com
```

On aura comme résultat

```
# bendjeddou, Enseignant, RSI, informatique, admin, ldap.com
dn: uid=bendjeddou,ou=Enseignant,ou=RSI,cn=informatique,cn=admin,dc=ldap,dc=co
 m
cn: ghoualmi bendjeddou
displayName: bendjeddou amira
gecos: bendjeddou amira
gidNumber: 5002
givenName: bendjeddou
homeDirectory: /home/bendjeddou
loginShell: /bin/bash
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
sn: amira
uid: bendjeddou
uidNumber: 10001

# search result
search: 3
result: 0 Success

# numResponses: 11
# numEntries: 10
```

Figure 19

- **Ajout des certificats**

Génération de la clé :

```
~$ sudo certtool --generate-privkey --outfile /home/mohamed/cert/nafaa.key
```

Génération du certificat :

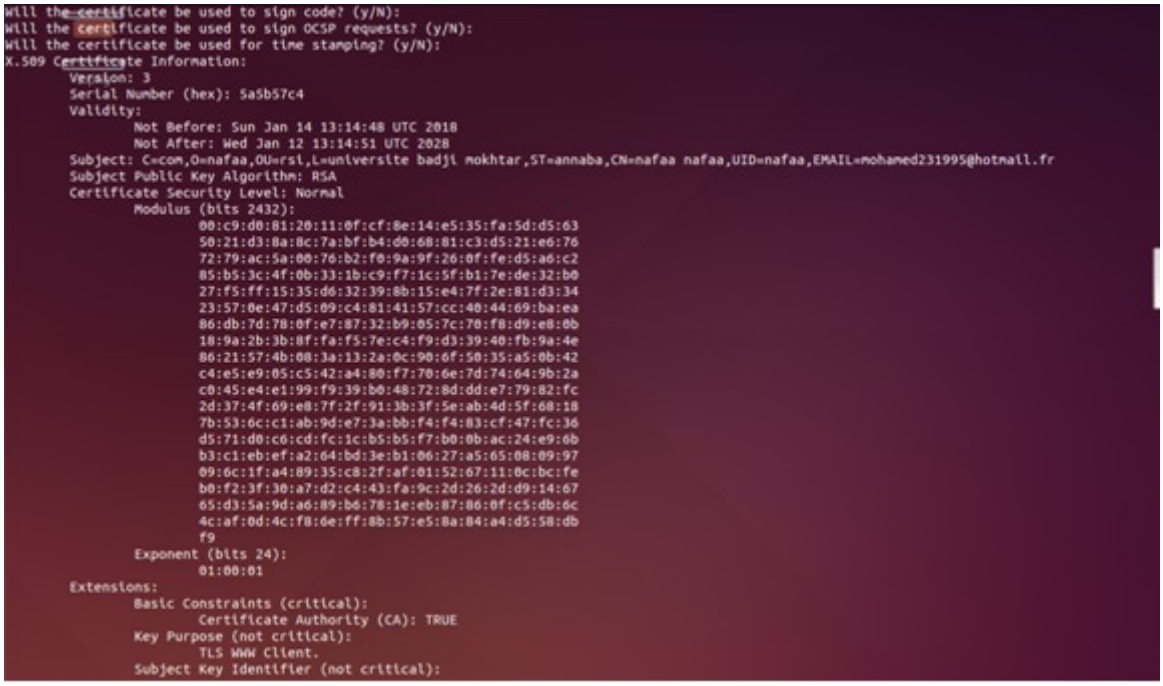


Figure 20

Et le même principe pour les autres utilisateurs en changeant juste (UID, CN, …..).

CHAPITRE 4

LE PROTOCOLE SSL/TLS

Chapitre 4

Le protocole SSL/TLS

4.1. Introduction

SSL (pour l'anglais Secure Socket Layer) fut conçu par Netscape pour sécuriser les transactions commerciales sur son navigateur Netscape Navigator 1.1, alors le leader du marché en 1994, en remplaçant http par https (avec s pour secure). La première version, connue sous le nom de SSLv1, de 1994 n'a pas été diffusée. La seconde version, SSLv2, a été diffusée en novembre 1994 et une implémentation a été intégrée dans le navigateur en mars 1995.
Wagner et Goldberg ont trouvé une faille de sécurité, due au générateur de nombres pseudo-aléatoires utilisé, qui permettait de casser une connexion SSL en moins d'une heure. Netscape recruta un consultant en sécurité bien connu, Paul Kocker, pour travailler avec Allan Freier et Phil Karlton afin de développer une nouvelle version de SSL. Cette troisième version, SSLv3, est sortie fin 1995.

Microsoft a conçu également un protocole analogue, PCT pour Private Communication Technology, également en 1995 .En mai 1996, l'Internet Engineering Task Force (IETF) a mis en place un groupe de travail Transport Layer Security (TLS) pour standardiser les protocoles du type SSL. TLS a été publié en janvier 1999, avec peu de différences par rapport à SSLv3.

La sécurisation des connexions à l'aide du protocole SSL doit assurer que :

• La connexion assure la confidentialité des données transmises

• La connexion assure que les données transmises sont intègres

• L'identité des correspondants peut être authentifiée

• La connexion est fiable

4.2. Place du protocole dans la suite TCP/IP[18][19]

SSL/TLS est un protocole qui prend place entre le protocole de la couche de transport (dans les faits uniquement TCP) et la couche application, Pour mettre en place une connexion SSL, il faut d'abord établir une connexion TCP/IP, car SSL utilise certaines "primitives" de TCP/IP. Ainsi SSL peut être vu comme un canal sûr au sein de TCP/IP, où tout le trafic entre deux applications "peer to peer" est échangé de manière cryptée.

Tous les appels de la couche d'application à la couche TCP, sont remplacés par des appels de l'application à SSL, et c'est SSL qui se charge des communications avec TCP comme le montre la figure 4.1.

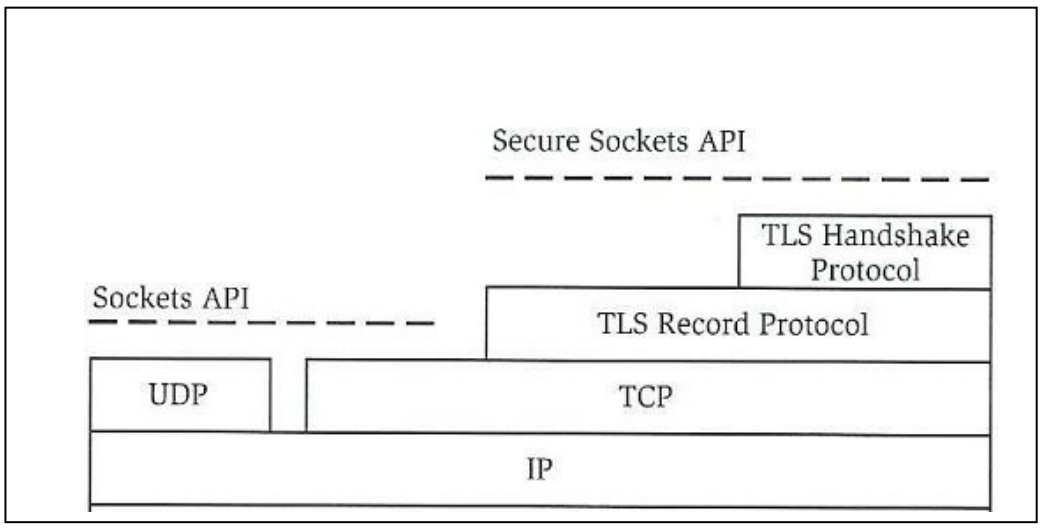


Figure 4.1 : Place du protocole SSL dans la suite TCP/IP[18][19]

4.3. Fonctionnalités [18][19][20]

SSL a trois fonctions:

- Authentification du serveur : Qui permet à un utilisateur d'avoir une confirmation de l'identité du serveur. Cela est fait par les méthodes de chiffrement à clés publiques qu'utilise SSL. Cette opération est importante, car le client doit pouvoir être certain de l'identité de son interlocuteur à qui par exemple, il va communiquer son numéro de carte de crédit.
- Authentification du client : Selon les mêmes modalités que pour le serveur, il s'agit de s'assurer que le client est bien celui qu'il prétend être.
- Chiffrement des données : Toutes les données qui transitent entre l'émetteur et le destinataire, sont chiffrées par l'émetteur, et déchiffrées par le destinataire, ce qui permet de garantir la confidentialité des données, ainsi que leur intégrité grâce souvent à des mécanismes également mis en place dans ce sens.

4.4. Principes du SSL/TLS [21][20]

SSL est un protocole qui utilise différents algorithmes de cryptographie afin de garantir "la sécurité", au travers de l'authentification avec des certificats, des sessions d'échanges des clés d'algorithmes, de l'encryptage, et de la vérification de l'intégrité des données. Le cryptage SSL, fonctionne par le choix aléatoire de deux nombres premiers, qui multipliés entre eux forment un très grand nombre. Ce dernier constitue la clé de cryptage. Sans la connaissance des deux nombres premiers ayant servis à générer cette clé, il n'est pas possible de pouvoir décrypter un message. En réalité une manière possible serait de défactoriser le nombre afin de retrouver les deux nombres premiers, mais les nombres sont tellement grands, que cela n'est pas à la portée d'ordinateurs conventionnels.

Il est à relever que déjà à partir de sa version 2.0, SSL a commencé à être utilisé. Raison pour laquelle encore aujourd'hui, la plupart des intervenants SSL, tentent lors de la phase d'établissement, de dialoguer avec le protocole v3.0, mais si l'un des deux partenaires ne supporte que la version 2.0, et bien c'est cette version antérieur qui va être utilisée. A noter que cette ancienne version contient des clés de cryptage considérées aujourd'hui comme peu sûres. (Par exemple au niveau de la longueur de la clé, il y a un passage de 40 à 128 bits et plus, pour la version v3.0).

4.5. Composition du SSL [21][20][22]

SSL se subdivise en quatre sous protocoles; *le SSL record Protocol*, et *le SSL handshake protocol*. Plus deux autres protocoles, mais qui ont un rôle moins essentiel, c'est *le SSL Change Ciffer Spec*, et *le SSL Alert*.

Le SSL record protocol définit le format qui sera utilisé pour l'échange des données. Alors que le SSL handshake se charge des différents échanges de messages entre le client et le serveur, au moment où ils établissent la connexion comme l'authentification, la version de protocole, l'algorithme de cryptage, etc. Une session SSL est définie par les paramètres suivants partagés entre un client et un serveur :

- Session identifier : un octet fixé par le serveur pour identifier la session
- Peer certificat : un certificat pour le serveur, éventuellement un autre pour le client
- Cipher Spec : définit l'algorithme de chiffrement symétrique et l'algorithme de condensation
- Master secret : clé de 48 octets négociée entre le serveur et le client
- Compression method : NULL pour l'instant (en SSLv3 ou TLS).
- Is resumable : flag qui indique si de nouvelles connexions peuvent être créées à partir de cette session.

4.5.1. Le protocole SSL handshake

Ce protocole permet au client et au serveur de s'authentifier mutuellement, de négocier les algorithmes de chiffrement, de négocier les algorithmes de MAC et enfin de négocier les clés symétriques qui vont servir au chiffrement. Chaque message échangé entre le client et le serveur contient trois champs :

- Type, indique l'objet du message (1 octet)
- Lenght, indique la longueur du message (3 octets)
- Content, contient les données transmises (plus d'un octet)

Ce protocole contient 4 phases :

Phase 1 : Etablissement des paramètres de sécurité

Cette phase à pour but d'établir le lien sécurisé. Le client envoie un message du type « **Client Hello** ». Ceci indique que le client désire établir une session de sécurité sur sa connexion TCP. Il envoie un nombre aléatoire (qui sera utilisé pour la

détermination des clés), un identificateur de session et la liste des algorithmes de sécurité et de compression qu'il est capable d'utiliser.

Les paramètres du premier message, « client hello », envoyé par le client, sont :

- Version : la plus haute version de SSL que puisse utiliser le client.
- Random : un horodatage de 32 bits et une valeur aléatoire de 28 octets générée par le client. Ces valeurs servent de sel et sont utilisées pour se prémunir contre les rejets de paquets.
- Session ID : Un nombre, qui identifie la connexion. Un zéro signifie la volonté du client d'établir une nouvelle connexion sur une nouvelle session. Un autre nombre signifie la volonté de changer les paramètres ou de créer une nouvelle connexion sur la session existante.
- CipherSuite : Une liste, par ordre décroissant de préférence, des algorithmes que supporte le client. Il s'agit des algorithmes d'échange de clé et de chiffrement. Voir tableau 4.1.
- Compression Method : liste, par ordre décroissant de préférence, des algorithmes de compression supportés par le client. Voir tableau 4.1

Après avoir envoyé ces requêtes, le client attend que le serveur réponde en générant une valeur aléatoire, en indiquant la version, et les meilleurs algorithmes qu'il peut utiliser : ce sera la réponse « server hello » du serveur

Phase 2 : Authentification du serveur et échange des clés :

Le serveur répond par une série des messages qui définissent les paramètres de sécurité du serveur:

- Il commence par un message du type « Server Hello ». Celui-ci fait savoir au client que le serveur a bien reçu son « Client Hello » (en envoyant le même numéro de session), envoie également un nombre aléatoire et indique l'algorithme de compression et l'algorithme de sécurité qu'il a choisi parmi ceux proposés par le client.
- Le serveur envoie ensuite un message du type « Certificate » pour s'identifier si le client l'a demandé.
- Le serveur envoie alors un message du type « Server Key Exchange » pour transmettre les clés de sécurité si le client l'a demandé.
- Le serveur envoie éventuellement un message du type « Certificate Request » s'il désire que le client s'identifie.
- Le serveur indique enfin qu'il a terminé sa série de messages en envoyant un message du type « Server Hello Done ». Ceci permet au client de savoir qu'il n'a pas à attendre un message du type « Certificate Request ».

Phase 3 : Authentification du client et échange des clés :

Le client doit vérifier que le certificat envoyé par le serveur est valide. Puis, le client s'embarque dans une série de messages qui définissent les paramètres de sécurité du client :

- Le message « Certificate » identifie le client, uniquement en réponse à un « Certificate Request ».
- Le message « Client Key exchange » permet de transmettre les clés de sécurité. Le contenu de ce message dépend de l'algorithme de clé publique sélectionné entre le Hello du client et les messages Hello du serveur. Le client produit un secret pré-maître (encrypted pre-master key). Quand la RSA est utilisée pour l'authentification du serveur et l'échange de clés, un **pre_master_secret** 48-byte est généré par le client, chiffré sous la clé publique du serveur, et envoyé au serveur. Le serveur emploie la clé privée afin de déchiffrer le **pre_master_secret**. Les deux interlocuteurs convertissent alors le **pre_master_secret** en **master_secret.**

- Le client confirme qu'il a accepté le certificat envoyé par le serveur (si c'est le cas) en envoyant un message du type « Certificate Verify », message contenant une empreinte (hash) signée numériquement et créer à partir des informations de clé et de tous les messages précédents. Ce message permet de confirmer au serveur que le client possède bien la clé privée correspondant au certificat client

Phase 4 : Fin

Le client envoie le message (1 octet) change_cypher_spec, qui est en fait l'unique message du protocole Change Cipher Spec, et active pour la session courante les algorithmes, clés et sels du handshake. Puis le client envoie le message finished, qui authentifie le client et valide l'échange de clés (le message est constitué d'un condensât de la clé symétrique échangée, de l'ensemble des messages échangées durant le handshake, du pad et d'un identificateur de l'expéditeur).

Le serveur répond en envoyant son propre change_cypher_spec et clos le handshake L'ordre des messages est imposé. La suite de message est montrée à la figure 4.2, elle peut être réinitialisée durant la session TCP sécurisée pour renégocier les paramètres de sécurité. Maintenant les données peuvent être échangées.

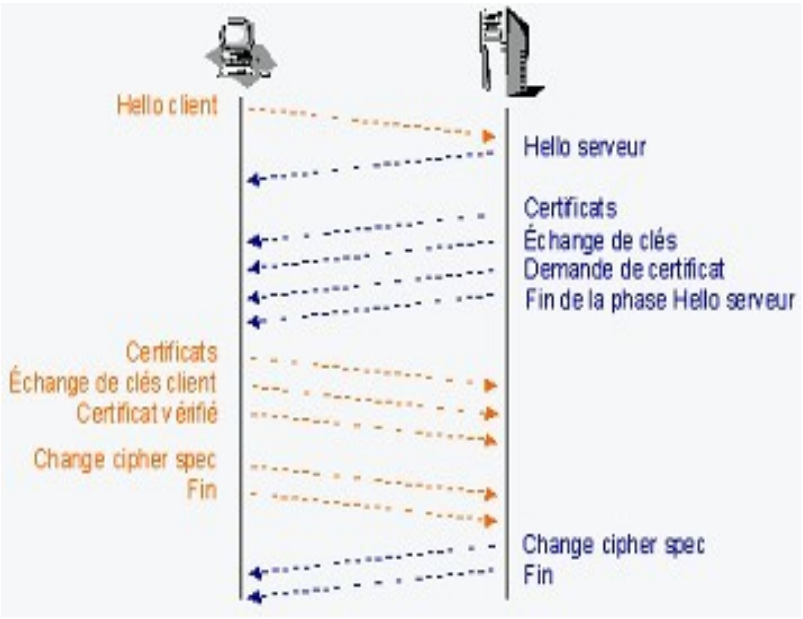


Figure 4.2 : Principe de négociation SSL/TLS[21][20][22]

Cipher ID	Name	Protocol	Kx	Au	Enc	Bits	Mac
0x000000	TLS_NULL_WITH_NULL_NULL	TLS	NULL	NULL	NULL	0	NULL
0x000001	TLS_RSA_WITH_NULL_MD5	TLS	RSA	RSA	NULL	0	MD5
0x000002	TLS_RSA_WITH_NULL_SHA	TLS	RSA	RSA	NULL	0	SHA
0x000003	TLS_RSA_EXPORT_WITH_RC4_40_MD5	TLS	RSA_EXPORT	RSA_EXPORT	RC4_40	40	MD5
0x000004	TLS_RSA_WITH_RC4_128_MD5	TLS	RSA	RSA	RC4_128	128	MD5
0x000005	TLS_RSA_WITH_RC4_128_SHA	TLS	RSA	RSA	RC4_128	128	SHA
0x000006	TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5	TLS	RSA_EXPORT	RSA_EXPORT	RC2_CBC_40	40	MD5
0x000007	TLS_RSA_WITH_IDEA_CBC_SHA	TLS	RSA	RSA	IDEA_CBC	128	SHA
0x000008	TLS_RSA_EXPORT_WITH_DES40_CBC_SHA	TLS	RSA_EXPORT	RSA_EXPORT	DES40_CBC	40	SHA
0x000009	TLS_RSA_WITH_DES_CBC_SHA	TLS	RSA	RSA	DES_CBC	56	SHA
0x00000A	TLS_RSA_WITH_3DES_EDE_CBC_SHA	TLS	RSA	RSA	3DES_EDE_CBC	168	SHA
0x00000B	TLS_DH_DSS_EXPORT_WITH_DES40_CBC_SHA	TLS	DH	DSS	DES40_CBC	40	SHA
0x00000C	TLS_DH_DSS_WITH_DES_CBC_SHA	TLS	DH	DSS	DES_CBC	56	SHA
0x00000D	TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA	TLS	DH	DSS	3DES_EDE_CBC	168	SHA
0x00000E	TLS_DH_RSA_EXPORT_WITH_DES40_CBC_SHA	TLS	DH	RSA	DES40_CBC	40	SHA
0x00000F	TLS_DH_RSA_WITH_DES_CBC_SHA	TLS	DH	RSA	DES_CBC	56	SHA
0x000010	TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA	TLS	DH	RSA	3DES_EDE_CBC	168	SHA
0x000011	TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA	TLS	DHE	DSS	DES40_CBC	40	SHA
0x000012	TLS_DHE_DSS_WITH_DES_CBC_SHA	TLS	DHE	DSS	DES_CBC	56	SHA
0x000013	TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	TLS	DHE	DSS	3DES_EDE_CBC	168	SHA
0x000014	TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA	TLS	DHE	RSA	DES40_CBC	40	SHA
0x000015	TLS_DHE_RSA_WITH_DES_CBC_SHA	TLS	DHE	RSA	DES_CBC	56	SHA
0x000016	TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	TLS	DHE	RSA	3DES_EDE_CBC	168	SHA

Table 4.1 : Exemple D'Algorithmes de sécurisation [21][20][22]

4.5.2. Le protocole SSL Alert

Ce protocole spécifie les messages d'erreurs que peuvent s'envoyer clients et serveurs. Les messages sont composés de deux octets, le premier est soit "warning" soit "fatal". Si le niveau est "fatal", la connexion est abandonnée. Les autres connexions sur la même session ne sont pas coupées mais on ne peut pas en établir de nouvelles. Le deuxième octet donne le code d'erreur.

Les erreurs fatales sont :

- ***unexpected_message*** : message non reconnu.
- ***bad_record_mac***: MAC non correct.
- ***decompression_failure*** : la fonction de décompression à reçu une mauvaise entrée.
- ***handshake_failure*** : impossible de négocier les bons paramètres.
- ***illegal_parameter*** : un champ était mal formaté ou ne correspondait à rien.

Les warnings sont :

- ***close_notify*** : annonce la fin d'une connexion.
- ***no_certificate*** : réponse à une demande de certificat s'il n'y en a pas.
- ***bad_certificate*** : le certificat reçu n'est pas bon (e.g. signature non valide)
- ***unsupported_certificate*** : le certificat reçu n'est pas reconnu.
- ***certificate_revoked*** : cert révoqué par l'émetteur.
- ***certificate_expired***.
- ***certificate_unknown*** : tous les problèmes concernant les certificats et non listés au dessus.

4.5.3. Le protocole SSL Record

Ce protocole permet de garantir :

- La confidentialité des données transmises (c'est le Handshake qui permet de négocier une clé symétrique partagée)
- L'intégrité des données transmises (c'est encore le Handshake qui négocie une clé partagée qui sert à faire des MAC sur les données)

Le flux des données est transmis sous la forme d'une série de fragments, chaque fragment étant protégé et transmis de façon individuelle. Pour transmettre un fragment, on commence par calculer son MAC. La concaténation du fragment et de son MAC est cryptée, obtenant ainsi une charge cryptée (encrypted payload en anglais). On lui attache un en-tête, obtenant ainsi un enregistrement SSL.

Les étapes de ce protocole sont :

Etape 1 :

Les blocs de données applicatives sont découpés en fragments de 16Ko (2^14 octets) maximum

Etape 2

Le fragment subit une compression (facultative). Si les blocs sont très courts, cette compression peut augmenter la taille du bloc initial. Toutefois, la longueur du contenu ne peut être augmentée de plus de 1024 octets.

Etape 3

Une empreinte est prise du fragment compressé. L'algorithme qui prend cette empreinte est choisi dans le message “Cipher Suite” du Handshake.

Etape 4

L'empreinte est ajoutée au message compressé formant ainsi un message M1

Etape 5

Le message M.1 est chiffré par l'algorithme symétrique, en utilisant la clé symétrique échangée lors du handshake. La résultante est le message Mc.1, crypté

Etape 6

Un en-tête de 5 octets est ajouté. Dans l'entête qui est ajoutée on trouve :

- Content Type : le plus haut protocole utilisé pour traiter ce fragment
- Major Version : plus haute version de SSL utilisée (3 pour SSLv3)
- Minor Version : plus basse version utilisée (0 pour SSLv3)
- Compressed Lenght : la longueur en octets des données (éventuellement compressées) à chiffrer.

Ces étapes sont illustrées dans la figure :

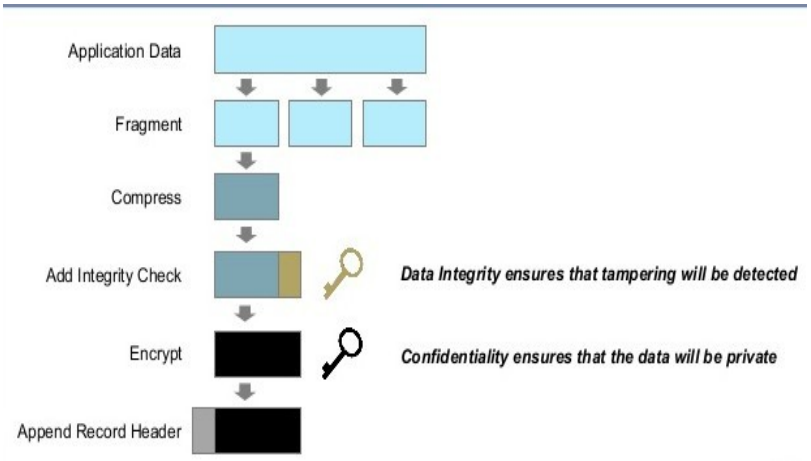


Figure 4.3. Fragmentation des données SSL
[21][20][22]

4.6. Calcule des MACs (*Message Authentication code) [23]*

Un **code d'authentification de message** est un code accompagnant des données dans le but d'assurer l'intégrité de ces dernières, en permettant de vérifier qu'elles n'ont subi aucune modification, après une transmission par exemple.

Le concept est relativement semblable aux fonctions de hachage. Il s'agit ici aussi d'algorithmes qui créent un petit bloc authentificateur de taille fixe. La grande différence est que ce bloc authentificateur ne se base plus uniquement sur le message, mais également sur une clé secrète (la clé secrète est générée dans le protocole handshake).

Tout comme les fonctions de hachage, les MACs n'ont pas besoin d'être réversibles. En effet, le récepteur exécutera le même calcul sur le message et le comparera avec le MAC reçu.

Le MAC assure non seulement une fonction de vérification de l'intégrité du message, comme le permettrait une simple fonction de hachage mais de plus authentifie l'expéditeur, détenteur de la clé secrète. Il peut également être employé comme un chiffrement supplémentaire (rare) et peut être calculé avant ou après le chiffrement principal, bien qu'il soit généralement conseillé de le faire avant

TP2

Autorité de certification/Le protocole SSL

(Ce TP s'effectuera individuellement, avec un PC sous Linux. Un rapport détaillé doit être fourni + une validation)

Le travail demandé dans ce TP est :

- Création d'une autorité de certification
- Mise en place d'un site sécurisé qui utilise les certificats de cette autorité

Ce TP comporte 4 parties :

- Partie 1 : Configuration du serveur Apache
- Partie 2: Configuration du serveur Apache avec SSL
- Partie 03 : Création d'autorité de certification
- Partie 04: Configuration des certificats sous Apache

.

Corrigé TP2

Autorité de certification/Le protocole SSL

Partie 1 : Configuration du serveur Apache

Dans un premier temps, nous allons mettre en place un serveur Web qui va prendre en charge l'affichage du site www.monsite.com.

1. Lancer le serveur Apache. Avec la commande : ***sudo service apache2 start***
2. Vérifier l'état du serveur. Avec la commande : ***service apache2 status***
3. Tester le fonctionnement du serveur : Avec le navigateur en vas à l'adresse http://127.0.0.1/, Le contenu du site par défaut (d'adresse 127.0.0.1) est disponible dans le répertoire /var/www/. La page qui s'affiche lorsque l'on se rend à l'adresse 127.0.0.1 est décrite dans le fichier /var/www/index.html.
4. Création d'un site web : Créer un répertoire /var/www/monsite qui contiendra les fichiers de notre site avec la commande : ***sudo mkdir /var/www/monsite***

5. Créer un fichier dans ce répertoire « index.html » avec la commande : ***sudo touch /var/www/monsite/index.html***. Puis éditer le avec la commande : ***sudo gedit /var/www/monsite/index.html***
6. La Configuration de notre site web : Copier le fichier de configuration du site par défault(/etc/apache2/sites- available/000-default.conf) dans le fichier de configuration de notre site (/etc/apache2/sites-available/001-default.conf) avec la commande : ***sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/001-default.conf***
7. Éditer le fichier de configuration de notre site (c'est-à-dire /etc/apache2/sites-available/001-default.conf). avec la commande ***gedit*** de la façon suivante :

- À la ligne ServerAdmin, indiquer une adresse électronique du domaine : @monsite.com
- Ajouter une ligne ServerName et indiquer le nom de domaine : www.monsite.com.
- Ajouter une ligne ServerAlias, et indiquer : monsite.com
- Spécifier la racine des fichiers du site DocumentRoot : /var/www/monsite

8. Indiquer à la machine que le nom de domaine www.monsite.com pointe en local, en éditant le fichier /etc/hosts, et en ajoutant www.monsite.com monsite.com à la première ligne.
9. Activer le site www.monsite.com sur Apache grâce à la commande suivante :

sudo a2ensite 001-default.conf. Recharger ensuite la configuration avec la commande : ***sudo service apache2 reload***.

10. Ouvrir votre navigateur Web préféré. Vérifier que l'adresse http://www.monsite.com affiche bien la page attendue.

Partie 2: Configuration du serveur Apache avec SSL

Nous allons maintenant autoriser SSL sur Apache pour permettre l'accès à https://www.monsite.com

1. Faut activer le module SSL avec la commande : ***'sudo a2enmod ssl'***
2. Puis recharger la configuration d'Apache2 : ***'sudo service apache2 force-reload'***.

Partie 03 : Création d'autorité de certification

1. Ouvrir tinyCA à l'aide de la commande tinyca2.
2. Nous allons d'abord créer une autorité de certification racine. Cette autorité sera la principale autorité, dont vous pourrez exporter et utiliser les certificats. À l'ouverture de tinyCA, une fenêtre de création d'autorité de certification s'ouvre. Dans les champs « Nom » et « Informations », indiquer ***ROOTCA-TP***. Dans le champ « Pays », indiquer ***DZ***. Choisir un mot de passe ; il est (exceptionnellement ! !) conseillé de le noter dans le cadre de ce TP, car vous allez créer d'autres mots de passe, et vous en aurez besoin plus tard. Dans le champ « Localité », indiquer ***Sidi Amar***. Dans le champ « Unité Organisationnelle », indiquer ***université Badji mokhtar***. Dans le champ « Adresse eMail », entrez une adresse électronique. Laisser les autres champs aux valeurs par défaut (notamment, 4096 et SHA1, qui sont de bons choix). Lorsque la seconde fenêtre s'affiche, laisser les valeurs par défaut.
3. Nous allons ensuite créer une sous-autorité de certification (***sub-CA***), dédiée à notre site www.monsite.com. Cliquer sur l'onglet « CA », puis cliquer sur la 3e icône en partant de la droite dans la barre d'outils. Dans le champ « mot de passe », indiquer le mot de passe de l'autorité de certification racine. (créée plus haut). Dans les champs « Nom » et « Informations », indiquer ***Sub-CA***. Dans le champ « Pays », indiquer ***Dz***. Choisir un second mot de passe (et le noter exceptionnellement dans un coin). Dans le champ « Localité », indiquer ***Sidi Amar***. Dans le champ « Unité Organisationnelle », indiquer ***université Badji Mokhtar***. Dans le champ « Adresse eMail », entrez une adresse électronique. Laisser les autres champs aux valeurs par défaut.
4. Nous allons maintenant créer un certificat pour notre site www.monsite.com Sélectionner l'onglet « Requêtes de certification » et cliquer sur l'icône « Nouvelle requête de certification » (4e icône en partant de la droite dans la barre d'outils). Dans le champ « Nom », entrer www.monsite.com. Dans le champ « Email », entrer la même adresse électronique que celle indiquée dans le fichier de configuration .Choisir un troisième mot de passe (et le noter exceptionnellement dans un coin).
5. Nous allons maintenant signer la requête. Effectuer un clic droit puis « signer la requête ». Quand on vous demande de choisir entre un client et un serveur,

sélectionner « serveur ». Le mot de passe demandé sera celui de la sous-autorité de certification (donc le 2e mot de passe).

6. Nous allons maintenant exporter le certificat racine. Dans le menu CA, sélectionner « ouvrir CA » puis sélectionner l'autorité de certification racine. Sélectionner l'icône d'export (deuxième icône en partant de la droite). Garder les options par défaut (PEM cochée).
7. Nous allons maintenant exporter le certificat du site. Dans le menu CA, sélectionner « ouvrir CA » puis sélectionner la sous-autorité de certification. Sélectionner l'onglet Certificats, puis clic droit et « Export de certificat ». Garder les options par défaut (PEM cochée / non / non). Vous pouvez exporter par exemple dans /home/TP/Sub-CA-cert.pem.
8. Nous allons maintenant exporter la clé du site. Sélectionner l'onglet Clés, puis clic droit et « Export de la clé ». Choisir les options PEM cochée / oui / non. (Indiquer « sans mot de passe » permet d'éviter de saisir la clé lors du démarrage du serveur.) Vous pouvez exporter par exemple dans /home/TP/Sub-CA-cle.pem. Entrer le mot de passe de la clé (3e mot de passe).
9. Enfin, nous allons exporter la chaîne de certification du site. Sélectionner l'onglet CA, puis cliquer sur l'icône d'export de la chaîne (icône la plus à droite). Vous pouvez exporter par exemple dans /home/TP/Sub-CA-chaine.pem.

Partie 4: Configuration des certificats sous Apache :

1. Copier les trois fichiers du site (c'est-à-dire tous les fichiers .pem à l'exception de rootca.pem) dans /etc/ssl/private/ avec la commande : « *sudo cp ……..pem /etc/ssl/private* »

2. Ouvrir le fichier de configuration de votre site (*/etc/apache2/site-available/001-default.conf*), et modifier les paramètres suivants :

SSLCertificateFile : /etc/ssl/private/certificat du site

SSLCertificateKeyFile /etc/ssl/private/ la clef du certificat

SSLCertificateChainFile /etc/ssl/private/la chaine

```
<VirtualHost 127.0.0.1:443>
   ServerName www.monsite.com
        DocumentRoot /var/www/monsite
    SSLEngine on
    SSLCertificateFile /etc/ssl/private/amira_bendjeddou@yahoo.fr-cert.pem
    SSLCertificateKeyFile /etc/ssl/private/amira_bendjeddou@yahoo.fr-key.pem
SSLCertificateChainFile /etc/ssl/private/Sub-CA-cachain.pem
</VirtualHost>
```

3. Ouvrir votre navigateur Web préféré (par exemple firefox). Aller sur https://www.monsite.com. Que se passe-t-il ?

Partie 5 : Configuration du navigateur Web

1. Ouvrir le navigateur. Ouvrir « Préférences » puis « Avancé » puis onglet « Certificats » puis « Afficher les certificats » puis onglet « Autorités » puis « Importer. . . », Importer le certificat racine, puis Cocher les 3 cases. Le certificat apparaître dans la liste

- NOTE : le certificat doit être en format pem, mais en extension .crt.

CHAPITRE 5

LE PROTOCOLE IPSEC

Chapitre 5

Le protocole IPsec

5.1. Introduction

Le terme IPsec (IP Security Protocol) désigne un ensemble de mécanismes destinés à protéger le trafic au niveau d'IP (IPv4 ou IPv6). Les services de sécurité offerts sont l'intégrité en mode non connecté, l'authentification de l'origine des données, la protection contre le rejeu et la confidentialité (confidentialité des données et protection partielle contre l'analyse du trafic). Ces services sont fournis au niveau de la couche IP, offrant donc une protection pour IP et tous les protocoles de niveau supérieur. Optionnel dans IPv4, IPsec est obligatoire pour toute implémentation d'IPv6. Une fois IPv6 en place, il sera ainsi possible à tout utilisateur désirant des fonctions de sécurité d'avoir recours à IPsec.

IPsec est développé par un groupe de travail du même nom à l'IETF (Internet Engineering Task Force), groupe qui existe depuis 1992. Une première version des mécanismes proposés a été publiée sous forme de RFC en 1995, sans la partie gestion des clefs. Une seconde version, qui comporte en plus la définition du protocole de gestion des clefs IKE, a été publiée en novembre 1998. Mais IPsec reste une norme non figée qui fait en ce moment même l'objet de multiples *Internet drafts*, notamment sur la protection des accès distants. [24][25]

5.2. Pourquoi IPsec ?

5.2.1.Les faiblesses d'IP

Originellement IP est le principal protocole d'interconnexion des machines informatiques. Les spécifications d'IP étaient axées essentiellement sur la robustesse (routage dynamique, par exemple) et non sur la sécurité (ni la fiabilité). Ainsi actuellement le protocole IP est sujet à de nombreux problèmes de sécurité dont nous décrirons quelques exemples. IP étant utilisé aussi bien pour les communications par l'Internet qu'au sein des réseaux d'entreprises, ces problèmes peuvent avoir de larges répercussions.

a. L'écoute des paquets

Sur IP, les données transportées sont contenues en clair dans les paquets, il est donc possible si on peut accéder aux paquets IP qu'échangent deux machines de connaître toutes les informations échangées entre ces deux machines. Il existe des outils pour accéder aux paquets en cours de transmission ("sniffer") comme, par exemple :

- Ethereal qui permet de visualiser les paquets et leur contenu. Il est capable de réassembler le message TCP d'origine à partir des paquets IP.
- Dsniff qui peut récupérer des mots de passe sur le réseau, est capable de clore n'importe quelle connexion TCP. Dsniff implémente aussi une attaque dite "de l'homme au milieu" (man-in-the-middle) sur SSH qui permet notamment de modifier

les paquets lors du transit sans que les deux machines communicantes puissent le savoir. Lorsque deux ordinateurs de deux réseaux différents communiquent les paquets peuvent par exemple transiter successivement sur le réseau de l'entreprise puis sur celui du fournisseur d'accès Internet (FAI) puis sur la dorsale de l'Internet (backbone) puis sur le FAI de l'autre réseau puis sur les réseaux de la machine destinatrice. Chaque routeur, ou chaque brin intermédiaire, sont autant de points où il est possible, pour quelqu'un mal-intentionné, d'écouter les communications.

b. L'usurpation d'adresse

Un autre problème d'IP est le contrôle des adresses sources. Les routeurs ne vérifient pas l'adresse source des paquets qui transitent. Une machine pirate peut donc émettre des paquets IP ayant comme source l'adresse d'une autre machine, le paquet arrivera bien à destination et le destinataire ne pourra pas identifier la véritable source. Dans le cadre d'UDP ou d'ICMP cela peut poser des problèmes importants car des notifications peuvent être faites (ICMP Redirect par exemple). Beaucoup des causes de déni de service (DoS) sont dues à ce problème d'usurpation d'adresse, une ou des personnes malintentionnées bombardent une cible avec des paquets dont l'adresse source est fausse. La machine cible ne peut pas déceler qui attaque et donc remédier à l'attaque

5.2.2. Nouveaux besoins

Aux vues de ces faiblesses, on peut définir de nouveaux besoins auxquels doit répondre IP[25] :

- **Confidentialité** : La capture des paquets ne doit pas permettre de savoir quelles sont les informations échangées. Seules les machines réceptrices et émettrices doivent pouvoir accéder à l'information.
- **Authentification**: Le récepteur doit être capable de vérifier si les données reçues proviennent bien de l'émetteur supposé.
- **Intégrité** : Le récepteur doit être capable de vérifier si les données n'ont pas été modifiées lors de la transmission.
- **Protection contre le rejeu** : Une personne qui intercepte un message d'une communication sécurisée entre deux machines ne pourra pas retransmettre ce message sans que cela soit détecté.

5.3. Architecture d'IPsec

Pour sécuriser les échanges ayant lieu sur un réseau TCP/IP, il existe plusieurs approches, en particulier en ce qui concerne le niveau auquel est effectuée la sécurisation :

Niveau applicatif (mails chiffrés par exemple), niveau transport (TLS/SSL, SSH…), ou à l'opposé niveau physique (boîtiers chiffrant toutes les données transitant par un lien donné). IPsec, quant à lui, vise à sécuriser les échanges au niveau de la couche réseau.

5.3.1.Les mécanismes AH et ESP

IPsec fait appel à deux mécanismes de sécurité pour le trafic IP, les “protocoles” AH et ESP, qui viennent s’ajouter au traitement IP classique :

- Authentication Header (AH) : est conçu pour assurer l’intégrité et l’authentification des datagrammes IP sans chiffrement des données (i.e. sans confidentialité). Le principe d’AH est d’adjoindre au datagramme IP classique un champ supplémentaire permettant à la réception de vérifier l’authenticité des données incluses dans le datagramme.
- Encapsulating Security Payload (ESP) : a pour rôle premier d’assurer la confidentialité, mais peut aussi assurer l’authenticité des données. Le principe d’ESP est de générer, à partir d’un datagramme IP classique, un nouveau datagramme dans lequel les données et éventuellement l’en-tête original sont chiffrés.

5.3.2.La notion d’Association de sécurité

Les mécanismes mentionnés ci-dessus font bien sûr appel à la cryptographie, et utilisent donc un certain nombre de paramètres (algorithmes de chiffrement utilisés, clefs, mécanismes sélectionnés...) sur lesquels les tiers communicants doivent se mettre d’accord. Afin de gérer ces paramètres, IPsec a recours à la notion d’association de sécurité (Security Association, SA)[22].

Une association de sécurité IPsec est une “connexion” simplexe qui fournit des services de sécurité au trafic qu’elle transporte. On peut aussi la considérer comme une structure de données servant à stocker l’ensemble des paramètres associés à une communication donnée.

Une SA est unidirectionnelle ; en conséquence, protéger les deux sens d’une communication classique requiert deux associations, une dans chaque sens Les services de sécurité sont fournis par l’utilisation soit de AH soit de ESP. Si AH et ESP sont tous deux appliqués au trafic en question, deux SA sont créées ; on parle alors de paquet (bundle) de SA [26].

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

- L’adresse de destination des paquets.
- L’identifiant d’un protocole de sécurité utilisé (AH ou ESP).
- Un index des paramètres de sécurité (Security Parameter Index, SPI).

Un SPI est un bloc de 32 bits inscrit en clair dans l’en-tête de chaque paquet échangé ; il est choisi par le récepteur.

Pour gérer les associations de sécurité actives, on utilise une “base de données des associations de sécurité” (Security Association Database, SAD). Elle contient tous les paramètres relatifs à chaque SA et sera consultée pour savoir comment traiter chaque paquet reçu ou à émettre.

5.3.3.La gestion des clefs et des associations de sécurité

Comme nous l’avons mentionné au paragraphe précédent, les SA contiennent tous les paramètres nécessaires à IPsec, notamment les clefs utilisées. La gestion des clefs pour IPsec n’est liée aux autres mécanismes de sécurité de IPsec que par le biais des SA.

Une SA peut être configurée manuellement dans le cas d'une situation simple, mais la règle générale est d'utiliser un protocole spécifique qui permet la négociation dynamique des SA et notamment l'échange des clefs de session. D'autre part, IPv6 n'est pas destiné à supporter une gestion des clefs "en bande", c'est-à-dire où les données relatives à la gestion des clefs seraient transportées à l'aide d'un en-tête IPv6 distinct. Au lieu de cela on utilise un système de gestion des clefs dit "hors bande", où les données relatives à la gestion des clefs sont transportées par un protocole de couche supérieure tel qu'UDP ou TCP.

Ceci permet le découplage clair du mécanisme de gestion des clefs et des autres mécanismes de sécurité. Il est ainsi possible de substituer une méthode de gestion des clefs à une autre sans avoir à modifier les implémentations des autres mécanismes de sécurité.

Le protocole de négociation des SA développé pour IPsec s'appelle "protocole de gestion des clefs et des associations de sécurité pour Internet" (Internet Security Association and Key Management Protocol, ISAKMP). ISAKMP est en fait inutilisable seul : c'est un cadre générique qui permet l'utilisation de plusieurs protocoles d'échange de clef et qui peut être utilisé pour d'autres mécanismes de sécurité que ceux de IPsec.

Dans le cadre de la standardisation de IPsec, ISAKMP est associé à une partie des protocoles SKEME et Oakley pour donner un protocole final du nom d'IKE (Internet Key Exchange).

5.3.4.Politiques de sécurité[24]

Les protections offertes par IPsec sont basées sur des choix définit dans une base de données de politique de sécurité (Security Policy Database, SPD). Elle permet de décider, pour chaque paquet, s'il se verra apporter des services de sécurité, s'il sera autorisé à passer ou sera rejeté.

La SPD contient une liste ordonnée de règles, chaque règle comportant un certain nombre de critères qui permettent de déterminer quelle partie du trafic est concernée. Les critères utilisables sont l'ensemble des informations disponibles par le biais des en-têtes des couches IP et transport. Ils permettent de définir la granularité selon laquelle les services de sécurité sont applicables et influencent directement le nombre de SA correspondante. Dans le cas où le trafic correspondant à une règle doit se voir attribuer des services de sécurité, la règle indique les caractéristiques de la SA (ou paquet de SA) correspondante : protocole(s), modes, algorithmes requis.

5.4. Principe de fonctionnement [27]

La figure 5.1, ci-dessous, représente tous les éléments présentés ci-dessus (en bleu), leurs positions et leurs interactions.

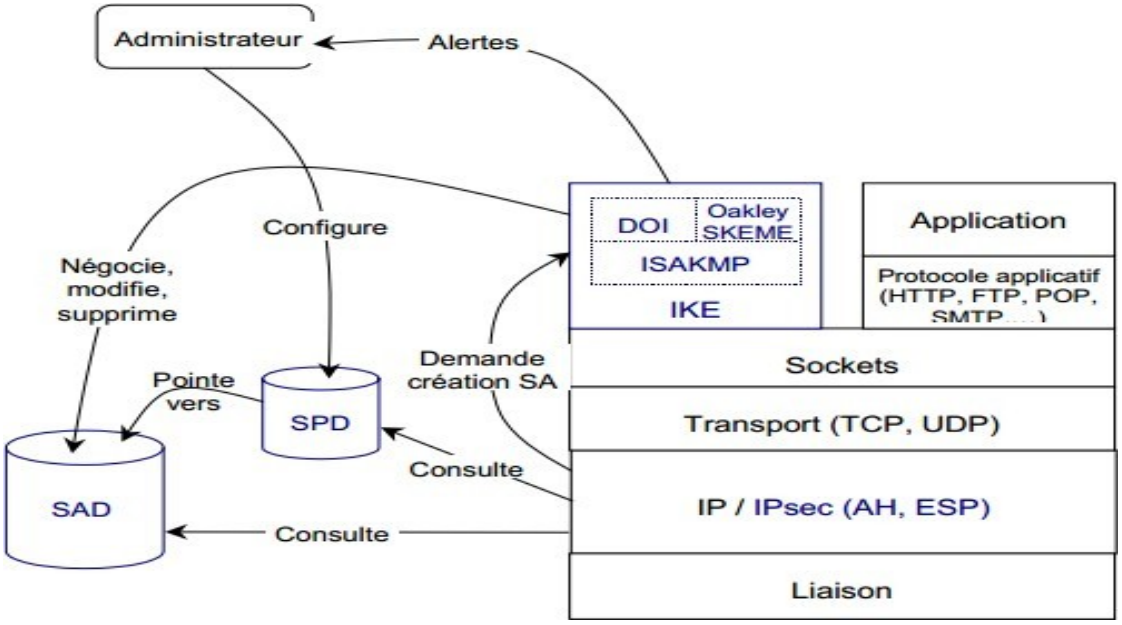


Figure 5.1. Schéma global d'IPsec[27]

On distingue deux situations :

- Trafic sortant :

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

- Trafic entrant :

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

5.5. Modes de fonctionnements

5.5.1. Le mode transport [28][29]

Dans le mode transport, IPSec intervient entre le niveau transport (TCP) et le niveau réseau (IP) du modèle OSI : le PDU de la couche transport se voit appliqué les mécanismes de signature et de chiffrement puis le résultat est passé à la couche réseau (encapsulation IP). Ce mode ne résout pas un problème majeur en matière de sécurité : l'en-tête du paquet est inchangé puisque produit par la couche IP. Il n'y a donc ni de masquage d'adresse ni de protection des options IP. Cependant ce mode est relativement aisé à mettre en œuvre. En mode transport, seules les données en provenance du protocole de niveau supérieur et transportées par le datagramme IP sont protégées. Ce mode n'est utilisable que sur des équipements terminaux ; en effet, en cas d'utilisation sur des équipements intermédiaires, on courrait le risque, suivant les aléas du routage, que le paquet atteigne sa destination finale sans avoir traversé la passerelle sensée le déchiffrer.

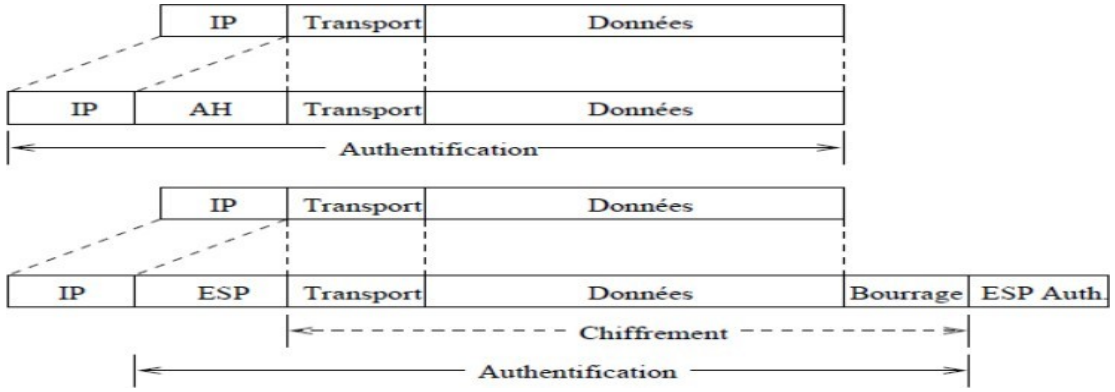


Figure 5.2. Le mode transport[30]

5.5.2.Le mode tunnel

Dans le mode tunnel, IPSec agit directement après l'encapsulation IP. La totalité du paquet IP est encapsulé dans un paquet IPSec sécurisé. Dans ce cas, l'en-tête IP d'origine est protégé et les adresses sont masquées. Ce mode est très utilisé pour la mise en place de VPNs. En mode tunnel, l'en-tête IP est également protégé (authentification, intégrité et/ou confidentialité) et remplacé par un nouvel en-tête. Ce nouvel en-tête sert à transporter le paquet jusqu'à la fin du tunnel, où l'en-tête original est rétabli. Le mode tunnel est donc utilisable à la fois sur des équipements terminaux et sur des passerelles de sécurité. Ce mode permet d'assurer une protection plus importante contre l'analyse du trafic, car il masque les adresses de l'expéditeur et du destinataire final.

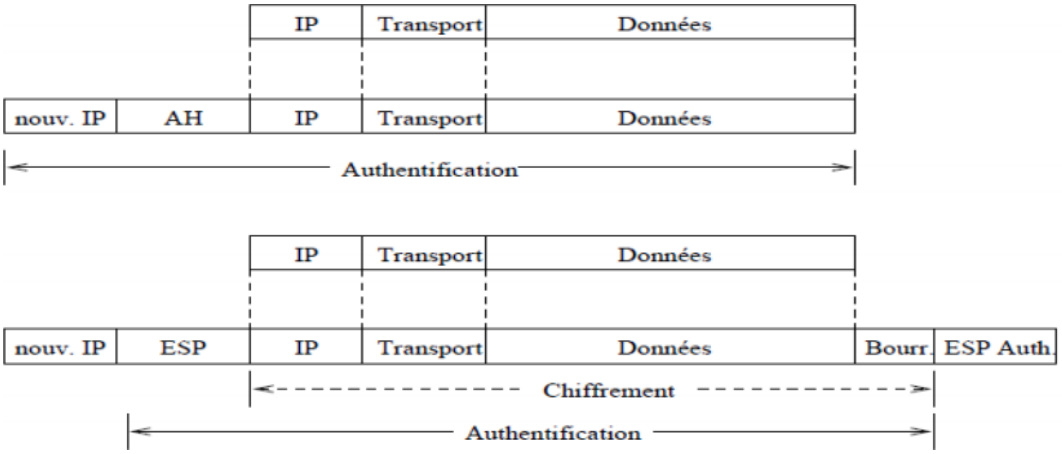


Figure 5.3. Le mode tunnel[30]

5.6. Les mécanismes de sécurité AH et ESP

5.6.1. Le mécanisme AH [31]

AH assure l'intégrité des données en mode non connecté, l'authentification de l'origine des données et, de façon optionnelle, la protection contre le rejeu. L'absence de confidentialité dans AH permet de s'assurer que ce standard pourra être largement répandu sur Internet, y compris dans les endroits où l'exportation, l'importation ou l'utilisation du chiffrement dans des buts de confidentialité est restreint par la loi. Cela constitue l'une des raisons de l'utilisation de deux mécanismes distincts.

Dans AH, intégrité et authentification sont fournies ensembles, à l'aide d'un bloc de données supplémentaire adjoint au message à protéger. Ce bloc de données est appelé "valeur de vérification d'intégrité" (Integrity Check Value, ICV), terme générique pour désigner soit un code d'authentification de message (Message Authentication Code, MAC), soit une signature numérique.

Pour des raisons de performances, les algorithmes proposés actuellement sont tous des algorithmes de scellement (code d'authentification de message et non signature).

La protection contre le rejeu se fait grâce à un numéro de séquence ; elle n'est disponible que si IKE est utilisé, car en mode manuel aucune "ouverture de connexion" ne permet d'initialiser le compteur. Voici l'organisation de l'en-tête d'authentification :

En-tête suivant	longueur	Réservé
Index des paramètres de sécurité (SPI)		
Numéro de séquence		
Données d'authentification (longueur variable)		

Figure 5.4. En-tête du AH[31]

L'expéditeur calcule les données d'authentification à partir de l'ensemble des champs invariants du datagramme IP final, AH compris, ce qui permet d'étendre l'authentification au SPI et au numéro de séquence notamment.

Les champs variables (TTL, routage...) et le champ destiné à recevoir les données d'authentification sont considérés comme égaux à zéro pour le calcul. Les données d'authentification sont alors adjointes au paquet IP par le biais de l'en-tête d'authentification(AH). Le récepteur vérifie l'exactitude de ces données à la réception. Les paramètres de sécurité liés à la communication sont identifiés par un identifiant unique (Security Parameters Index) caractérisant la Security Association (SA). C'est une combinaison de l'adresse du destinataire et du protocole utilisé.

5.6.2.Encapsulating Security Payload (ESP)[32]

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

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

La confidentialité peut être sélectionnée indépendamment des autres services, mais son utilisation sans intégrité/authentification (directement dans ESP ou avec AH) rend le trafic vulnérable à certains types d'attaques actives qui pourraient affaiblir le service de confidentialité.

Comme dans AH, authentification et intégrité sont deux services qui vont de pair et que l'on désigne souvent sous le terme "authentification" ; ils sont fournis par l'utilisation d'une ICV (en pratique, un MAC). La protection contre le rejeu ne peut être sélectionnée que si l'authentification l'a été et que IKE est utilisé. Elle est fournie par un numéro de séquence que le destinataire des paquets vérifie.

Contrairement à AH, où l'on se contentait d'ajouter un en-tête supplémentaire au paquet IP, ESP fonctionne suivant le principe de l'encapsulation : les données originales sont chiffrées puis encapsulées entre un en-tête et un trailer. Voici l'organisation d'ESP :

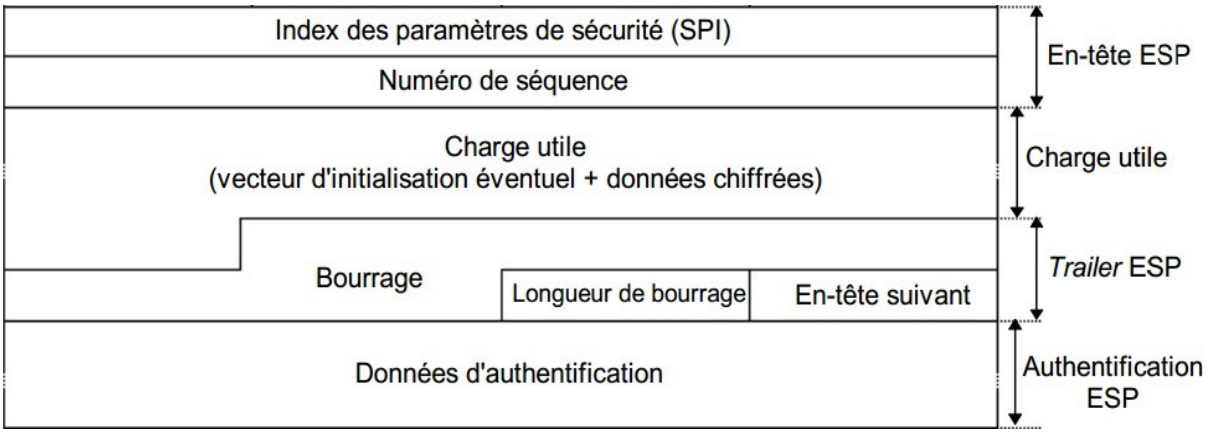


Figure 5.5. Format d'ESP[32]

Le champ bourrage peut être nécessaire pour les algorithmes de chiffrement par blocs ou pour aligner le texte chiffré sur une limite de 4 octets.

Voyons maintenant comment est appliquée la confidentialité dans ESP. L'expéditeur :

- Encapsule, dans le champ "charge utile" d'ESP, les données transportées par le datagramme original et éventuellement l'en-tête IP (mode tunnel).
- Ajoute si nécessaire un bourrage.
- Chiffre le résultat (données, bourrage, champs longueur et en-tête suivant)
- Ajoute éventuellement des données de synchronisation cryptographiques (vecteur d'initialisation) au début du champ "charge utile"

Si elle a été sélectionnée, l'authentification est toujours appliquée après que les données ne soient chiffrées. Cela permet, à la réception, de vérifier la validité du datagramme avant de se lancer dans la coûteuse tâche de déchiffrement. Contrairement à AH, l'authentification dans ESP est appliquée uniquement sur le "paquet" (en-tête + charge utile + trailer) ESP et n'inclut ni l'en-tête IP ni le champ d'authentification.

- Le SPI (Security Parameters Index) permet de caractériser l'association de sécurité utilisée pour la communication (SA).
- Les données d'authentification contiennent la valeur de vérification d'intégrité (ICV) permettant de vérifier l'authenticité des données du paquet.
- Le numéro de séquence pour éviter le rejeu.
- Les données chiffrées sont contenues dans la partie « champ libre » (ou PayLoad Data) du paquet.
- Le champ En-tête suivant (Next Header) indique la nature des informations contenues dans le Payload Data (champ libre).

5.7. Le protocole IKE

5.7.1. Principe [33][34][35][36]

On a décrit précédemment une Security Association, elle peut être configurée manuellement par un administrateur dans la base SADB, mais en général elle est positionnée de façon automatique. En effet dans le cadre d'un VPN, le tunnel sera stable dans la durée, l'administrateur peut positionner une fois pour toute une SA. Mais dans le cadre d'une utilisation d'un nomade, par exemple, il faut que cette tâche soit automatisée. Il existe donc un démon appelé démon IKE (Internet Key Exchange) qui se charge de la négociation et de la mise en place des SA. IKE se décompose en plusieurs protocoles :

- ISAKMP (Internet Security Association and Key Management Protocol) : il négocie les "Security Associations" (établissement, supression, paramètres...). ISAKMP n'est pas spécifique à IPsec, il existe donc un DOI (Domain Of Interpretation) qui permet d'utiliser ce qui est produit par ISAKMP dans IPsec.
- Oakley et Scheme sont des protocoles d'échange de clé et d'authentifications.

Le principe est que si un paquet est envoyé par la machine, le système vérifie dans la SPD quelle est la politique IPsec à lui associer. Si IPsec doit ou peut se charger de ce paquet, le système va rechercher une SA qui correspond à cette SP et à ce paquet. Si le système trouve une SA correspondante, le paquet se voit appliquer les règles définies dans la SA, sinon le système va appeler le démon IKE pour créer si possible une SA.

A noter qu'IKE effectue ses échanges au-dessus du niveau transport (UDP, port 500, en général). Cela permet bien de découpler la négociation IPsec des fonctionnalités d'IPsec. Voire figure 5.1

Le protocole ISAKMP a été conçu pour être souple et pourrait être utilisé par d'autres applications qu'IPsec. La structure des paquets ISAKMP est constituée de chaînage de blocs. Il existe 13 types de blocs possibles :

Nom	Sigle
Security Association	SA
Proposal	P
Transform	T
Key Exchange	KE
Identification	ID
Certificate	CERT
Certificate Request	CR
Hash	HASH
Signature	SIG
Nonce	NONCE
Notification	N
Delete	D

Vendor ID	VID

Tableau 5.1.Bloc d'ISAKMP[33][34][35][36]

- Le bloc **Security Association** indique le contexte et la situation de l'échange. Il comporte notamment le DOI qui précise si on est dans un échange de type ISAKMP (première phase) ou dans l'étape plus spécifique d'IPsec (deuxième phase). Ce bloc est composé d'un ou plusieurs blocs Proposal.
- Le bloc **Proposal** contient des propositions sur les mécanismes de sécurité à fournir tel que les ESP avec les SPIs correspondants. Ce bloc est composé d'un ou plusieurs blocs **Transform**.
- Le bloc **Transform** contient les algorithmes et les attributs associés pour les mécanismes de sécurité contenus dans les blocs **Proposal**.
- Le bloc **Key Exchange** contient des données nécessaires à la génération de clés. Ces données dépendent du contexte et du protocole d'échange de clés.
- Le bloc **Identification** contient des données nécessaires à l'identification des entités présentes. Ces données dépendent du contexte et du procédé d'identification. Plusieurs types sont possibles : adresses IP (v4 ou v6), voire plage d'adresses IP, mais aussi FQDN (Fully Qualified Domain Name) ou USER FQDN. Son interprétation est différente en phase 1 ou en phase 2. En phase 1 ce bloc sert à l'authentification, tandis qu'en phase 2 cela sert de **traffic selector**.
- Le bloc **Certificate** contient un certificat ainsi que le type de certificat contenu. Ce certificat peut être de différents types: PKCS#7, DNS signed KEY, X509 (signature key exchange ou attribute) ou SPKI certificate.
- Le bloc **Certificate Request** permet de réclamer un certificat à l'entité tiers (en précisant le type de certificat voulu).
- Le bloc **Hash** contient le résultat d'une fonction de hachage sur le message ou sur un attribut.
- Le bloc **Signature** contient le résultat d'un mécanisme de signature sur le message ou sur un attribut. Cela est souvent utilisé pour faire de l'authentification.
- Le bloc **Nonce** contient des valeurs aléatoires choisies par une des entités.
- Le bloc **Notification** contient des messages d'erreur ou d'information. Son interprétation dépend du contexte.
- Le bloc **Delete** permet à une entité de préciser à son homologue qu'elle vient de supprimer une SA.
- Le bloc **Vendor ID** comme il est spécifié dans les RFC peut-être utilisé à des fins spécifiques à une implémentation. Mais en général, il sert à négocier des extensions.

5.7.2. Les phases du protocole

IKE[37] [38] :

Phase 1 :

Deux modes sont possibles pour la première phase, le mode "main" et mode "aggressive".

Nous détaillerons le mode main.

Cette phase va servir à la création d'une première clé qui va permettre par la suite la génération de 3 autres clés dérivant de celle-ci. Cette clé peut être générée selon 3 modes offerts par IKE.

- Le mode ***«secret partagé»*** implique que les hôtes partagent déjà un secret qui permettra la mise au point de cette clé.
- Le mode « ***Chiffrement asymétrique »*** se base sur les crypto système à clé publique pour échanger les données sensibles et donc établir le secret partagé.
- Le mode «Signature», quant à lui, se sert du chiffrement asymétrique pour signer et authentifier les hôtes alors que le secret partagé est établi grâce à Diffie-Hellman.

Une fois la première clé générée, elle est dérivée en 3 autres clés qui serviront à la création du tunnel IKE sécurisé entre les hôtes (en faîte une association de sécurité ISAKMP). L'une des clés sera utilisée pour l'authentification, l'autre pour le chiffrement et la dernière sera utilisé lors de la phase 2 du protocole. Ce canal, sécurisé, est ensuite utilisé pour la deuxième phase IKE. Plus précisément, lors de cette phase, les échanges permettent de définir l'association de sécurité puis d'établir le secret partagé et enfin d'authentifier les hôtes.

Il faut noter que le mode « aggressive » permet de limiter les communications en utilisant certains paramètres d'office. D'autre part, les SA ISAKMP utilisent un chiffrement (DES ou 3DES) lors de l'échange des clefs de session.

Phase 2 :

L'objectif de la deuxième phase est de créer les tunnels IPSec (SA) pour les échanges effectifs entre les hôtes. Deux SA par hôtes, un pour chaque sens de communication, conservées dans la SAD (Security Association Database). C'est lors de cette phase que chaque hôte donne ses préférences en matière d'algorithme et établissent le matériel cryptographique. Les clés de session sont générées à partir de l'une des clés dérivées, générée durant la phase 1 d'IKE.

Cependant, lorsque le mode «Perfect Secrecy» est utilisé, les hôtes doivent échanger de nouveaux secrets, ceci afin de couper la relation systématique entre les nouvelles clés générés et la clé de la phase 1. Cet échange s'effectue via le protocole d'échange Diffie-Hellman. Cette phase sert aussi à spécifier les échanges devant bénéficier des services IPSec (utilisation de la Security Policy Database),

5.7.3. Les caractéristiques communes du protocole IKE [37][34]

- Les composantes publiques DH générées par l'initiateur (noté g^{Xi}) et le répondeur (noté g^{Xr}) seront mis dans les charges utiles d'échange de clé (KE).
- Le calcul des données d'authentification [AUTH] dépend de la méthode d'authentification utilisée. Mais, quelle que soit la méthode d'authentification, les données d'authentification sont toujours calculées dans les valeurs de hachage d'information suivantes:

  - $HASH_I = prf(SKEYID, g^{Xi} \| g^{Xr} \| cookie\text{-}I \| cookie\text{-}R \| R \| SA \| ID_I)$. $HASH_I$ est envoyé par l'initiateur;

- $HASH_R$ = prf(SKEYID, g^{Xr} | g^{Xi} |cookie-R|cookie-I|R|SA|ID_R). $HASH_R$ est envoyé par le répondeur;

  - Le symbole "|" signifie la concaténation;
  - « SA » dans $HASH_I$ et $HASH_R$ est la charge utile SA envoyée par l'initiateur;
  - « Prf » est une fonction pseudo-aléatoire habituellement mise en œuvre par une clé-hash telle que HMAC; la transformation mathématique exacte est déterminée lors de la négociation des paramètres;
  - SKEYID est la clé de prf, Elle est différente pour chaque méthode d'authentification.

La sortie finale de la première phase du protocole IKE est une SA ISAKMP avec trois clés secrètes partagées exclusivement entre l'initiateur et le répondeur. Les trois clés sont:

- $SKEYID_d = prf$(SKEYID, $g^{Xi\ Xr}$ ‖cookie-I‖cookie-R‖0). La clé $SKEYID_d$ est utilisée pour dériver d'autres clés dans les phases I et II du protocole IKE;
- $SKEYID_a = prf(SKEYID, SKEYID_d$ ‖$g^{Xi\ Xr}$ ‖cookie-I‖cookie-R‖1). La clé $SKEYID_a$ est utilisée pour authentifier des messages de la deuxième phase de protocole IKE;
- $SKEYID_e = prf(SKEYID, SKEYID_a$ |$g^{Xi\ Xr}$ |cookie-I|cookie-R|2). La clé $SKEYID_e$ est utilisée pour chiffrer les messages 5 et 6 en mode principal et tous les messages de la Phase II du protocole IKE;

Le secret partagé DH, $g^{Xi\ Xr}$, est la principale source d'entropie (aléatoire) pour dériver ces trois clés.

IKE définit ses propres paramètres lors de la négociation des paramètres. Ces paramètres incluent les méthodes d'authentification, les algorithmes de hachage, les algorithmes de chiffrement, les fonctions pseudo-aléatoires, et les groupes algébriques Diffie-Hellman.

Parmi les méthodes d'authentification utilisées, on trouve la méthode de clé pré-partagée qui représente la forme de base. Les trois autres peuvent être considérées comme des variantes de la méthode clé pré-partagée

5.7.4. Le protocole IKE utilisant une clé pré-partagé

La figure 5.5 représente le mode principal du protocole IKE utilisant une clé pré-partagée. Une clé secrète doit être partagée exclusivement entre l'initiateur et le répondeur avant que la négociation IKE ne s'effectue. AUTH sont remplacés par $HASH_I$ et $Hash_R$. La clé SKEYID est dérivée de la clé pré-partagée [37]:

SKEYID = prf (preshared-key, $NONCE_I$|$NONCE_R$)

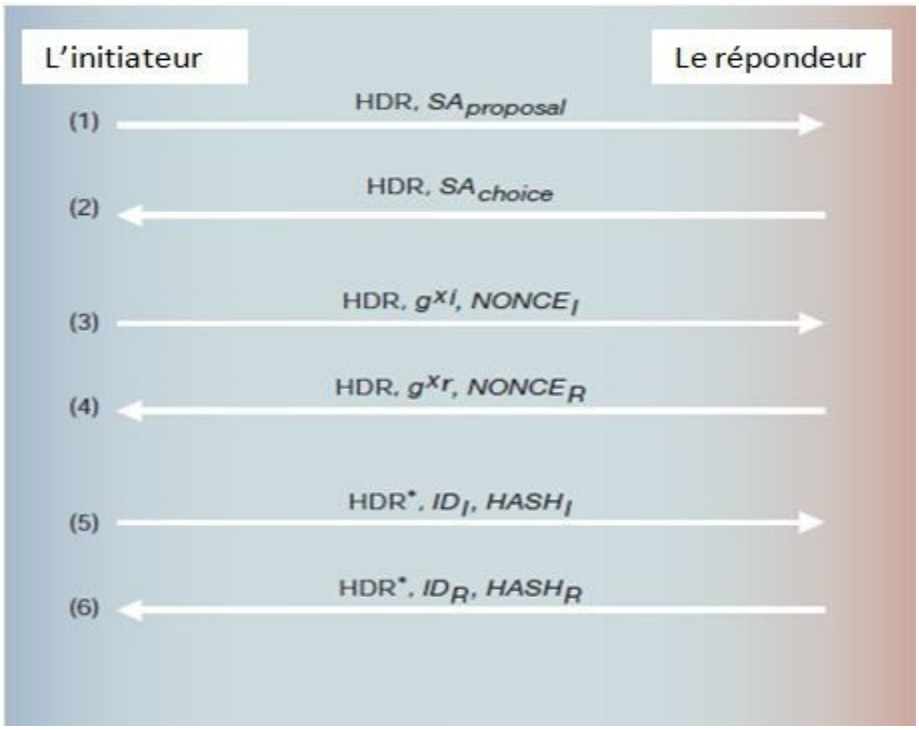


Figure 5.5. Le mode principal du protocole IKE utilisant une clé pré-partagée [37]

- **Le mode agressif**

Figure 5.6. Le mode agressif du protocole IKE utilisant une clé pré-partagée[37]

La figure 5.6 représente le mode agressif du protocole IKE utilisant la méthode d'authentification à clé pré-partagée.

Les caractéristiques de ce mode sont :

Les identités ne sont pas cryptées, sauf dans la méthode de chiffrement à clé publique ou révisée;

- ❖ Le groupe algébrique Diffie-Hellman ne peut être négocié. Il est choisi par l'initiateur;

- ❖ La méthode d'authentification ne peut être négociée dans le cas où l'initiateur choisit la méthode de chiffrement à clé publique ou le chiffrement à clé publique révisée. Mais, il peut offrir au répondeur un choix lors de l'utilisation de méthode à clé pré-partagée ou une signature à clé publique.

- **Le mode rapide**

Comme son nom l'indique, le mode rapide est proposé pour être rapide et efficace. Une négociation de mode rapide se compose de trois messages et il peut se conduire sans l'utilisation de toutes les opérations de cryptographie à clé publique.

Le mode rapide fournit la possibilité d'utiliser des techniques D-H, de sorte que g^{Xi} et g^{Xr} illustrés dans la figure suivante soient facultatifs. En outre, les identités ID_{Ui} et ID_{Ur} sont facultatives; si les identités ne sont pas envoyées, il suppose que les identités non envoyées sont les mêmes que l'ID_I et ID_R dans le SA ISAKMP. De plus, si l'initiateur envoie g^{Xi} ou (ID_{Ui} , ID_{Ur}) dans le premier message, le répondeur doit envoyer g^{Xr} (ID_{Ui} et ID_{Ur}) dans le second message pour continuer la négociation.

Dans la figure 5.7, le symbole « * » après les en-têtes des messages indiquent que les corps de ces messages sont cryptés par la clé $SKEYID_e$ et l'algorithme de chiffrement utilisé se trouve dans SA ISAKMP. $HASH_1$, $HASH_2$ et $HASH_3$ authentifient les messages correspondants. Ils sont calculés par la clé $SKEYID_a$ et la fonction pseudo-aléatoire dans la SA ISAKMP:

$HASH_1 = prf\ (SKEYID_a,\ message\text{-}ID|SA|NONCE_I\ [|g^{Xi}]\ \ [|ID_{Ui}|ID_{Ur}])$

$HASH_2 = prf\ (SKEYID_a,\ message\text{-}ID|SA|NONCE_R\ [|g^{Xr}]\ \ [|ID_{Ui}|ID_{Ur}])$

$HASH_3 = prf(SKEYID_a, 0|message\text{-}ID|NONCE_I|NONCE_R)$

Les informations entre crochets ([]) sont facultatives lors du calcul des valeurs de hachage; ils sont utilisés si et seulement si les messages contiennent ses informations.

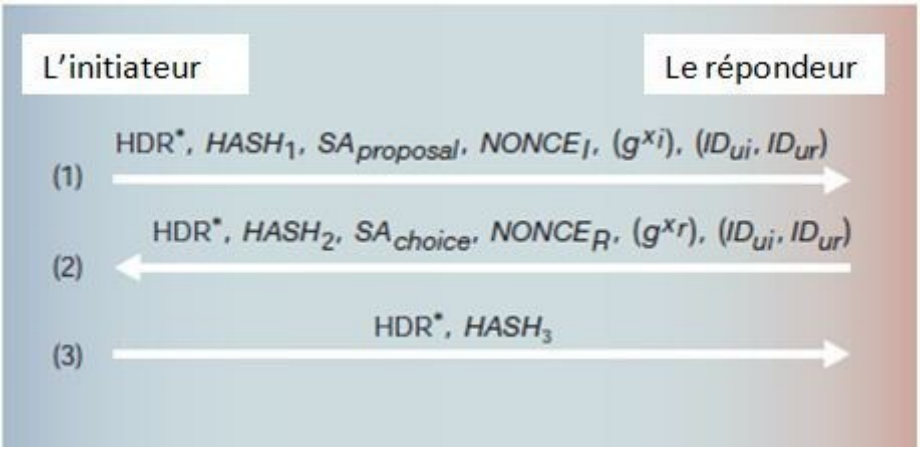


Figure 5.1 Le mode rapide du protocole IKE[37]

Les propositions contenues dans la charge utile SA sont pour la sécurité du protocole IPSec. Les paramètres de sécurité de l'IPSec sont générés pour chaque protocole de sécurité et ils sont choisis par le répondeur. La clé du protocole, appelé KEYMAT, est dérivée comme suit :

$KEYMAT = prf(SKEYID_d, [g^{XiXr}]protocol\|SPI\|NONCE_I\|NONCE_R)$

TP3

Création d'un tunnel IPSEC sécurisé entre deux sites

(rapport détaillé+ une validation)

Objectif du TP

Le TP à pour objectif de mettre en œuvre un VPN sécurisé entre deux réseaux locaux en

Utilisant le protocole IPSEC.

IP Sec sera paramétré en mode de gestion automatique (avec un secret pré- partagé) des clefs basé sur le protocole ISAKMP.

Vous avez le choix soit de faire une configuration sous linux ou sous Windows

Corrigé TP3

Création d'un tunnel IPSEC sécurisé entre deux sites

Partie 1 : Configuration du pfsense sur VirtualBox

Dans l'objectif de mettre en œuvre un vpn sécurisé entre deux réseaux locaux, nous avons installé deux pfsenses, chaque pfsense avec deux interfaces réseaux, l'une accès par pont (interface WAN) et l'autre réseau interne (LAN). Chaque interface LAN est connectée à un réseau local également sur VM.

Pare-feu 1 : nous avons configuré ce 1er pfsense comme suit :

Adresse WAN : est attribuer par le point d'accès : v4 DHCP ->172.20.10.7

Adresse LAN :192.168.1.1, qui est aussi la passerelle du réseau interne

```
*** Welcome to pfSense 2.5.2-RELEASE (amd64) on pfSense ***

 WAN (wan)       -> em0        -> v4/DHCP4: 172.20.10.7/28
 LAN (lan)       -> em1        -> v4: 192.168.1.1/24

 0) Logout (SSH only)                  9) pfTop
 1) Assign Interfaces                 10) Filter Logs
 2) Set interface(s) IP address       11) Restart webConfigurator
 3) Reset webConfigurator password    12) PHP shell + pfSense tools
 4) Reset to factory defaults         13) Update from console
 5) Reboot system                     14) Enable Secure Shell (sshd)
 6) Halt system                       15) Restore recent configuration
 7) Ping host                         16) Restart PHP-FPM
 8) Shell
```

Figure 1

Pare-feu 2 :nous l'avons configuré manuellement comme suit :

Adresse WAN : est attribuer par le point d'accès : v4 DHCP ->172.20.10.4

Adresse LAN :192.168.0.1, qui est aussi la passerelle du réseau privé

```
*** Welcome to pfSense 2.5.2-RELEASE (amd64) on pfSense ***

 WAN (wan)       -> em0        -> v4/DHCP4: 172.20.10.4/28
 LAN (lan)       -> em1        -> v4: 192.168.0.1/24

 0) Logout (SSH only)                  9) pfTop
 1) Assign Interfaces                 10) Filter Logs
 2) Set interface(s) IP address       11) Restart webConfigurator
 3) Reset webConfigurator password    12) PHP shell + pfSense tools
 4) Reset to factory defaults         13) Update from console
 5) Reboot system                     14) Enable Secure Shell (sshd)
 6) Halt system                       15) Restore recent configuration
 7) Ping host                         16) Restart PHP-FPM
 8) Shell
```

Figure 2

Partie 2 :La suite de configuration de l'interface pfsense via le navigateur de la machine interne

Pare-feu 1 : tout d'abord, faut créer un tunnel VPN IP sec qui comporte deux phases.

1ère phase: la négociation de paramètre de sécurité ;

2 eme phase : relative à la sécurité des données ;

Figure 3

IP sec : ici nous avons mis une règle qui autorise chaque trafic depuis au réseau LAN

Figure 4

WAN :

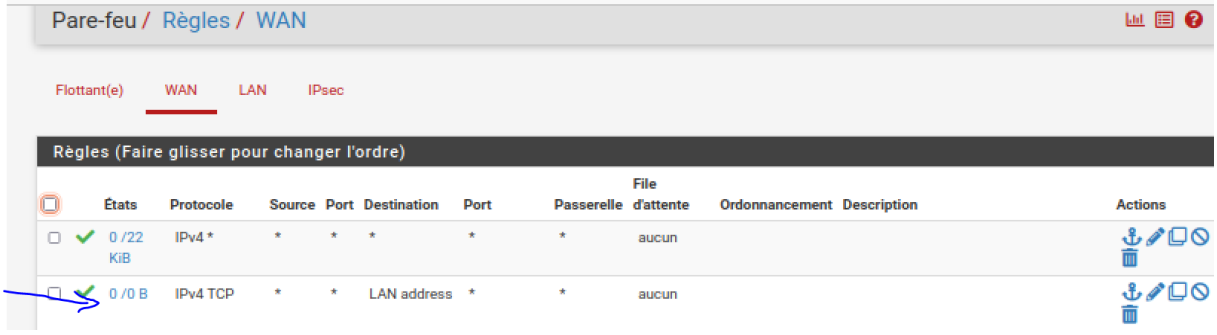


Figure 5

Routage :

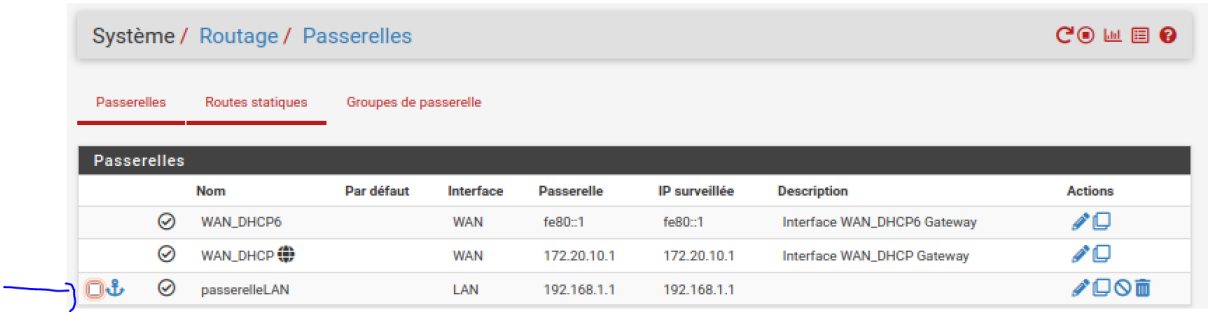


Figure 6

IP sec vue : Etablissement de la connexion avec le tunnel vpn sec

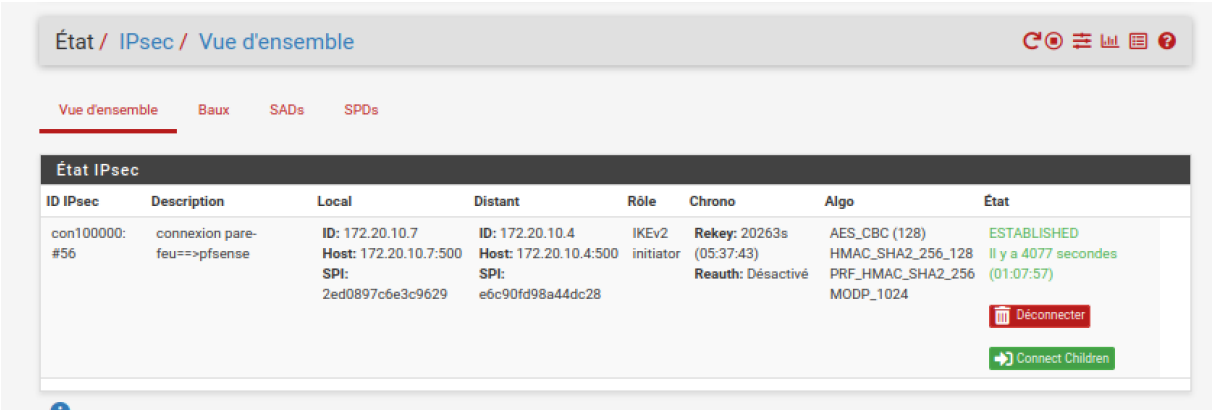


Figure 7

Pare-feu 2 : tunnel VPN IP sec qui comporte deux phases.

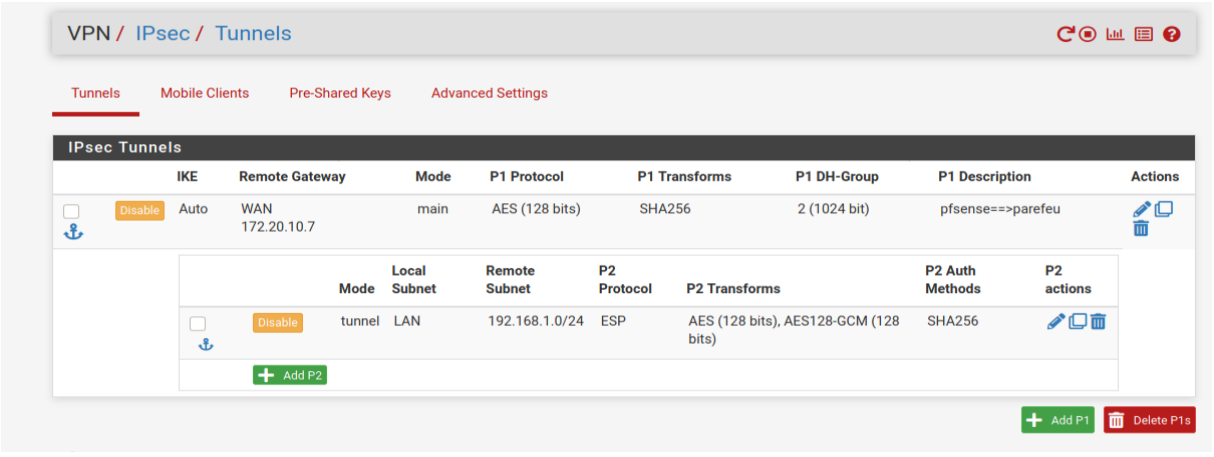


Figure 8

IP sec : ici nous avons mis une règle qui autorise chaque trafic depuis un réseau LAN extérieur ;

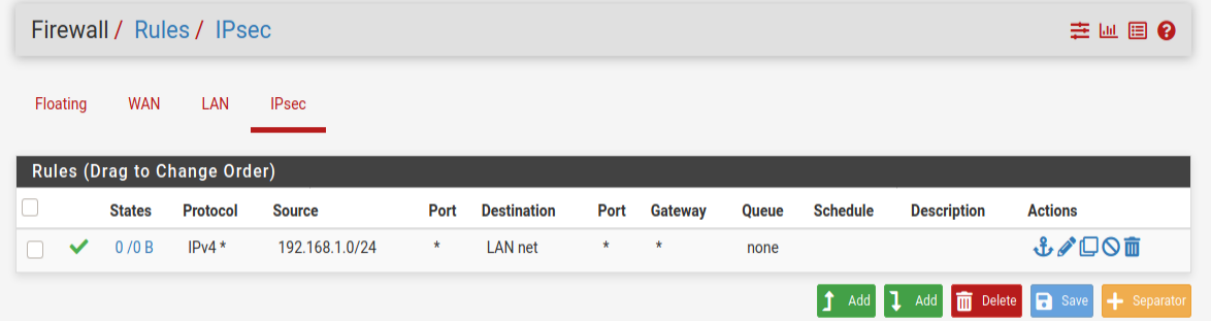


Figure 9

Routage :

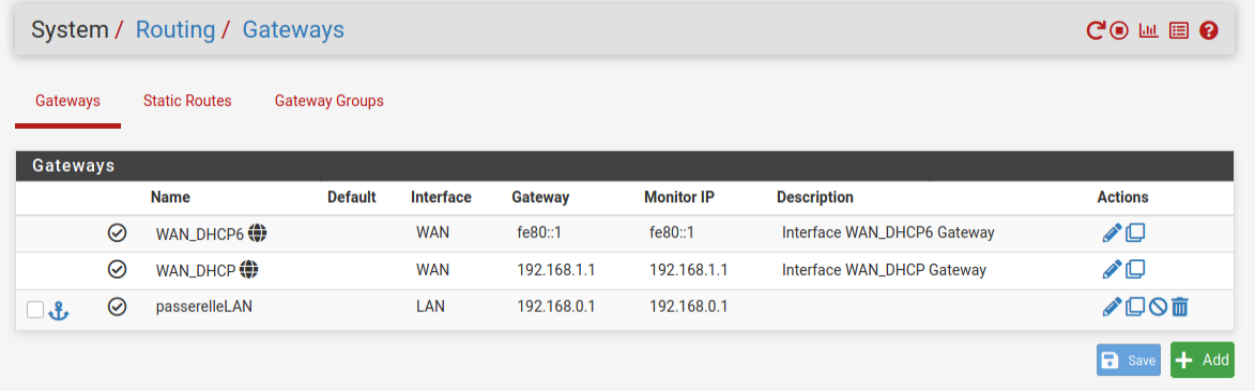


Figure 10

IP sec vue : Etablissement de la connexion avec le tunnel vpn sec ;

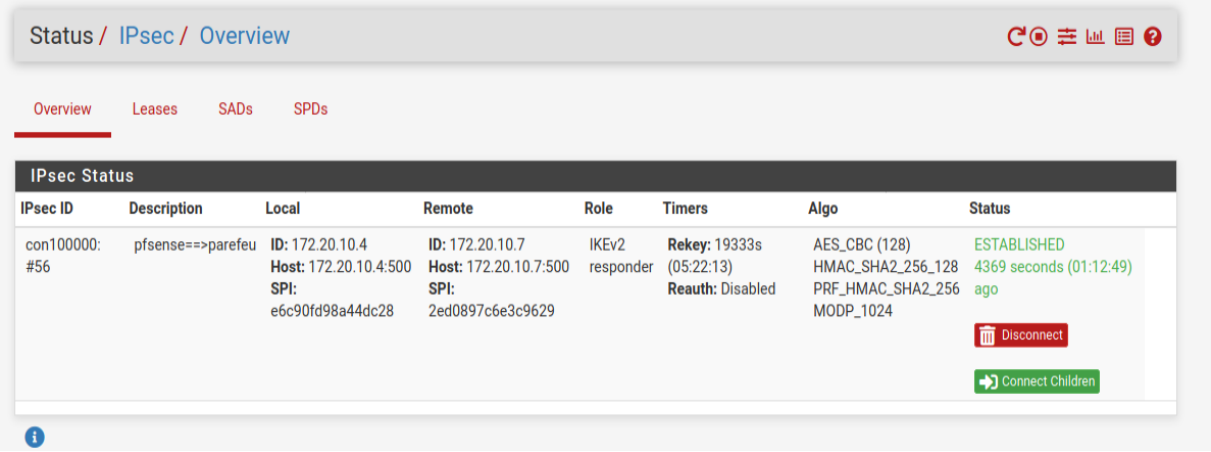


Figure 11

<u>Partie 3 : Test sur VM kali linux vers la page par défaut éditer sur l'autre VM Ubuntu</u>.

La page sur kali : Ad :192.168.1.37

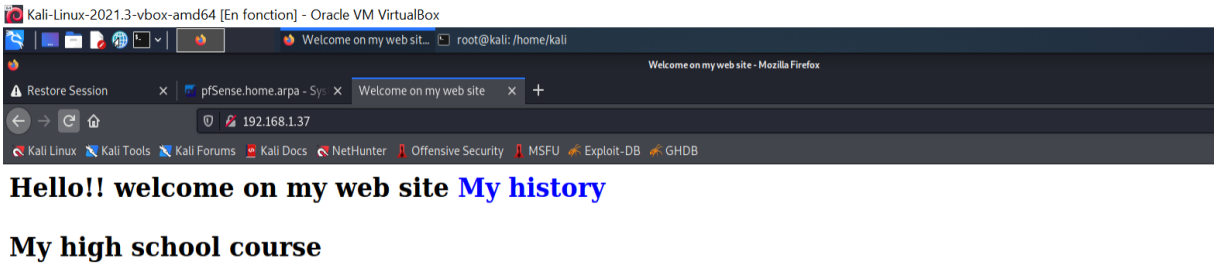


Figure 12

La page sur Ubuntu : Ad :192.168.1.37

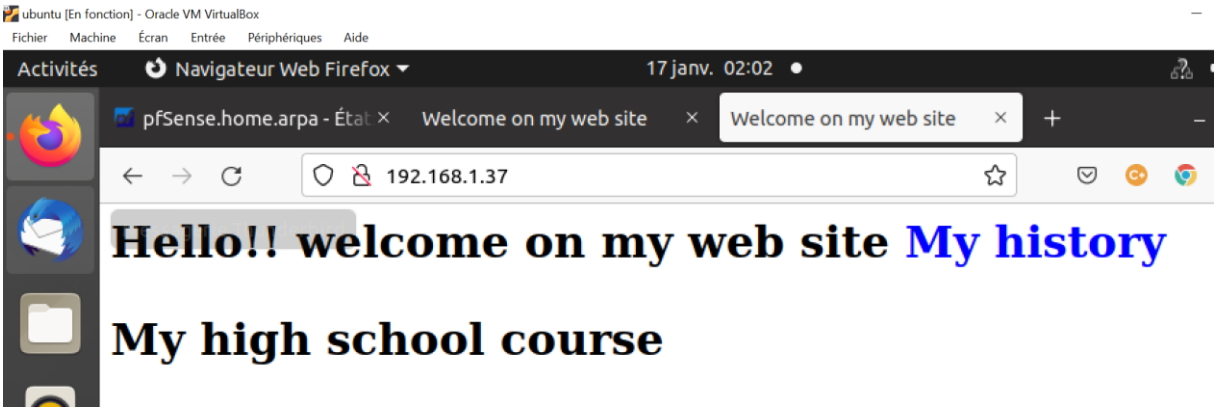


Figure 13

Teste des pings de kali vers Ubuntu et Ubuntu vers kali

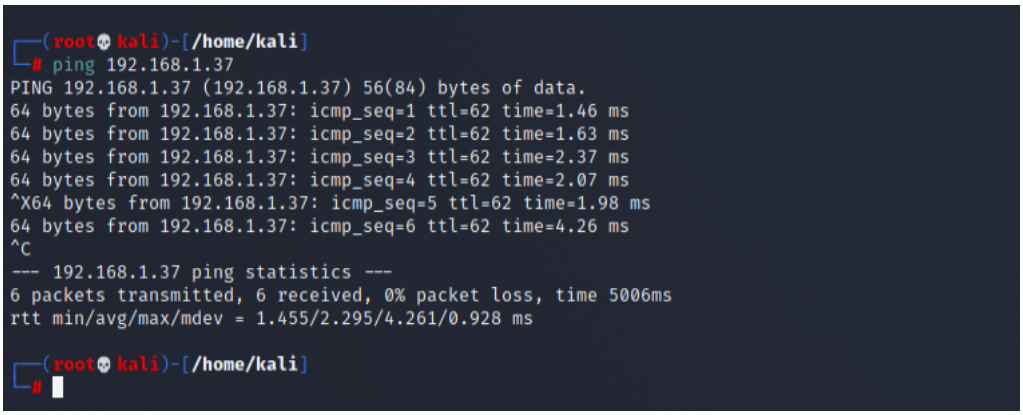


Figure 14

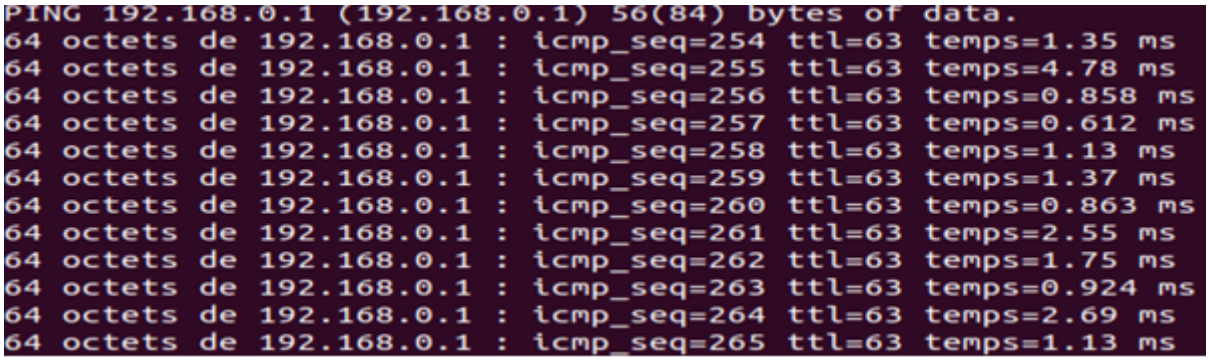

PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 octets de 192.168.0.1 : icmp_seq=254 ttl=63 temps=1.35 ms
64 octets de 192.168.0.1 : icmp_seq=255 ttl=63 temps=4.78 ms
64 octets de 192.168.0.1 : icmp_seq=256 ttl=63 temps=0.858 ms
64 octets de 192.168.0.1 : icmp_seq=257 ttl=63 temps=0.612 ms
64 octets de 192.168.0.1 : icmp_seq=258 ttl=63 temps=1.13 ms
64 octets de 192.168.0.1 : icmp_seq=259 ttl=63 temps=1.37 ms
64 octets de 192.168.0.1 : icmp_seq=260 ttl=63 temps=0.863 ms
64 octets de 192.168.0.1 : icmp_seq=261 ttl=63 temps=2.55 ms
64 octets de 192.168.0.1 : icmp_seq=262 ttl=63 temps=1.75 ms
64 octets de 192.168.0.1 : icmp_seq=263 ttl=63 temps=0.924 ms
64 octets de 192.168.0.1 : icmp_seq=264 ttl=63 temps=2.69 ms
64 octets de 192.168.0.1 : icmp_seq=265 ttl=63 temps=1.13 ms

Figure 15

CHAPITRE 6

LE CRC DANS LES PROTOCOLES

Chapitre 6

Le CRC dans les protocoles

6.1. Introduction

Beaucoup de système, notamment informatique, utilisent le transfert de données. Par exemple, un réseau informatique : il s'agit d'envoyer des données d'un point A à un point B. Celles ci sont transférées par « paquet » de plusieurs bits qui renferment l'adresse de l'émetteur, l'adresse du destinataire et bien sûr, les données elles mêmes. Mais comment s'assurer, une fois réceptionnées, que ces données sont conformes à celles qui ont été envoyées ? Plusieurs solutions existent. La première et la plus simple serait de ré envoyer la trame de données, cependant, cela est bien trop coûteux en taille. D'autres méthodes consistent à rajouter quelques bits de vérification à la trame. Si ces bits sont le résultat d'une fonction mathématique appliqué aux données, il suffit d'appliquer une fonction inverse à la réception pour déterminer si oui ou non la trame est bien la même que celle envoyé. C'est cette méthode qui nous intéresse ici. Une technique simple est d'utiliser un bit de parité (1 si la trame est paire, 0 si elle est impaire); voilà une méthode peu coûteuse en place mais pas forcément très sûre. De manière un peu plus efficace on peut se rajouter un « cheksum » qui n'est autre que la somme des bits constituant de la trame de données. C'est une solution plus coûteuse mais qui permet de détecter déjà beaucoup d'erreurs. Nous arrivons maintenant à notre méthode qui consiste à calculer un code de redondance cyclique à partir de la trame des données. Ce code se détermine par des calculs relativement simples basés sur l'arithmétique modulaire. Nous verrons plus tard que la qualité de ses résultats est fonction d'un polynôme dit « générateur ». Selon ce dernier, on peut obtenir des probabilités d'intégrité de la trame proche des 100%.

6.2.Principe[39]

Le principe du *CRC* consiste à traiter les séquences binaires comme des polynômes binaires, c'est-à-dire des polynômes dont les coefficients correspondent à la séquence binaire. Ainsi la séquence binaire *0110101001* peut être représentée sous la forme polynomiale suivante :

$0*X^9 + 1*X^8 + 1*X^7 + 0*X^6 + 1*X^5 + 0*X^4 + 1*X^3 + 0*X^2 + 0*X^1 + 1*X^0$
soit :

$X^8 + X^7 + X^5 + X^3 + X^0$ ou encore : $X^8 + X^7 + X^5 + X^3 + 1$

De cette façon, le bit de poids faible de la séquence (le bit le plus à droite) représente le degré 0 du polynôme ($X^0 = 1$), le $4^{ème}$ bit en partant de la droite représente le degré 3 du polynôme (X^3).etc.

Une séquence de *n* bits constitue donc un polynôme de degré maximal *n-1*. Toutes les expressions polynomiales sont manipulées par la suite avec une arithmétique modulo 2.

Dans ce mécanisme de CRC, un polynôme prédéfini (appelé *polynôme générateur* et noté G(X)) est connu de l'émetteur et du récepteur. La détection d'erreur consiste pour l'émetteur à effectuer un algorithme sur les bits de la trame afin de générer un CRC, et de transmettre ces deux éléments au récepteur. Il suffit alors au récepteur d'effectuer le même calcul afin de vérifier que le CRC est valide [39].

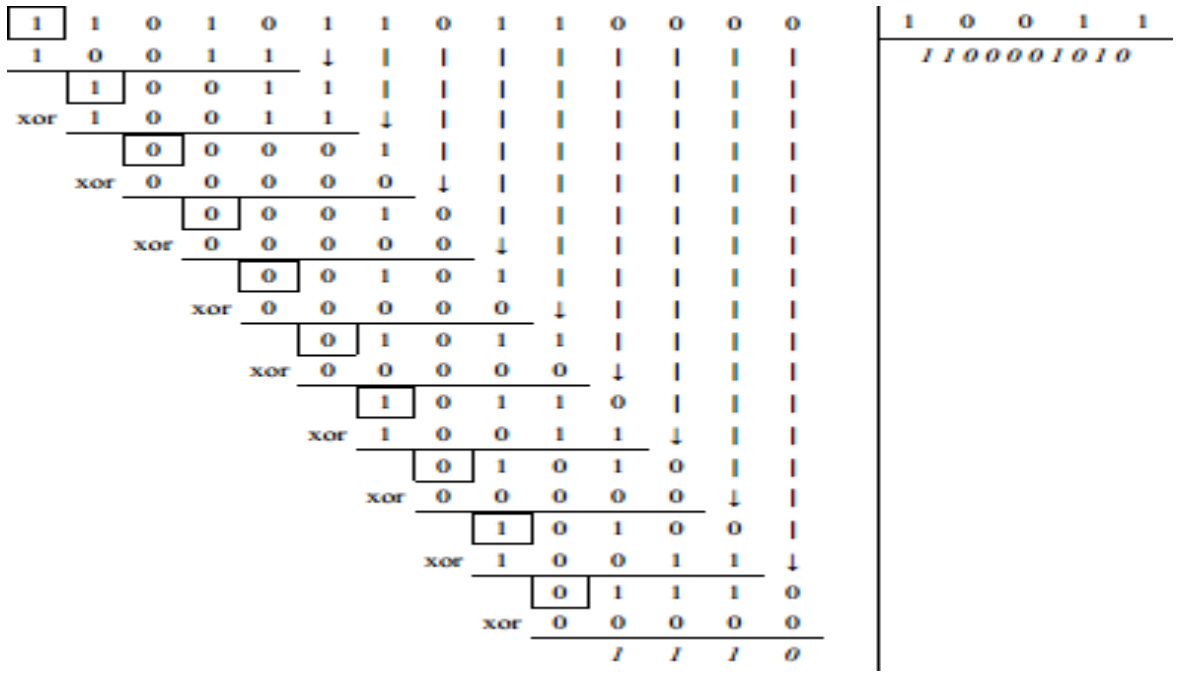
6.3. Rappel sur l'arithmétique sur les nombres binaires

6.3.1. L'opérateur OU exclusif (XOR) [40]

Cet opérateur est très utilisé dans l'algorithme de création du CRC, en voici ces caractéristiques:

A	B	A xor B
0	0	0
0	1	1
1	0	1
1	1	0

6.3.2. La division entre 2 nombre binaires [41]



Pour ce faire, on sélectionne le bit de poids fort du dividende, ici il vaut 1, il sera le bit de poids fort du quotient. Maintenant, on peut faire (1 x diviseur) = 1 0011. On réalise ensuite un XOR entre le dividende et ce résultat qui est décalé de suffisamment de bit pour commencer au début du dividende. Ensuite, on ajoute au résultat de l'opération le bit suivant correspondant au dividende (on le fait « descendre »). L'opération s'enchaîne ainsi de suite. On sélectionne le bit de poids fort du résultat précédemment trouvé, ici, c'est un 1. On lui

applique donc un XOR avec 1 0011. Le résultat sera 0000 auquel on fera descendre 1, ce qui donne 00001. Ici, le bit de poids fort est un 0, donc (0 x diviseur) = 00000. On refait l'opération avec le XOR.

Ces opérations seront réalisées jusqu'à ce qu'on ne puisse plus « descendre » de bit depuis le dividende, autrement dit, on réalise (taille_du_dividende – taille_du_diviseur + 1) opérations (la taille correspondant au nombre de bits).

6.4. Fonctionnement [39]

Le calcul va utiliser le principe de la division polynomiale. Ainsi, on va diviser la trame d'entrée A par un polynôme et donc récupérer le CRC. A l'arrivée on divisera la trame B complète (A+CRC) par ce même polynôme. Si le reste est nul, c'est que la trame de départ est la même que la trame d'arrivée.

Le choix du polynôme générateur est fonction de la qualité du résultat du CRC. Sans rentrer dans les détails mathématiques, le plus simple est d'utiliser ceux qui sont définie comme étant de bonne qualité. Les CRC-16 et 32 donne une fiabilité de près de 100% au niveau de l'intégrité de la trame. Les polynômes générateurs les plus connus sont :

Nom	Polynôme générateur
CRC-4	x4 + x2 + x1
CRC-12	x12 + x11 + x3 + x2 + x1 + 1
CRC-16 SDLC (CCITT)	x16 + x12 + x5 + 1
CRC-16	x16 + x15 + x2 + 1
CRC-16 Reverse	x16 + x14 + x1 + 1
SDLC Reverse	x16 + x11 + x4 + 1
CRC-32 (ethernet)	x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x1 + 1

Tableaux 6.1 : Polynômes générateurs
[39]

L'algorithme de base qui renvoi la trame d'origine suivit de son CRC est le suivant :

```
Début
   Décaler, vers la gauche, du degré du polynôme générateur le mot d'entrée
   Faire la division (mot d'entrée / polynôme générateur)
   Faire le calcul (mot d'entrée XOR polynôme générateur)
Fin
```

L'algorithme qu'il faut appliquer à une trame reçu (contenant un CRC) est le suivant :

```
Début
  Faire la division (trame reçu / polynôme générateur)
     Si le reste est nul Alors la trame est correcte
     Sinon re-transférer la trame
Fin
```

6.5. Exemple pratique [39]

Pour cet exemple on va réutiliser l'exemple de la division vu dans la section 6.3.2.

On veut transférer le mot : 11 0101 1011. On décide d'utiliser le polynôme générateur de degré 4 (le degré étant le nombre de bit – 1); ici 10011.

D'abord on décale 11 0101 1011 de 4 rang vers la gauche, résultat : 11 0101 1011 0000.
Ensuite, on cherche le reste de la division de 11 0101 1011 0000 par 10011, résultat : 1110 (c'est le FCS pour Frame Control Sequence, nommé par abus de langage CRC).

Enfin on applique un XOR entre le mot d'entrée (décalé de 4 rang) et le reste de la division, 11 0101 1011 0000 XOR 1110 = 11 0101 1011 1110, ce qui correspond bien au mot de départ « suivi » du CRC.

On transfert cette trame. Si on la récupère telle quelle, on obtient 11 0101 1011 1110. On divise ceci par 10011, et on trouve un reste nul, le transfert s'est donc correctement déroulé, aucune perte de données n'est à déplorer

Bibliographie

[1] Baudet, M. ―Sécurité des protocoles cryptographiques: aspects logiques et calculatoires‖, l'École Normale Supérieure de Cachan, (2007) [online] https://tel.archives-ouvertes.fr/tel-00140916

[2] Les mécanismes de chiffrement Chapter 23. Le chiffrement [online] http://www.linux-france.org/prj/edu/archinet/systeme/ch23s02.html#chif_sym

[3] Guillot, P. (2013) ―La cryptologie : L'art des codes secrets ‖, EDP sciences, France, ISBN : 978-2-7598-0811-3.

[4] Public Key Infrastructure (PKI), Les concepts de base de la cryptographie [Online] http://igm.univ-mlv.fr/~dr/XPOSE2007/vma_PKI/concepts_de_base.html

[5]C.Cachet D.Carella. ― PKI Open source . Edition O'Reilly, Paris,(2003),ISBN 2-84177-235-7

[6] Jean-Luc Archimbaud, LES IGC, ―INFRASTRUCTURES DE GESTION DE CLÉS, Lavoisier | « Les Cahiers du numérique » 2003/3 Vol. 4 | pages 111 à 134 ISSN 1622-1494 ISBN 2746209071.

[7] Introduction aux concepts d'infrastructures à clés publiques [online] SecurIT@free.fr.

[8] Autorités de Certification et Hiérarchies de Confiance [online]

https://www.globalsign.com/fr/centre-information-ssl/autorite-de-certification-hierarchies-de-confiance

[9] Autorité de certification privée AWS Certificate Manager Guide de l'utilisateur Version latest,[online] https://docs.aws.amazon.com/fr_fr/acm-pca/latest/userguide/pca-ug.pdf#ca-hierarchy

[10] Qu'est-ce qu'une AC subordonnée et pourquoi vouloir la vôtre, [online]https://www.globalsign.com/fr/blog/qu-est-ce-qu-une-ac-subordonnee-ou-intermediaire

[11] R.ousley, W.Ford, W.Polk, D.Solo. ―RFC 2459- Internet X.509 Public Key Infrastructure Certificate and CRL Profile", January (1999).

[12] S.Thomas. ―SSL and TLS Essential securing the Web , Wiley Computer Publishing, USA, February (2000), ISBN 0-471-38354-6

[13]. Présentation du concept d'annuaire LDAP, [online]

https://openclassrooms.com/fr/courses/2257706-presentation-du-concept-dannuaire-ldap,

[14]. ZIE FOMEKONG Dany Stéphane, Les services d'annuaires LDAP : Application au référencement dans les transports terrestres Camerounais, mémoire de Master en Informatique Approfondie, Ecole supérieure de commerce et de gestion ,ESIG- SIANTOU, Année académique 2005/2006

[15]. Lightweight Directory Access Protocol [online] https://ldap.com/

[16]. - L. Mirtain, Tutorial LDAP, LORIA-INRIA, [online] http://wwwsop.inria.fr/members/Laurent.Mirtain/LDAP.html

[17] Dominique Colombani, ― LDAP maitrise du protocole exploitation d'un service d'annuaire (OpenLDAP, Active Directory), Edition ENI, decembre (2006), Isbn 10 : 2-7460-3423-9

Isbn : 13 :978-2-7460-3423-5, issn : 1627-8224.

[18]. T. Dierks et C. Allen, ―The TLS Protocol Version 1.0 , . RFC 2246 (Proposed Standard), jan. (1999). Obsoleted by RFC 4346, updated by RFCs 3546, 5746, 6176, 7465, 7507

[19]. T. Dierks et E. Rescorla, ―The Transport Layer Security (TLS) Protocol Version 1.2 . RFC 5246 (Proposed Standard), août 2008. Updated by RFCs 5746, 5878, 6176, 7465, 7507, 7568, 7627, 7685, 7905

[20]. Stephen A .Thomas ―SSL and TLS Essentials Securing the Web . Wiley Computer Publishing, John Wiley & Sons, Enc (2000)

[21]. Recommandations de sécurité relatives à TLS, Document réalisé par l'ANSSI, N°SDE-NT-35/ANSSI/SDE/NP, Version 1.1 : 19/08/2016.

[22]. Oppliger, R. ―SSL and TLS: Theory and Practice‖, Artech House, Suisse, (2009), ISBN: 9781596934481

[23]. Pravir Chandra, Matt Messier, John Viega― Network Security with Openssl . O'Ruilly, June (2002)

[24] Labouret, G. ―IPSec: présentation technique‖,Hervé schauer consultants, France, (2000) [online] http://www.hsc.fr/

[25] Michael, E.W. and Herbert J. M. ―Principles of Information Security‖, Course Technology Cengage Learning, USA, (2009) , ISBN-13: 9781111138219.

[26] Doraswamy, N., Harkin, D. ―IPSec: The New Security Standard for the Internet, Intranets, and Virtual Private Networks (2nd Edition)‖, Prentice Hall, États-Unis. (2003) ISBN: 007-6092018759

[27] Lasserre, X. and Klein, T. ―Réseaux Privés Virtuels–Vpn‖,(2011) [online] http://www.frameip.com/vpn/

[28] Doraswamy, N., Harkin, D. ―IPSec: The New Security Standard for the Internet, Intranets, and Virtual Private Networks (2nd Edition)‖, Prentice Hall, États-Unis. (2003) ISBN: 007-6092018759.

[29] Kozierok, M.C. ―The TCP/IP Guide‖, (2005) [online] http://www.tcpipguide.com/free/t_IPSecAuthenticationHeaderAH.htm

[30] Van Quang, D. ―Contribution à l'étude de la qualité de service pour les protocoles sécurisés de télécommunications. Application à IPSec‖, l'université Paris XII-Val Marne, (2005) ,[online] http://doxa.upec.fr/theses/th0231084.pdf

[31] Kent, S. and Atkinson, R. ―RFC 2402: IP Authentication Header (AH)‖, IETF, (1998) [online] http://www.ietf.org/rfc/rfc2402.txt

[32] Kent, S. and Atkinson, R. ―RFC 2406: IP Encapsulating Security Payload (ESP)‖, (1998) ,IETF [online] http://www.ietf.org/rfc/rfc2406.txt

[33] Allard, F. and Bonnin, J.M. ―An application of the context transfer protocol: IPsec in a IPv6 mobility environment‖, Communication Networks and Distributed Systems, Vol.1, No.1, pp.110-126. (2008)

[34] Thomas, J. and Elbirt, A.J. ―Understanding Internet Protocol Security‖, Information Systems Security, Vol. 13, No. 4, pp.39- 43. (2006)

[35] Perlma, R. and Kaufman, C. ―Analysis of the IPSec Key Exchange Standard‖, In Proceedings of Tenth IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, Cambridge, USA, pp.150-156, (2001)

[36] Su, M. and Chang, J.F. ―An efficient and secured internet key exchange protocol design‖, In Proceedings of the fifth annual conference on Communication Networks and Services Research (CNSR'07), Fredericton, New Brunswick, Canada, pp.184-192, (2007)

[37] Cheng, P.C, Garay, J. A., Herzberg, A. and Krawczyk, H. ―An architecture for internet key exchange protocol‖, IBM System, Vol. 40, No. 3, pp.721-746. (2001)

[38] Zhu, x., Haigang, Z. and Jun, L. ―Analysis and Improvement of IKEv2 against Denial of Service Attack‖, In Proceedings of International Conference on Information, Networking and Automation (ICINA), Kunming, pp.350-355, (2010)

[39]Julien Rosener, Le contrôle CRC, 2004, [online] https://docplayer.fr/3573726-Julien-rosener-julien-rosener-digital-scratch-org-le-controle-crc-17-05-2004-rajout-des-references.html

[40] La fonction logique OU Exclusif,[online]https://www.positron-libre.com/cours/electronique/logique-combinatoire/fonctions-logiques/fonction-ou-exclusif-xor.php

[41] Structure interne des ordinateurs, arithmétique, IFT-17583, [online]http://www2.ift.ulaval.ca/~marchand/ift17583/Support/Arithm.html#:~:text=La%20division

%20binaire%20s'effectue,diviseur%2C%20sinon%20il%20est%200