الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'enseignement Supérieur et de la Recherche scientifique

Université Mohamed Khider Biskra
Faculté des Sciences et de la Technologie
Département de Génie Electrique
Filière : électronique
Option : télécommunication

Réf:……

Mémoire de Fin d'Etudes
En vue de l'obtention du diplôme :

MASTER

Thème

Sécurisation des communications dans les réseaux d'ordinateurs (couche SSL)

Présenté par :
Ait Ameur Yacine Mehdi
Soutenu le : Juin 2014

Devant le jury composé de :

M. Hezabra Adel	**MAB**	**Président**
Mlle. Hendaoui Mounira	**MAB**	**Examinatrice**
M. Guesbaya Tahar	**MCA**	**Encadreur**

Année universitaire : 2013 / 2014

الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'enseignement Supérieur et de la recherche scientifique

Université Mohamed Khider Biskra
Faculté des Sciences et de la Technologie
Département de Génie Electrique
Filière : électronique
Option : télécommunication

Mémoire de Fin d'Etudes
En vue de l'obtention du diplôme :

MASTER

Thème

Sécurisation des communications dans les réseaux d'ordinateurs (couche SSL)

Présenté par :

Ait Ameur Yacine Mehdi

Avis favorable de l'encadreur :

Nom Prénom ***signature***

Avis favorable du Président du Jury

Nom Prénom ***Signature***

Cachet et signature

الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'enseignement Supérieur et de la Recherche scientifique

Université Mohamed Khider Biskra
Faculté des Sciences et de la Technologie
Département de Génie Electrique
Filière : électronique
Option : télécommunication

Thème :

Sécurisation des communications dans les réseaux d'ordinateurs (couche SSL)

Proposé par : Ait Ameur Yacine Mehdi
Dirigé par : Guesbaya Tahar

Résumé

L'acheminement des données entre les ordinateurs est garanti par le réseau. Pour le faire, celui-ci doit effectuer diverses opérations d'un bout à l'autre de ses nœuds (ordinateurs). Aussi, pour garantir la sécurité des échanges de données, des opérations additionnelles sont effectuées. Toutes ces opérations s'appellent protocoles et sont toutes basées sur le codage et la cryptographie.

Dans ce travail, on a étudié les protocoles du réseau, les outils cryptographiques, ainsi que les méthodes les plus courantes en termes de sécurisation des communications dans les réseaux d'ordinateurs.

تقوم الشبكة بإيصال المعلومات بين أجهزة الكمبيوتر، ولفعل هذا يجب أن تمر بمراحل من التحويل والمعالجة وهذا على كلا الطرفين من الشبكة بالإضافة إلى أنه من أجل ضمان الحماية لهذه المعلومات يجب أن تمر بمراحل أخرى من المعالجة.
كل هذه العمليات تسمى بروتوكولات وهي تعتمد على التشفير في أداء مهامها.
في هذا العمل قمنا بدراسة البروتوكولات الخاصة بالشبكة، التشفير وحماية الاتصالات في شبكات الكمبيوتر.

Dédicaces

Je dédie ce modeste travail à mes parents qui ont veillé à ce que je puisse arriver à ce stade de mes études, que Dieu me les garde.

Je le dédie aussi à Farouk et tous les membres de ma famille ainsi que mes amis qui m'ont encouragé.

Remerciement

Je tiens tout d'abord à remercier Dieu le tout puissant et miséricordieux, qui m'a donné la force et la patience d'accomplir ce Modeste travail.

En second lieu, je tiens à remercier mes parents qui m'ont épaulé pendant tout mon cursus.

Je tiens à remercier mon encadreur Mr : T.Guesbaya, qui m'a proposé ce sujet et dirigé mon travail.

Mes vifs remerciements vont également aux membres du jury pour l'intérêt qu'ils ont porté à ma recherche en acceptant d'examiner mon travail et de l'enrichir par leurs propositions.

Enfin, je tiens également à remercier toutes les personnes qui ont contribué, de près ou de loin à la réalisation de ce travail.

Liste des tableaux

Liste des figures

Abréviations et acronymes

AC	Autorité de Certification
AES	Advanced Encryption Standard
BSD	Berkeley Software Distribution
CBC	Cipher Bloc Chaining mode
CCITT	Comité Consultatif International Téléphonique et Télégraphique
CCS	Change Cypher Specification
CRC	Cyclic Redundancy Check
CRL	Certificate Revocation List
DEC	Digital Equipment Corporation
DES	Data Encryption Standard
DH	Diffie-Hellman
DLP	Discret Logarithm Problem
DNA	Distributed Network Architecture
FIPS	Federal Information Processing Standard
FTAM	File Transfer and Access Management
FTP	File Transfer Protocol
GNU	Gnu's Not Unix
GPRS	General Packet Radio Service
HTTPs	Hyper Text Transfer Protocol Secure
IBM	International Business Machines
IDEA	International Data Encryption Algorithm
IETF	Internet Engineering Task Force
IMAP	Internet Message Access Protocol
IP	Internet Protocol
IPsec	Internet Protocol Secure
ISO	International Standard Organization
MD5	Message Digest 5
MHS	Message Handling System
MSIE	Microsoft Internet Explorer
NIST	National Institute of Standards and Technology
NSA	National Security Agency
OCSP	Online Certificate Status Protocol
ONU	Organisation des Nations Unies
OSI	Open Systems Interconnection
PCI	Protocol Control Information
PDU	Protocol Data Unit
PKI	Public Key Infrastructure
POP	Post Office Protocol
RC4	Rivest Code 4
RSA	Rivest Shamir Adleman
SDU	Service Data Unit
SHA	Secure Hash Algorithm
SNA	System Network Architecture
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP	Transport Control Protocol

TLS	Transport Layer Secure
UDP	User Datagram Protocol
VPN	Virtual Private Network
WIFI	Wireless Fidelity
WWW	World Wide Web

Sommaire

Introduction générale

Introduction générale

La communication est un besoin vital pour l'espèce humaine depuis son apparition, et s'effectue à l'aide de tout ou partie de ses sens, d'abord par l'image, ensuite par le son, et enfin par l'écrit. Les télécommunications permettent de mettre en relation des individus qui sont géographiquement éloignés, leur développement et leur amélioration ont fait l'objet d'une recherche constante produisant des solutions de plus en plus rapides, travaillant à l'échelle planétaire voire spatiale.

Internet est un réseau mondial reliant des milliards d'ordinateurs qui communiquent entre eux. L'ampleur de ce réseau a permis l'apparition de plusieurs sociétés proposant divers services (banque en ligne, achat ou vente en ligne, messagerie, réseaux sociaux etc.). Bon nombre de ces services nécessitent des informations privées ou confidentielles qui peuvent nuire à l'usager en cas de vol ou de perte de ces informations. Pour cela, des méthodes de sécurisation ont été mises en place.

L'objectif de ce travail est de démontrer comment une communication entre deux ordinateurs peut être sécurisée.

Pour arriver à cet objectif, on est passé par ces chapitres :

- Le premier chapitre consiste à démontrer comment une communication est établie entre deux ordinateurs afin de faire connaitre les différentes couches de protocoles du réseau.
- Le deuxième chapitre contient les termes de base de la cryptographie utilisés dans la sécurisation des réseaux.
- Dans le troisième chapitre, on a pris les deux protocoles les plus utilisés pour la sécurisation des communications dans les réseaux d'ordinateurs.
- Dans le quatrième chapitre, on a élaboré un exemple de la cryptographie de césar illustré dans un programme en C++ qui envoie un message crypté à partir d'un ordinateur jusqu'à un autre qui décrypte le message. A l'aide d'une clé et ensuite, on a simulé une attaque en capturant les paquets réseau transférés.
- A la fin, on a terminé par une conclusion générale.

Chapitre 1 :
Notions de base sur les réseaux d'ordinateurs

1.1. Introduction :

La communication entre les ordinateurs est un échange de données (bits). Pour qu'elle soit établie, cette communication nécessite un support. Ce support s'appelle réseau, il s'occupe de l'acheminement des données entre les ordinateurs. Qu'il soit filaire (paire torsadée, câble coaxial…) ou sans fil (WIFI, WI-MAX, GPRS…), le but du réseau reste le même, garantir un bon échange de données entre les ordinateurs connectés. Pour y arriver, ces données doivent passer par plusieurs processus (ajout de bits, conversion…). On les appelle protocoles. Ces protocoles se réunissent sous la forme d'une pile appelée modèle.

Afin de comprendre les protocoles et ainsi le fonctionnement du réseau, l'étude d'un modèle s'impose. Le modèle en couches ISO/OSI est un modèle de référence sur lequel est basée la communication dans les réseaux d'ordinateurs. Son étude nous fournira les connaissances de base des communications dans les réseaux d'ordinateurs.

1.2. Etude du modèle en couches ISO/OSI :

1.2.1. Définition :

Le modèle OSI est un modèle standard des communications dans les réseaux d'ordinateurs. Il justifie de tous les protocoles nécessaires au bon fonctionnement du réseau. Comme son nom l'indique (modèle en couches), il se compose de 7 couches de protocoles. Chacune de ces couches exécute une opération spécifique sur les données envoyées par la couche précédente pour qu'elles soient ensuite envoyées à la couche suivante à travers toute la pile formant le modèle.

1.2.2. Historique [1]:

Au départ, les constructeurs informatiques ont proposé des architectures réseaux propres à leurs équipements. Par exemple, IBM a proposé SNA, DEC a proposé DNA.... Ces architectures ont toutes le même défaut : du fait de leur caractère propriétaire, il n'est pas facile de les interconnecter, à moins d'un accord entre constructeurs. Avec l'apparition d'internet, l'interconnexion entre les architectures réseaux, quelques soient leurs appartenances, s'est avérée nécessaire.

Pour éviter la multiplication des solutions d'interconnexion d'architectures hétérogènes, l'ISO (International Standards Organisation), organisme dépendant de l'ONU et composé de 140 organismes nationaux de normalisation, a développé un modèle de

référence appelé modèle OSI (Open Systems Interconnection). Ce modèle décrit les concepts utilisés et la démarche suivie pour normaliser l'interconnexion de systèmes ouverts (un réseau est composé de systèmes ouverts lorsque la modification, l'adjonction ou la suppression d'un de ces systèmes ne modifie pas le comportement global du réseau).

Au moment de la conception de ce modèle, la prise en compte de l'hétérogénéité des équipements était fondamentale. En effet, ce modèle devait permettre l'interconnexion avec des systèmes hétérogènes pour des raisons historiques et économiques. Il ne devait en outre pas favoriser un fournisseur particulier. Enfin, il devait permettre de s'adapter à l'évolution des flux d'informations à traiter sans remettre en cause les investissements antérieurs. Cette prise en compte de l'hétérogénéité nécessite donc l'adoption de règles communes de communication et de coopération entre les équipements, c'est à dire que ce modèle devait logiquement mener à une normalisation internationale des protocoles.

Le modèle OSI n'est pas une véritable architecture de réseau, car il ne précise pas réellement les services et les protocoles à utiliser pour chaque couche. Il décrit plutôt ce que doivent faire les couches. Néanmoins, l'ISO a écrit ses propres normes pour chaque couche, et ceci de manière indépendante au modèle.

Le tableau ci-dessous illustre les publications de l'ISO sur le modèle OSI[2] :

Année	Certificat	Publication
1977		Publications d'articles sur le modèle OSI
1978	CCITT X.200	Premiers modèles OSI (Comité technique 97/SC16) Premier modèle OSI du CCITT (livre gris)
1980	Les séries CCITT T	Télex
1983		Premier travaux sur OSI par le NIST
1984	ISO7498 Les séries X.400	Modèle de référence OSI Prise en charge des messages (MHS)
1987	ISO 8326/8327	Protocole Session
1988	ISO 8571 Séries X.500 ISO 8822/8823 ISO 8072/8073 ISO 8473/9542 FIPS 146	Transfert de ficher (FTAM) Le chemin ISO 9594 Protocole présentation Protocole de transport Protocole et routage internet Première sortie d'US GOSIP (8/88)
1990	ISO 9595/9596	Service et protocole de gestion

Tableau 1.1 : Publications de l'ISO concernant le modèle en couches OSI

1.2.3. Principes :

Le modèle OSI est basé sur trois principes :

1.2.3.1. Architecture en couches :

La plupart des systèmes de communication sont construits selon une architecture en couches, c'est-à-dire une segmentation en plusieurs niveaux, empilés l'un sur l'autre. Chaque niveau à un rôle différent mais tous participent à la transmission de la communication entre plusieurs nœuds.

Chaque couche réalise donc une partie des opérations nécessaires et est liée aux deux couches adjacentes par la notion d'interface, qui est l'ensemble des opérations proposées à la couche supérieure et utilisées par la couche inférieure. Deux couches de même niveau de nœuds différents peuvent ainsi communiquer, en utilisant un protocole spécifique.

Voici une figure qui illustre l'architecture en couches [2]:

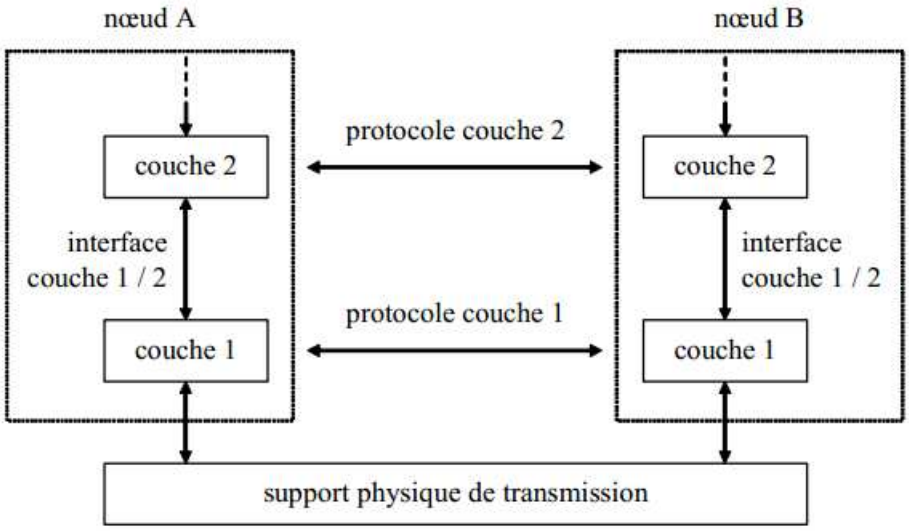


Figure 1.1 : Architecture en couches d'un réseau

1.2.3.2. Communication virtuelle :

Les couches de même niveau ne communiquent pas directement entre elles, mais via un mécanisme transparent ; on parle ici de communication virtuelle.

Au niveau du nœud émetteur, une donnée émise d'une couche (n) est prise en charge par la couche (n-1) qui réalise diverses opérations et transmet la donnée à la couche (n-2) et ainsi de suite jusqu'à atteindre la couche la plus basse laquelle a notamment pour fonction

de gérer l'accès au support physique de transmission et véhicule notamment la donnée jusqu'à l'autre nœud.

La donnée est alors prise en charge par la couche la plus basse du nœud destinataire, qui réalise diverses opérations et la transmet à la couche supérieure et ainsi de suite, jusqu'à arriver à la couche (n) où l'on doit retrouver la donnée telle qu'elle a été émise de cette même couche dans le nœud émetteur, comme si les couches avaient pu communiquer directement entre elles.

1.2.3.3. Interface entre deux couches :

Le découpage en couches nécessite que chacune d'entre elles propose un nombre d'opérations à la couche qui lui est supérieure. La manière dont les opérations d'une couche (n) mises à disposition à la couche (n+1) ont été réalisées doit rester méconnue par cette dernière, car à la fois elle n'en a pas besoin, et aussi parce que la manière dont elle réalise ses propres opérations doit en être indépendante ; on parle alors d'interface.

Les opérations ainsi mises à disposition par une couche (n) sont appelées services ; la couche (n) est un fournisseur de services, alors que la couche (n+1) est un utilisateur de services ; la couche (n) est en elle-même un utilisateur des services de la couche (n-1).

Au niveau du nœud émetteur, lorsqu'une donnée est émise de la couche (n), celle-ci fait appel à l'un des services de la couche (n-1) et lui transmet une trame constituée de la donnée à émettre et des informations de contrôle, qui permettront à la couche (n) du destinataire d'exploiter les données reçues.

La donnée à transmettre appelée SDU (pour Service Data Unit) est associée aux informations de contrôle (PCI : Protocol Control Information) par encapsulation dans une trame de données PDU (Protocol Data Unit). Cette trame est alors considérée par la couche (n-1) comme la donnée qu'elle doit prendre en charge (SDU). Ceci est illustré dans le schéma.

Le schéma suivant illustre l'interface entre deux couches [2] :

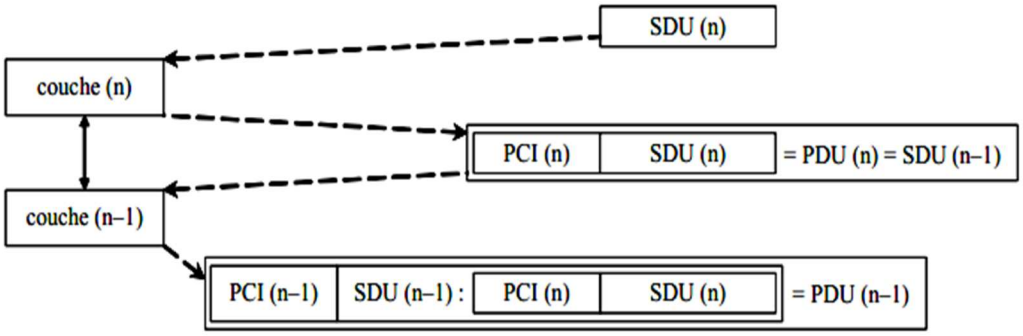


Figure 1.2 : Encapsulation des données lors du passage vers une autre couche

1.2.4. Les différentes couches du modèle OSI :

Le modèle OSI est découpé en 7 couches dont on distingue :

- Couches de 1-4 : Appelées les couches basses, elles s'occupent du transfert de l'information par les différents services de transport.
- Couches de 5-7 : Appelées les couches hautes, elles s'occupent du transfert de l'information par les différents services applicatifs.

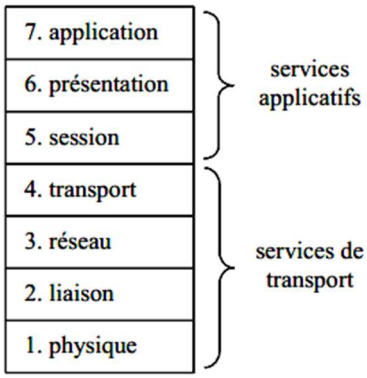


Figure 1.3 : Le modèle OSI et ses différentes couches

1.2.4.1. Couche physique :

C'est la première couche du modèle OSI. Elle définit les caractéristiques électriques et mécaniques des interconnexions entre les ordinateurs. Le rôle ultime de cette couche est de définir comment envoyer un 1 ou un 0. Les machines sont les équipements électriques sur les réseaux et peuvent être des ordinateurs mais aussi des concentrateurs ou de simples répéteurs.

Dans les réseaux locaux, la couche 1 fournit aussi des services restreints de couche 2. Ces services restreints sont les règles d'accès, la délimitation de la trame, l'adressage

physique, l'information sur le type de protocole transporté et enfin l'adjonction d'un code de redondance permettant de détecter les trames erronées.

1.2.4.2. Couche liaison de données :

La couche liaison assure la fiabilité de la transmission en élaborant les datagrammes à partir des données de la couche réseau, à destination de la couche physique. Cela signifie que les données seront structurées en datagrammes ou paquets que la couche physique se chargera de transmettre. Le type de datagrammes dépendra du type du réseau. Ainsi, un paquet Ethernet n'est pas structuré de la même manière qu'un paquet Token Ring.

Le but principal de la couche liaison est de garantir aux couches de niveau supérieur une transmission fiable par le réseau. Cette couche doit donc recevoir un accusé de réception des données expédiées. Si elle ne le reçoit pas, elle renouvelle la transmission. Le type de l'accusé de réception dépend également du type de réseau.

1.2.4.3. Couche réseau :

Cette couche assure la transmission des données sur les réseaux. C'est ici que la notion de routage intervient, permettant l'interconnexion de réseaux différents. C'est dans le cas de TCP/IP, la couche Internet Protocol. En plus du routage, cette couche assure la gestion des congestions. Lorsque les données arrivent sur un routeur, il ne faudrait pas que le flot entrant soit plus gros que le flot sortant maximum possible, sinon il y aurait congestion. Une solution consiste à contourner les points de congestion en empruntant d'autres routes. Le problème de la congestion est un problème épineux, auquel il nous arrive assez souvent d'être confrontés.

1.2.4.4. Couche transport :

La couche transport assure le transfert sans erreur des paquets. Elle subdivise en petits blocs les messages longs. Les paquets trop petits sont assemblés en grands paquets. Les paquets générés sont numérotés et transmis à la couche 3. Symétriquement, les données sont extraites des paquets reçus par le destinataire, mises en ordre et un accusé de réception est éventuellement envoyé.

Les contrôles de flux et d'erreurs sont également assurés par la couche transport. Elle traite les erreurs de constitution des paquets et de transmission de données ainsi que la réception par la station cible. Lorsque deux systèmes communiquant ouvrent une session ou

créent une liaison, cela se déroule au niveau 4 du modèle OSI, ainsi que de coutume lors d'un transport par TCP.

1.2.4.5. Couche session :

Le niveau session assure l'établissement correct, le maintien et l'arrêt d'une communication sécurisée de deux applications de plusieurs ordinateurs.

Egalement qualifiée de couche de contrôle des communications. Elle gère les noms de ressource et prend en compte l'aspect sécurité de l'application. Dès qu'une application tente de communiquer avec une application d'un autre ordinateur, la couche session se procure l'adresse de l'ordinateur cible et demande à la couche transport d'établir une liaison. Aux applications sont alors proposés des services pour contrôler cette liaison. Ces services, tels que prescrits par ISO, sont articulés selon la classe Basic Combined pour les fonctions générales, la classe Basic Synchronized pour les méthodes de synchronisation de la liaison, et la classe Basic Activity pour gérer les activités réseau entre les stations.

1.2.4.6. Couche présentation :

Cette couche définit un format des données par lequel les informations circuleront dans le réseau. Les données de la couche Application sont adaptées à un format uniforme pour que tous les ordinateurs concernés puissent les traiter.

Cela est nécessaire car les plates-formes PC, Macintosh ou les différents UNIX représentent les données de manière différente. Il convient donc d'adopter une représentation unique si nous souhaitons que ces plates-formes puissent communiquer. Les données sont alors transcrites en un format intermédiaire et transmises en ce format. Le destinataire retranscrira les données reçues en fonction des impératifs de la plate-forme. Cette couche traite également des éléments tels la compression, le changement de jeux de caractères ou le codage des données. Dans le cas des ordinateurs qui, dans un intranet, sont exploités en tant que serveurs ou stations de travail, nous trouverons également les utilitaires qui assurent des entrées/sorties au travers du réseau. Nous pensons ici aux lecteurs ou aux imprimantes en réseau.

1.2.4.7. Couche application :

La couche application est la couche supérieure du modèle OSI et donne accès aux services réseau de l'ordinateur. Elle comprend des programmes tels que le navigateur qui

permet à l'utilisateur de lire des pages web, le client FTP pour le téléchargement de fichiers, le programme de messagerie, les applications de base de données et de nombreux autres logiciels qui nécessitent un accès au réseau.

1.2.5. Transmission des données à travers le modèle OSI :

La transmission de données à travers le modèle OSI utilise le principe de communication virtuelle en usant des interfaces inter-couches. Il y'a donc encapsulation successive des données à chaque interface.

La figure ci-dessous démontre le passage de la donnée à travers le modèle OSI [2] :

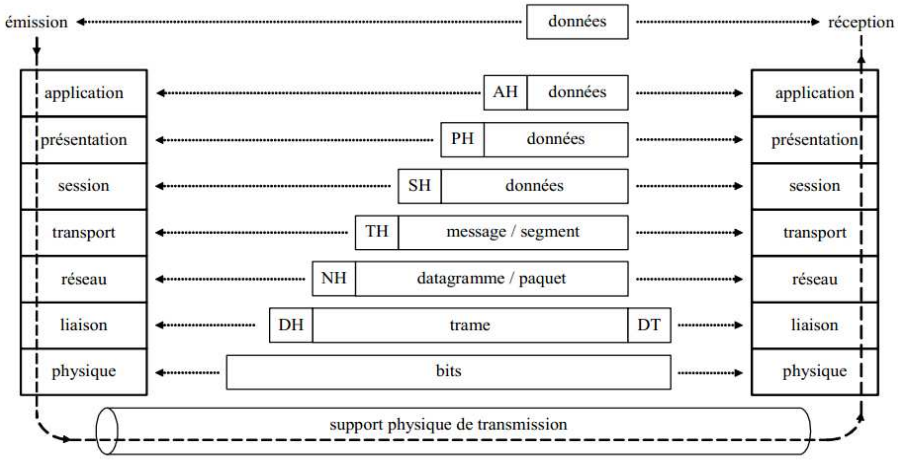


Figure 1.4 : Transmission des données via le modèle OSI (H : en-tête, T : en-queue)

1.3. Conclusion :

Le modèle en couches ISO/OSI est un cadre compatible avec les protocoles et les types de réseaux les plus courants. Cependant, cette compatibilité l'a rendu complexe. De plus, les couches présentation et session sont rarement utilisées ce qui les transforme en fardeau pour le réseau.

Par souci de rapidité et d'efficacité d'un réseau, le modèle en couches ISO/OSI n'a pas trouvé sa place. Les réseaux utilisent des protocoles plus légers et plus rapides que celui-ci, ce qui a compromis sa réalisation pratique.

Malgré cela, le modèle en couches OSI reste le modèle type pour la compréhension des réseaux et de leur fonctionnement et reste ainsi le modèle de référence pour les constructeurs de matériels pour les réseaux.

Chapitre 2 :

Outils cryptographiques pour la sécurisation

2.1. Introduction :

Les méthodes de sécurisation des communications dans les réseaux d'ordinateurs utilisent différentes sortes d'algorithmes de cryptographie afin de crypter les données échangées dans le réseau et empêcher tout utilisateur non désiré de lire et de modifier le contenu de ces données.

Une connaissance de base des algorithmes de cryptographie est nécessaire à la compréhension des mécanismes de sécurité des réseaux d'ordinateurs.

2.2. Définition :

La cryptographie est le domaine de la cryptologie où on étudie et on établit des procédés de chiffrement des informations afin de les rendre illisibles pour certaines personnes.

2.3. Historique :

2.3.1. Cryptographie antique :

L'utilisation de la cryptographie remonte à l'époque antique. En effet, une tombe égyptienne récemment découverte datant de 4000ans contenait des hiéroglyphes qui ont été modifiés afin d'assombrir le sens des inscriptions.

2.3.2. Scytale spartiate :

Mais la vraie utilisation de la cryptographie fut au 5ème siècle avant J.C, en Grèce antique, elle était basée sur la transposition des lettres. La cité Sparte utilisait un bâton d'un diamètre connu, autour duquel est enroulée une bandelette sur laquelle on écrit le message. Une fois terminé, la bandelette est déroulée et le message devient de ce fait incompréhensible. A la réception, on ré-enroule la bandelette autour d'un bâton de même diamètre qu'à l'émission afin d'obtenir le message clair. La Scytale spartiate est représentée par la figure suivante [3]:

Figure 2.1 : Scytale spartiate

2.3.3. Carré de Polybe :

150 ans avant J.C, l'écrivain grec, Polybe mit en place un nouveau procédé cryptographique, la cryptographie par substitution. Elle se présente sous la forme d'un tableau de 25 cases, où chaque case contient une lettre de l'alphabet ce qui fait que chaque lettre est représentée par une paire de chiffres indiquant le numéro de ligne et de colonne dans le tableau. Au lieu d'envoyer des lettres dans le message, on envoie des chiffres et à la réception, on a le même tableau qui va aider à déchiffrer le message.

	1	2	3	4	5
1	a	b	c	d	e
2	f	g	h	i, j	k
3	l	m	n	o	p
4	q	r	s	t	u
5	v	w	x	y	z

Tableau 2.1 : Carré de Polybe

2.3.4. Cryptographie de César (substitution mono-alphabétique) :

Un siècle plus tard, précisément en 44 avant J.C, une autre méthode cryptographique a vu le jour, la substitution mono-alphabétique. Elle fut utilisée par Jules César, celui-ci faisait décaler les lettres des messages qu'il envoie à ses généraux de 3 chiffres vers le droite dans l'ordre alphabétique, par exemple pour envoyer un 'A', il écrivait 'D', pour envoyer un 'B', il écrivait 'E' etc. A la réception, on faisait décaler de 3 les lettres dans le sens inverse de l'alphabet pour obtenir le message.

2.3.5. Le code de Vigenère (substitution poly-alphabétique) :

En 1586, une méthode cryptographique plus avancée a été inventée, la cryptographie par substitution poly-alphabétique. Basée sur la cryptographie de César, cette méthode effectue le décalage par un mot où chaque lettre indique le décalage alphabétique à appliquer sur le texte en clair.

2.3.6. Mécanisation de la cryptographie :

La mécanisation de la cryptographie consiste à développer des machines effectuant le cryptage des textes en clair automatiquement. De ce fait, plusieurs machines ont été inventées.

2.3.6.1. Machine de Jefferson :

Cette machine effectue le décalage poly-alphabétique automatiquement. Son fonctionnement est simple, on fait tourner les rouleaux pour former le message qu'on souhaite envoyer, et on choisit un autre message se trouvant sur le cylindre. Du côté du récepteur, il doit disposer de cette même machine. Pour déchiffrer, il doit faire tourner les rouleaux afin de formuler le message reçu (crypté) et recherche ensuite le message clair.

2.3.6.2. Enigma :

L'histoire de la machine Enigma commença en 1919, quand un ingénieur hollandais, Hugo Alexander Koch, déposa un brevet de machine à chiffrer électromécanique. Ses idées ont été reprises par le Dr Arthur Scherbius, qui crée à Berlin une société destinée à fabriquer et à commercialiser une machine à chiffrer civile : l'Enigma. Cette société fit un fiasco, mais la machine Enigma avait attiré l'attention des militaires. Sa cryptographie a été brisée par une équipe polonaise (Marian Rejewski) en 1933. Elle a été renforcée par les allemands pour être utilisée pendant la 2nde guerre mondiale avant qu'elle soit à nouveau brisée par les bombes de Turing.

Le cryptage repose ici sur cinq éléments :

- un tableau de connexions (cryptage)

- trois rotors

- un réflecteur (décryptage)

Le tableau de connexions attribue à chaque lettre de l'alphabet une autre lettre de l'alphabet.

Les rotors établissent des connexions entre différentes lettres de l'alphabet et effectuent des rotations (d'où le problème du décryptage si on ne connaît pas leur position d'origine). En effet, dès qu'une lettre a été codée, le premier rotor tourne d'un cran, le second rotor tourne d'un cran chaque fois que le premier a fait un tour et le troisième rotor avance d'un cran quand le deuxième a fait un tour.

Le réflecteur (décryptage), quant à lui effectue des permutations entre les lettres et renvoie la lettre à coder une nouvelle fois à travers les 3 rotors.

2.3.7. Cryptographie moderne :

L'avènement de l'ordinateur a transformé l'information en série de bits (1 et 0), ceci a facilité sa manipulation, ce qui a donné naissance à de nouveaux procédés cryptographiques offrant des algorithmes de plus en plus complexes avec des clés d'une taille phénoménale. Ceux-ci sont jusqu'à aujourd'hui utilisés et se développent chaque jour. Ces algorithmes seront évoqués plus tard dans ce chapitre.

2.4. Utilité de la cryptographie :

2.4.1. Confidentialité :

Il s'agit de rendre l'information inintelligible à tous les opposants tant lors de sa conservation qu'au cours de son transfert par un canal de communication.

2.4.2. Authentification :

Consiste à s'assurer que le destinataire est bien celui qui prétend être (authentification des partenaires) et d'obtenir une garantie que l'expéditeur a bien signé l'acte (authentification de l'origine des informations).

2.4.3. Intégrité :

Le contrôle d'intégrité d'une donnée consiste à vérifier que cette donnée n'a pas été altérée, frauduleusement ou accidentellement.

2.4.4. Non-répudiation :

Il s'agit de garantir l'authenticité de l'acte. L'expéditeur ne peut nier le dépôt d'information, le réceptionneur ne peut nier la remise d'information, ni l'un ni l'autre ne peut nier le contenu de cette information.

2.5. Les Différents types de cryptographie :

2.5.1. Cryptographie symétrique :

2.5.1.1. Définition :

Un algorithme de chiffrement symétrique est un couple de fonctions : une fonction de chiffrement E et une fonction de déchiffrement D. La fonction E prend en entrée un message clair p représenté sous la forme d'une suite d'éléments d'un alphabet fini, le plus souvent égal à {0,1} et à l'aide de la clé K sélectionnée (le plus souvent selon la loi uniforme) dans un ensemble K, le transforme en un message chiffré c=E (p, K). La fonction de déchiffrement réalise l'opération inverse p=D(c, K).

Figure 2.2 : Principe de la cryptographie symétrique

2.5.1.2. Principe de fonctionnement :

Pour transmettre entre deux parties A et B un message chiffré à l'aide d'un algorithme de chiffrement symétrique, il est nécessaire que A et B possèdent la clé secrète commune K.

En pratique, soit cette clé commune a été donnée aux deux parties de manière préalable, soit elle est dérivée à l'aide d'un algorithme asymétrique.

Les algorithmes de chiffrement symétrique ont des clés courtes : 80, 128 ou 256 bits et leurs performances sont en général élevées. Ils sont donc presque toujours utilisés pour réaliser le chiffrement des données à protéger. Par exemple, pour l'envoi d'un courriel sécurisé, le chiffrement à clé publique ne sert qu'à chiffrer la clé choisie aléatoirement qui permet de chiffrer l'ensemble des données à l'aide d'un algorithme symétrique.

2.5.1.3. Chiffrement par flots :

- **Chiffre de Vernam (masque jetable) :**

Basé sur le chiffrement de Vigenère, Gilbert Vernam (1917) proposa une méthode où la clé de chiffrement a la même taille du message à crypter. En plus, une clé ne peut être utilisée qu'une seule fois et elle doit être formée d'une suite de caractères aléatoires.

Cette méthode est la seule à être jugée parfaitement sûre.

L'inconvénient avec cette technique est évidemment la taille de la clé. En effet, transmettre une clé de la même taille que le message clair n'est pas chose aisée. En plus, la génération d'une suite de caractères aléatoires utilisable qu'une seule fois demande des moyens complexes. Malgré cela, elle est toujours utilisée par les pays pour communiquer avec leurs diplomates en transférant les clés dans des valises diplomatiques.

- **Le code RC4 :**

C'est le code le plus répandu. Inventé par Ronald Rivest en 1987, il fut adopté par plusieurs protocoles de sécurisation tels SSL ou encore WEP.

La clef RC4 permet d'initialiser un tableau de 256 octets en répétant la clef autant de fois que nécessaire pour remplir le tableau. Par la suite, des opérations très simples sont effectuées : les octets sont déplacés dans le tableau, des additions sont effectuées, etc. Le but est de mélanger autant que possible le tableau. Finalement on obtient une suite de bits pseudo-aléatoires qui peuvent être utilisés pour chiffrer les données via un XOR.

- **Le code E0 :**

E0 est un algorithme de chiffrement par flot utilisé par le protocole Bluetooth pour protéger les transmissions. Il crée une suite pseudo-aléatoire avec laquelle on effectue un XOR avec les données. La clé peut avoir une taille variable mais sa longueur est généralement de 128 bits.

2.5.1.4. Chiffrement par blocs :

Comme leur nom l'indique, les algorithmes de chiffrement par blocs sont conçus pour chiffrer des blocs de messages de taille fixe. En voici des exemples :

- **DES :**

L'algorithme DES transforme un bloc de 64 bits en un autre bloc de 64 bits. Il manipule des clés individuelles de 56 bits, représentées par 64 bits (avec un bit de chaque octet servant pour le contrôle de parité). Ce système de chiffrement symétrique fait partie de la famille des chiffrements itératifs par blocs, plus particulièrement il s'agit d'un schéma de Feistel (du nom de Horst Feistel à l'origine du chiffrement Lucifer).

D'une manière générale, on peut dire que DES fonctionne en trois étapes :

- permutation initiale et fixe d'un bloc (sans aucune incidence sur le niveau de sécurité) ;
- le résultat est soumis à 16 itérations d'une transformation, ces itérations dépendent à chaque ronde d'une autre clé partielle de 48 bits. Cette clé de ronde intermédiaire est calculée à partir de la clé initiale de l'utilisateur (grâce à un réseau de tables de substitution et d'opérateurs XOR). Lors de chaque ronde, le bloc de 64 bits est découpé en deux blocs de 32 bits, et ces blocs sont échangés l'un avec l'autre selon un schéma de Feistel. Le bloc de 32 bits ayant le poids le plus fort (celui qui s'étend du bit 32 au bit 64) subira une transformation ;
- le dernier résultat de la dernière ronde est transformé par la fonction inverse de la permutation initiale.

DES utilise huit tables de substitution (les S-Boxes) qui furent l'objet de nombreuses controverses quant à leur contenu. On soupçonnait une faiblesse volontairement insérée par les concepteurs. Ces rumeurs furent dissipées au début des années 1990 par la découverte de la cryptanalyse différentielle qui démontra que les tables étaient bien conçues.

- **AES (Rijndael) :**

Le principe de l'AES est très proche du DES. C'est aussi un système cryptographique produit constitué d'une suite d'opérations de permutation et de substitution.

AES travaille sur des blocs de 128 bits avec des clefs de longueur 128, 256 ou 384 bits. Le passage à une clé de 128 bits minimum rend impossible dans le futur prévisible les recherches exhaustives de clefs. Si on suppose que l'on a un algorithme capable de comparer en une seconde 2^{56} clefs (ce qui est capable de casser DES en une seconde), il lui faudra 149 mille milliards d'années pour casser AES.

2.5.2. Cryptographie asymétrique :

2.5.2.1. Définition

La cryptographie à clé publique (asymétrique) consiste en l'existence d'une paire de clés de chaque côté (émetteur et récepteur). Chaque paire est composée d'une clé secrète et d'une clé publique.

2.5.2.2. Principe de fonctionnement :

La cryptographie asymétrique se base sur des fonctions à sens unique. Cela veut dire que les données cryptées avec la clé publique ne peuvent être décryptées que s'il on possède la clé secrète. Ça signifie que même si l'on a obtenu la clé publique, on ne pourra pas déchiffrer les informations.

2.5.2.3. RSA :

Le système RSA est nommé d'après le nom de ses inventeurs: Rivest, Shamir, Adleman. Il est basé sur la fonction à sens unique produit de deux entiers $f(n, m)=n\times m$.

On a un ensemble $(A_i)_{i\in I}$ de correspondants (personnes physiques ou ordinateurs). Le principe du cryptosystème RSA est le suivant :

- Chacun des correspondants A_i choisit deux nombres premiers p_i et q_i distincts. On note $n_i = p_i q_i$, le module du cryptosystème de A_i est :
  $\phi(n_i) = (p_i - 1)(q_i - 1)$, l'indicatrice d'Euler de n_i.
- A_i choisit ensuite un nombre e_i l'exposant de la clé publique premier avec $\phi(n_i)$ (autrement dit le Plus Grand Commun Diviseur (PGCD ou GCD en anglais) de e_i et $\phi(n_i)$ est 1. On note ce PGCD $e_i \wedge \phi(n_i) = 1$, en pratique on impose de plus $1 \leq e_i \leq \phi(n_i) - 1$.
- Puis A_i calcule l'inverse di de e_i modulo $\phi(n_i)$, d_i est appelé exposant de la clé secrète, c'est à dire en pratique, on cherche $d_i \in N$ tel qu'il existe $k_i \in N$ avec $d_i \times e_i = 1 + k_i \phi(n_i)$ et $1 \leq d_i \leq \phi(n_i) - 1$.
- Chacun des correspondants A_j publie dans un annuaire public les nombres n_j et e_j qui forment sa clé publique de chiffrement et gardent secret p_j, q_j et d_j qui forment sa clé secrète de déchiffrement.

En pratique on choisit les nombres premiers pi et qi de taille comparable de telle sorte que leur produit $n_i = p_i q_i$ soit un nombre d'au moins 300 chiffres en base 10 et plutôt 500 pour une protection longue.

2.5.2.4. Diffie-Hellman :

Elaboré par Diffie, Merkel et Hellman, en se basant sur le DLP (Discret Logarithm Problem) défini comme suit :

- Supposons $G=<g>=\{g^i\}_{0\leq i<n}$ un groupe mono-gène fini d'ordre n.
- Soit h ϵ G. Alors le logarithme discret de h en base g, noté $\log_g h$, est l'unique entier x tel que $h= g^x (0\leq x<n)$.
- DLP consiste donc à résoudre le problème suivant :

  Etant donné G,g,h, trouver $x=\log_g h$.

  Diffie et Hellman ont établi un algorithme de cryptographie comme décrit par l'exmple suivant :

  Alice et Bob veulent partager une clé secrète K. on suppose que les données G,n=│G│ et g sont publiques.

  Alice choisit un entier $1\leq a\leq n-1$ au hasard.

  Alice calcule $A=g^a$ et l'envoie à Bob.

  Bob choisit un entier $1\leq a\leq n-1$ au hasard.

  Bob calcule $B= g^b$ et l'envoie à Alice.

  Alice est en mesure de calculer B^a et Bob de calculer A^b. La clé commune est donc : $K=g^{ab}=g^a=g^b$

Alice	Bob
Génère a	Génère b
$A=g^a \bmod n$	$B= g^b \bmod n$
A →	
	← B
Dispose de [a,A,B,n]	Dispose de [b,B,A,n]
Clé secrète : $K=B^a \bmod n$	Clé secrète : $K=A^b \bmod n$

Tableau 2.1 : Protocole d'échange de clé de Diffie-Hellman

2.6. Fonctions de hachage :

On nomme fonction de hachage une fonction particulière qui, à partir d'une donnée fournie en entrée, calcule une empreinte servant à identifier rapidement, bien qu'incomplètement, la donnée initiale.

Une fonction de hachage est typiquement une fonction qui pour un ensemble de très grande taille et de nature très diversifiée va renvoyer des résultats aux spécifications précises (en général des chaînes de caractère de taille limitée ou fixe) optimisées pour des applications

particulières. Les chaînes permettent d'établir des relations (égalité, égalité probable, non-égalité, ordre...) entre les objets de départ sans accéder directement à ces derniers, en général soit pour des questions d'optimisation (la taille des objets de départ nuit aux performances), soit pour des questions de confidentialité.

Le résultat d'une fonction de hachage peut être appelé selon le contexte somme de contrôle, empreinte, hash, résumé de message, condensé, condensat ou encore empreinte cryptographique lorsque l'on utilise une fonction de hachage cryptographique. Les fonctions de hachage servent à rendre plus rapide l'identification des données.

2.6.1. Le hachage MD5 (Message Digest 5) :

2.6.1.1. Définition :

Une des fonctions de hachage les plus utilisées est le MD5. Elle n'est plus considérée comme sûre pour un usage en cryptographie car durant l'été 2004, des chercheurs chinois ont montré comment réaliser des collisions avec MD5, c'est-à-dire comment produire deux messages ayant la même empreinte (le même résumé) en appliquant l'algorithme MD5. Cela dit, le MD5 est encore utilisé pour des usages non sensibles, par exemple pour vérifier l'intégrité d'un fichier téléchargé.

2.6.1.2. Fonctionnement du MD5 :

- **Etape 1 : Complétion**

Le message est constitué de b bits $m_1...m_b$. On complète le message par un 1, et suffisamment de 0 pour que le message étendu ait une longueur congruente à 448, modulo 512. Puis on ajoute à ce message la valeur de b, codée en binaire sur 64 bits (on a donc b qui peut valoir jusque 2^{64}... ce qui est énorme). On obtient donc un message dont la longueur totale est un multiple de 512 bits. On va travailler itérativement sur chacun des blocs de 512 bits.

- **Etape 2 : Initialisation**

On définit 4 buffers de 32 bits A, B, C et D, initialisés ainsi (les chiffres sont hexadécimaux, exemple : a=10, b=11...).

A=01234567 (1)

B=89abcdef (2)

C=fedcba98 (3)

D=76543210 (4)

On définit également 4 fonctions F, G, H et I, qui prennent des arguments codés sur 32 bits, et renvoie une valeur sur 32 bits, les opérations se faisant bit à bit.

F(X,Y,Z) = (X AND Y) OR (not(X) AND Z) (5)

G(X,Y,Z) = (X AND Z) OR (Y AND not(Z)) (6)

H(X,Y,Z) = X xor Y xor Z (7)

I(X,Y,Z) = Y xor (X OR not(Z)) (8)

Ce qu'il y a d'important avec ces 4 fonctions et que si les bits de leurs arguments X, Y et Z sont indépendants, les bits du résultat le sont aussi.

- **Etape 3 : Calcul itératif**

Pour chaque bloc de 512 bits du texte, on fait les opérations suivantes :

- On sauvegarde les valeurs des registres dans AA, BB, CC, DD.
- On calcule de nouvelles valeurs pour A, B, C, D à partir de leurs anciennes valeurs, à partir des bits du bloc qu'on étudie, et à partir des 4 fonctions F, G, H, I.
- On fait :

  A=AA+A (9)

  B=BB+B (10)

  C=CC+C (11)

  D=DD+D (12)

- **Etape 4 : Ecriture du résumé**

Le résumé sur 128 bits est obtenu en mettant bout à bout les 4 buffers A,B,C,D de 32 bits.

2.6.2. Le hachage SHA-1 :

2.6.2.1. Définition :

SHA-1 (Secure Hash Algorithm) est une fonction de hachage cryptographique conçue par la National Security Agency des États-Unis (NSA), et publiée par le gouvernement des États-Unis comme un standard fédéral de traitement de l'information (Federal Information

Processing Standard du National Institute of Standards and Technology (NIST)). Elle produit un résultat (appelé « hash » ou condensat) de 160 bits.

2.6.2.2. Fonctionnement :

Les spécifications de SHA-1 sont décrites pour la première fois en avril 1995 dans le document officiel du NIST FIPS 180-1 pour un standard de fonction de hachage cryptographique. Elles sont reprises dans les versions successives de ce document, FIPS-180-2, FIPS-180-3 et FIPS-180-4.

L'algorithme SHA-1 transforme un message de longueur inférieure à 2^{64} bits en un haché, ou condensé de ce message, qui est de longueur 160 bits. Il adopte la construction de Merkle-Damgaard : schéma itératif à partir d'une fonction de compression dont l'entrée est de taille fixe, et ajout en fin de message de sa longueur (renforcement de Merkle-Damgaard, ce qui permet de réduire la résistance aux collisions de la fonction de hachage à celle de la fonction de compression.

Cet algorithme peut être découpé en deux phases :

- Le prétraitement implique :
  a) de compléter le message par des informations le rendant compatible avec l'algorithme SHA-1 (remplissage)
  b) son analyse pour le découper en blocs de 512 bits
  c) l'initialisation de variables de travail
- Le calcul du condensé génère un tableau à partir du message complété, puis le transforme via l'utilisation de fonctions, de constantes, d'opérations binaires détaillées plus loin. L'ensemble effectué de manière itérative permet de générer des séries de valeurs de hachage à chaque tour. Le condensé final est le dernier état de ces valeurs de hachage.

2.7. Signature numérique :

La signature numérique (parfois appelée signature électronique) est un mécanisme permettant de garantir l'intégrité d'un document électronique et d'en authentifier l'auteur, par analogie avec la signature manuscrite d'un document papier. Elle se différencie de la signature écrite par le fait qu'elle n'est pas visuelle, mais correspond à une suite de nombres.

2.8. Conclusion :

La communication dans les réseaux d'ordinateurs, spécifiquement à travers les réseaux ouverts tel internet, est toujours exposée aux attaques visant les informations contenues dans celle-ci. Les algorithmes classiques de cryptographie s'avèrent désormais inefficaces face à ces attaques. Pour corriger ce problème, une combinaison de plusieurs algorithmes de sécurisation doit être utilisée.

Chapitre 3 :

Sécurisation des communications dans les réseaux d'ordinateurs

3.1. Introduction :

Afin de protéger des communications dans un réseau d'ordinateurs, on a recours à la cryptographie. Avec l'avancée technologique, des ordinateurs ultra performants ont vu le jour. Avec cela, les algorithmes de cryptographie auparavant jugées sûrs sont devenus désormais très vulnérables et des attaques par force brutale finissent toujours par les mettre au tapis. Pour combler cela, de nouveaux algorithmes de sécurisation ont été développés. Utilisant une combinaison de plusieurs algorithmes, ces méthodes de sécurisation ont remplacé leurs prédécesseurs et se sont révélés très efficaces contre les attaques. Dans ce chapitre, on va présenter les méthodes les plus utilisées pour la sécurisation des échanges de données.

3.2. SSH (Secure Shell)

3.2.1. Définition :

SSH (Secure SHell) est un logiciel de type client-serveur permettant l'administration sécurisée d'un système distant via une interface en ligne de commande. Ce logiciel est devenu un standard de fait pour l'administration des systèmes de type Unix.

Le principal avantage de SSH comparé à un outil comme Telnet est l'utilisation de mécanismes cryptographiques afin d'assurer la sécurité d'une session distante. SSH permet ainsi de réaliser de l'authentification (serveur vis à vis du client, utilisateur vis à vis du système) et d'assurer la confidentialité de la communication, en utilisant des algorithmes de chiffrement symétrique approuvés.

Conçu à l'origine pour les systèmes Unix, SSH est aussi disponible sur les systèmes Windows. Différentes mises en œuvre de SSH sont disponibles, aussi bien côté client que côté serveur, qu'elles soient propriétaires ou libres. Le protocole SSH se situe en haut du modèle OSI et exactement dans la couche application.

Le protocole SSH est illustré par la figure suivante [7]

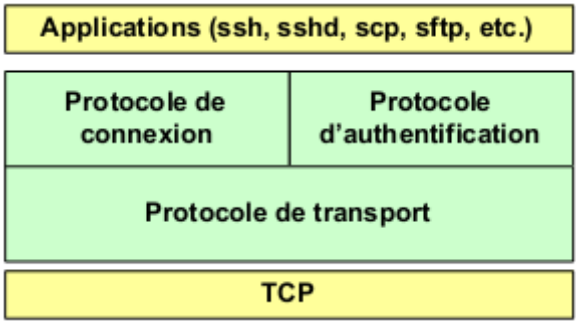


Figure 3.1 : Protocole SSH dans le modèle OSI

3.2.2. Historique [8]:

En 1995, Tatu Ylönen, un chercheur à la Helsinki University Of Technology, en Finlande a conçu la première version de ce protocole (connue aujourd'hui sous le nom SSH-1) suite aux attaques ciblant les mots de passe dans le réseau interne de l'université. Le but de SSH était de remplacer les premiers protocoles rlogin, Telnet et rsh qui ne pouvaient fournir ni l'authentification, ni la confidentialité. Ylönen a publié son implémentation en freeware en juillet 1995, et l'outil est ainsi devenu populaire. A la fin de 1995, SSH comptait environ 20 000 utilisateurs de 50 pays.

En décembre 1995, Ylönen créa son entreprise SSH Communications Security et développa SSH.

Au début, SSH utilisait des parties freeware comme GNU libgmp, mais les versions ultérieures ne contenaient que des parties propriétaires.

En 1998, une vulnérabilité a été décrite dans SSH 1.5, qui autorisait une insertion de contenu dans la suite cryptée SSH, à cause d'une protection insuffisante de l'intégrité par le code CRC-32 utilisé dans cette version du protocole. Une rectification connue comme 'SSH Compensation Attack Detector' a été incluse dans la plupart des versions. Plusieurs de ces versions mises à jour contenaient une nouvelle vulnérabilité, les attaquants pouvaient exécuter des codes aléatoires avec les privilèges de SSH daemon.

En l'an 2000, il y'avait environ 2 millions d'utilisateurs de SSH.

En janvier 2001, une nouvelle vulnérabilité a été découverte, qui permettait aux attaquants de récupérer le dernier bloc de la session cryptée avec IDEA. Le même mois, une autre faille a été découverte, celle-ci permettait à un serveur malicieux de détourner le client vers un autre serveur.

En 2006, nouvelle version (SSH-2) a été adoptée comme standard par l'IETF. Cette version offre une meilleure sécurisation en incluant la cryptographie de Diffie-Hellman et des codes d'authentification des messages. Cette nouvelle version offre en plus la possibilité d'ouvrir plusieurs sessions avec une seule connexion SSH.

3.2.3. Utilité du protocole SSH :

SSH est un protocole qui peut être utilisé pour une multitude d'applications à travers diverses plates-formes comme les variables d'UNIX (Linux, BSD utilisant le système d'exploitation d'Apple), ainsi que Windows. Certaines de ces applications requièrent des fonctionnalités qui ne se trouvent ou ne sont compatibles qu'avec des serveurs ou clients spécifiques de SSH. Voici quelques exemples d'utilisation de SSH :

- Pour se connecter à un poste à distance (à la place de Telnet et rlogin).
- Pour exécuter une commande sur un ordinateur distant.
- Sécuriser les transferts de fichiers.
- En le combinant avec rsync, on peut sauvegarder, copier et partager des fichiers d'une manière sécurisée et efficace.
- Pour la gestion automatique des serveurs distants.

La couche SSH se compose de trois sous-couches. Celles-ci se superposent dans la partie haute du modèle OSI (couche application).

Le tableau ci-dessous décrit la fonction de chacune de ces sous-couches :

Sous-couche	Rôle à jouer dans la connexion SSH
Transport	Authentification du serveur, négociation de l'algorithme de cryptographie, échange de la clé de session, intégrité des données, identification de la session
Authentification	Authentification du client, changement du mot de passe
Connexion	Transfert de port TCP et transfert X, Transfert d'agent d'authentification, Gestion des sessions interactives, exécution des programmes distants, contrôle de flux, Gestion des terminaux, Compression des données.

Tableau 3.1 : Description des sous-couches SSH

3.2.4. Fonctionnement de SSH :

- Dès que la connexion est établie (protocole TCP sur le port 21 du serveur), le client et le serveur échangent en clair leurs numéros de version du protocole SSH. Sitôt ce premier échange est effectué, le client et le serveur utilisent un protocole binaire par paquet.
- Le serveur commence par envoyer au client la liste des méthodes de cryptage supportées, la liste des méthodes d'authentification supportées, des indicateurs d'extensions de protocole (par exemple, la méthode de compression, etc.) et un cookie sur 64 bit que le client devra renvoyer. Ce cookie a comme but de protéger le serveur contre une attaque par déni de service.
- Le client envoie à son tour la liste des méthodes de cryptage supportées, la liste des méthodes d'authentification supportées et une copie du cookie du serveur.
- Le client et le serveur choisissent les meilleurs algorithmes supportés par les deux.
- Le client et le serveur calculent séparément un identifiant de session à partir des valeurs Diffie-Hellman échangées entre les deux entités. SSHv2 utilise aussi une méthode d'échange des groupes DH (par exemple, Diffie-Hellman-Group1-sha1) pour simplifier l'échange Diffie-Hellman.
- Le serveur envoie sa clé publique au client et signe avec sa clé privée les valeurs échangées précédemment.
- Le client vérifie la signature du client et passe ensuite en mode crypté.
- Le serveur répond au client par un message de confirmation crypté.
- Les deux parties, le client et le serveur, sont maintenant en mode crypté avec utilisation de l'algorithme et de la clé sélectionnés.
- Le client envoie maintenant la demande d'un service.
- Le serveur précise les méthodes d'authentification qu'il peut accepter.
- Finalement, le client envoie sa méthode d'authentification que le serveur accepte ou rejette. Dans la plupart des implémentations SSH et suivant la politique du serveur, le client a le droit à trois essais pour s'authentifier.

Le rôle principal de la sous-couche transport est de fournir un tunnel sécurisé en s'occupant du cryptage et du décryptage des données. Durant l'échange des clés, le serveur s'identifie au client au moyen d'une clé publique RSA ou DSA, ce qui veut dire que la clé privée du serveur n'est pas encore connue par le client. Ensuite, lors des connexions

suivantes, la clé du serveur peut être vérifiée au moyen d'une version enregistrée au niveau du client, ce qui permet au client de s'assurer qu'il communique bien avec le serveur désiré.

A partir des valeurs publiques DH échangées, les deux entités génèrent automatiquement une clé session utilisée avec des clés dérivées pour le chiffrement des données. Une fois une certaine quantité de données est transmise au moyen d'une clé et d'un algorithme précis, il y'à un nouvel échange de clés. Ce qui produit un autre ensemble de valeurs repère et une autre valeur secrète partagée.

Une fois que la couche transport a créé un tunnel sécurisé pour envoyer les informations entre les deux systèmes, le serveur indique au client les différentes méthodes d'authentification prises en charge, telles que l'utilisation d'une signature chiffrée privée ou l'entrée d'un mot de passe. Le client doit ensuite essayer de s'authentifier au serveur au moyen d'une des méthodes spécifiées.

Etant donné que les serveurs peuvent être configurés de façon à permettre différents types d'authentification, cette méthode donne aux deux parties un niveau de contrôle optimal. Le serveur peut décider quelles méthodes d'authentification prendre en charge en fonction de son modèle de sécurité et le client peut choisir l'ordre des méthodes d'authentification à utiliser parmi celles qui sont disponibles. Grâce à la nature sécurisée de la couche authentification SSH, même les méthodes d'authentification qui, de prime abord, semblent non sécurisées telles que l'authentification d'ordinateur hôte, peuvent être utilisées en toute sécurité.

Finalement, vient le rôle de la couche connexion. Celle-ci ouvre plusieurs canaux afin de permettre à plusieurs sessions d'être ouvertes sur une seule connexion SSH.

La figure suivante illustre un exemple d'une connexion SSH [7].

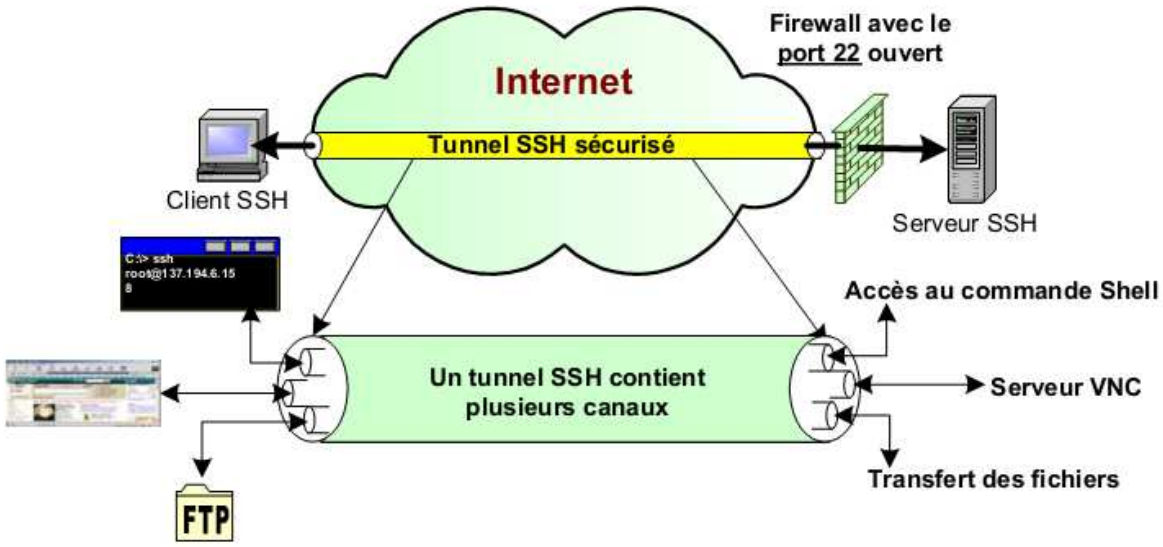


Figure 3.2 : Connexion SSH avec plusieurs canaux ouverts

3.2.5. Avantages :

- Etablissement de connexions sécurisées dans un réseau ouvert.
- La mise en place d'un tunnel sécurisé afin de faire passer d'autres types de protocoles.

3.2.6. Inconvénients :

- Possibilité d'une attaque du type Man In The Middle due au fait que la première connexion avec le serveur n'est pas authentifiée.
- Vu les découvertes multiples de failles dans la sécurité de SSH (la dernière en 2008 dans toutes les versions de SSH qui permet de récupérer jusqu'à 32 bits du texte clair d'un bloc crypté avec CBC), il est désormais devenu très vulnérable et des données sensibles ne peuvent être envoyées par son biais.

3.3. La couche SSL :

3.3.1. Définition :

SSL (Secure Sockets Layer), que l'on pourrait traduire par couche de sockets sécurisée est un procédé de sécurisation des transactions effectuées via Internet. Le standard SSL a été mis au point par Netscape, en collaboration avec Mastercard, Bank of America, MCI et Silicon Graphics. Il repose sur un procédé de cryptographie par clef publique afin de garantir la sécurité de la transmission de données sur internet.
Son principe consiste à établir un canal de communication sécurisé (chiffré) entre deux machines (un client et un serveur) après une étape d'authentification.
Le système SSL est indépendant du protocole utilisé, ce qui signifie qu'il peut aussi bien

sécuriser des transactions faites sur le Web par le protocole HTTP que des connexions via le protocole FTP, POP ou IMAP.

En effet, SSL agit telle une couche supplémentaire, permettant d'assurer la sécurité des données, située entre la couche application et la couche transport (protocole TCP par exemple). De cette manière, SSL est transparent pour l'utilisateur (entendez par là qu'il peut ignorer qu'il utilise SSL). Par exemple un utilisateur utilisant un navigateur internet pour se connecter à un site de commerce électronique sécurisé par SSL enverra des données chiffrées sans aucune manipulation nécessaire de sa part. La quasi-intégralité des navigateurs supporte désormais le protocole SSL.

Netscape Navigator affiche par exemple un cadenas verrouillé pour indiquer la connexion à un site sécurisé par SSL et un cadenas ouvert dans le cas contraire, tandis que Microsoft Internet Explorer affiche un cadenas uniquement lors de la connexion à un site sécurisé par SSL.

sous Internet Explorer sous Mozilla

Figure 3.3 : Accès sécurisé à un site web

Un serveur web sécurisé par SSL possède une URL commençant par https://, où le "s" signifie bien évidemment Secured (sécurisé).

3.3.2. Historique :

La première version de SSL a été développée par Netscape Communications en 1994. L'objectif de Netscape était de créer un canal sécurisé où les données pourraient transiter entre un client et un serveur, indépendamment de la plateforme et du système d'exploitation. Netscape souhaitait aussi pouvoir bénéficier des nouvelles méthodes de chiffrage, telles AES (Advanced Encryption Standard), qui venait de remplacer le chiffrement DES (Data Encryption Standard).

Quoique SSL soit destiné à l'origine uniquement pour sécuriser les transactions entre un client et un serveur web (HTTP), la spécification a été connue de façon à ce que les autres

protocoles de niveau application puissent l'exploiter (FTP, Telnet, etc.). La spécification SSL 1.0 ne fut diffusée qu'en interne et aucune application ne supportera SSL 1.0.

Quelques mois plus tard, en février 1995, Netscape publie la version 2.0 du protocole. Il est implémenté dans la première version de son client web Navigator. SSL 2.0 se fonde sur l'authentification du serveur par le poste client et sur l'utilisation d'un certificat serveur au format X.509 v3. Cette authentification ne nécessite, du côté du poste client, que des calculs en clé publique.

En novembre 1996, Netscape publie la version 3.0 du protocole. Par rapport SSL 2.0, SSL 3.0 offre en plus la capacité, pour le serveur, d'authentifier le client. Dans ce cas, le client doit pouvoir d'une part exploiter sa clé privée et d'autre part fournir son certificat au format X.509 v3.

L'IETF (Internet Engineering Task Force) propose à son tour un protocole de transfert sécurisé basé sur les concepts de SSL, baptisé TLS 1.0 (parfois nommée SSL 3.1) et décrit dans la RFC 2246. Elle rachète le brevet de Netscape sur le protocole SLL en 2001. Puis en juin 2003, des extensions sont proposées pour TLS sous la forme d'une nouvelle RFC. La dernière RFC 4366, mise à jour de la précédente, elle est sortie en Avril 2006 et ensuite TLSv1.2, fut décrite par la RFC 5246 et publiée en 2008.

A l'heure actuelle, les protocoles SSL 2.0, SSL 3.0 et TLS 1.0 sont utilisés par la plupart des navigateurs Web. La version 2.0 de SSL présente cependant des failles de sécurité, ce qui représente une contre-indication à son emploi.

3.3.3. Sous-couches composant SSL :

La couche SSL se subdivise en quatre sous couches (sous protocoles) :

- **SSL Handshake :** Définit le format qui sera utilisé pour l'échange des données.
- **Le SSL Record Protocol :** Se charge des différents échanges de messages entre le client et le serveur.
- **Le SSL Alert :** Ce protocole spécifie les messages d'erreur que peuvent s'envoyer clients et serveurs.

- **Le SSL Change Cipher Spec :** Son but est d'activer pour la session SSL courante les algorithmes, les clés et les nombres aléatoires négociés durant la phase d'initialisation (la phase Handshake). Ceci en passant ces informations au protocole SSL Record. Les protocoles formant SSL sont illustrés dans la figure qui suit [7] :

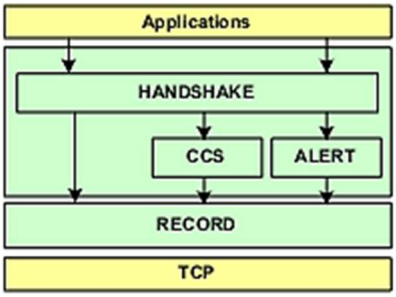


Figure 3.4 : Emplacement de la couche SSL dans le modèle OSI

3.3.4. Principe de fonctionnement:

Le principe de fonctionnement de SSL se présente comme suit :

- **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.

3.3.5. Fonctionnement du Handshake :

Dans le protocole SSL, tout se joue au niveau du sous-protocole SSL Handshake. Il permet l'échange des paramètres de sécurité (nombres aléatoires, liste des algorithmes de chiffrement, de hachage, etc.) entre le client et le serveur avant que les données applicatives

ne soient transmises. Il permet aussi l'authentification des deux communicateurs. Cependant, dans la plupart des cas, le serveur est uniquement authentifié. Ce protocole peut opérer en deux formes : soit il établit un échange complet avec la négociation des paramètres de sécurité (le Handshake complet), soit il essaye d'utiliser une ancienne session SSL/TLS déjà négociée (Handshake abrégé).

- **Le Handshake Complet :**

La figure 3 illustre l'échange initial nécessaire pour établir une connexion complète entre le client et le serveur. Le client commence l'échange SSL/TLS en envoyant le message Client Hello qui contient un nombre aléatoire (R1), un identificateur de session (S_ID), une liste des algorithmes de compression (compression List) et une suite de chiffrement (cipher_list). Notons que le nombre aléatoire est une concaténation entre le temps système en format Unix et un nombre aléatoire.

Le serveur répond en envoyant le message Server Hello contenant un nombre aléatoire (R2), un identificateur non nul de session et une suite choisie des algorithmes de chiffrement et de hachage. Le serveur envoie également un certificat X.509 contenant sa clé publique pour s'authentifier. Ensuite, le client vérifie la clé publique du serveur et génère un nombre aléatoire sur 48 octets connu sous le nom du pre_master_secret. Le client chiffre le pre_master_secret avec la clé publique du serveur et l'envoie dans le message ClientKeyExchange.

A la réception de ces messages, le serveur déchiffre le pre_master_secret en utilisant sa clé privée. C'est à ce moment-là que le client et le serveur peuvent calculer un secret partagé, le master secret, à partir des nombres aléatoires R1 et R2 et du pre_master_secret. Ce secret servira par la suite à la dérivation des clés de chiffrement et de déchiffrement dans les connexions SSL/TLS. Le dernier échange achève l'instauration d'une connexion sûre. Les deux entités échangent les messages Finished contenant le hachage de l'ensemble des messages et des paramètres négociés.

Si le certificat client est exigé par le serveur (le serveur envoie le message Certificate Request), alors le client répond en envoyant le message Certificate Verify contenant sa signature sur toutes les informations échangées avec le serveur. Même si l'authentification par certificat X.509 et par clés RSA reste la plus utilisée pour l'authentification des serveurs SSL/TLS, ces derniers peuvent avoir d'autres méthodes d'authentification telles que

l'authentification par des valeurs DH Anonymes (un nombre premier, un nombre qui lui est relativement premier et la clé publique DH du serveur), des valeurs DH temporaires (les paramètres DH et une signature), un échange RSA (où le serveur n'a qu'une clé RSA pour signer et établir une clé RSA temporaire) ou finalement par Fortezza (norme de sécurité du gouvernement américain) [2]. Parmi toutes ces méthodes, l'échange du DH temporaire reste le plus sûr. De plus, il assure le service de PFS sur les données échangées. Toutes ces valeurs sont envoyées à travers les deux messages Client_Key_Exhange et Server_Key_Exchange envoyés respectivement avant les messages ChangeCipherSpec et server_Hello_Done.**[7]**

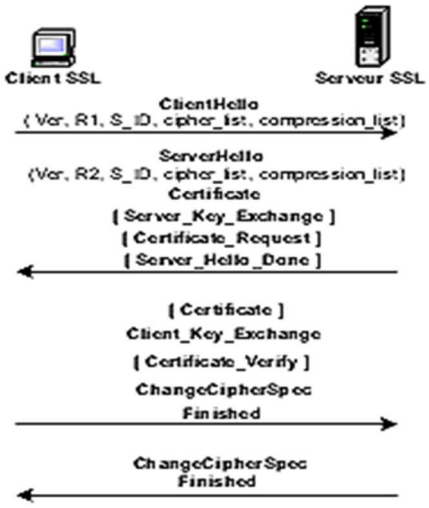


Figure 3.5 : Illustration du Handshake complet

3.4.1. Avantages :

3.4.1.1. Authentification et intégrité fortes des messages :

Le dispositif primaire de SSL/TLS est la capacité de sécuriser les flux de données transitant sur des réseaux publics, tout en utilisant des algorithmes de chiffrement symétrique. SSL/TLS offre également l'authentification de serveur et sur option, l'authentification du client pour prouver les identités des deux parties. Il fournit également l'intégrité de données par une valeur de contrôle d'intégrité. En plus de la protection contre la révélation de données par le chiffrage, le protocole de sécurité SSL/TLS peut être employé pour se protéger contre les attaques de type man-in-the-middle, re-jeux des paquets et attaques de changement ou baisse des versions SSL/TLS.

3.4.1.2. Interopérabilité et facilité de déploiement :

SSL/TLS fonctionne avec la plupart des navigateurs Web, y compris le navigateur de Microsoft (Internet Explorer) et de Netscape (Netscape Navigator) et sur la plupart des serveurs Web inclus dans les différents systèmes d'exploitation. En outre, il est souvent intégré avec les serveurs LDAP et une variété d'autres applications.

3.4.1.3. Facilité d'utilisation :

Puisque SSL/TLS est mis en application sous la couche application, la plupart de ses opérations sont complètement invisibles au client. Ceci permet au client d'avoir peu de connaissances sur ses communications sécurisées et d'être toujours protégé contre les attaquants.

3.4.1.4. SSL VPN :

Le terme SSL VPN désigne simplement une encapsulation du trafic d'une application particulière (HTTP ou IMAP par exemple) par SSL/TLS afin d'atteindre les ressources informatiques de l'entreprise. Le SSL VPN ne concurrence pas directement les VPNs IPSec. Les constructeurs fournissant des SSL VPN les positionnent comme une solution complémentaire à IPSec dans le but de répondre aux nouveaux besoins des entreprises. Le SSL VPN est une solution qui palie aux difficultés rencontrées avec IPSec et les accès distants (notamment dus à la translation d'adresse et au filtrage d'application dans le réseau distant). Le principal avantage est de permettre aux utilisateurs nomades (télétravailleurs par exemple) de pouvoir se connecter au réseau de leur entreprise depuis n'importe quel ordinateur (PDA, cybercafé, ré seau protégé par un Firewall la issant passé les flux HTTP …) sans modification à apporter sur le poste distant. Ainsi, on parle souvent de VPN Clientless [9] : il n'y a pas besoin d'installer de logiciel spécifique sur le client pour atteindre le réseau de l'entreprise, contrairement au VPN IPSec. Ceci est possible avec des sessions HTTPS car SSL/TLS est aujourd'hui répandu sur tous les systèmes d'exploitation et supporté par n'importe quel navigateur Web.

3.4.2. Inconvénients :

3.4.2.1. Problèmes liés au client SSL : le navigateur

Les navigateurs n'ont pas encore de fonctionnalités évoluées de gestion des clés, les certificats ne peuvent par exemple pas être automatiquement renouvelés et l'historique des

clés n'est pas conservé. Quand un certificat expire, l'utilisateur reçoit un message et doit obtenir manuellement un nouveau certificat, ce qui n'est pas forcément trivial pour un utilisateur lambda. Ces problèmes seront probablement résolus lorsque les navigateurs deviendront des clients de PKI à part entière.

La cryptographie utilisée par les navigateurs peut être soumise à des restrictions d'exportation (hors des Etats-Unis par exemple pour Netscape et Microsoft) cela force les utilisateurs à utiliser des clés de longueur insuffisante. Dans le cas de Netscape, on peut renforcer la crypto chez www.fortify.com par exemple... tout en veillant à rester en conformité avec la législation... (On peut aussi y voir quel est la crypto utilisée par son navigateur)

La relation de confiance est définie par la liste préinstallée des autorités de certification.

Les navigateurs du commerce sont livrés avec de nombreuses clés publiques préinstallées (Netscape en contient 33). Celles-ci sont utilisées pour la vérification de la signature de l'autorité de certification pour les certificats d'autres navigateurs ou serveurs. Pour être confirmé, un certificat doit être signé par n' importe lequel des AC présentes dans le navigateur. Par conséquent, si l'une quelconque parmi les autorités de certification certifie un site frauduleux, ce certificat sera vérifié correctement par des millions de navigateurs. En tant qu'utilisateur, il est important d'aller voir dans les propriétés «sécurité» de son navigateur et de valider la confiance que l'on attribue aux diverses autorités de certification. Par défaut, les navigateurs font confiance à toutes...

3.4.2.2. Les mots de passe :

Sur MSIE (jusqu'à la version 5.0) le mot de passe protégeant les certificats est optionnel alors que protéger sa clé privée est vital. Avant d'obtenir un certificat (qui pourra être utilisé pour SSL-TLS), vous devez générer une bi-clé. Vous enverrez la clé publique à une AC qui en fera un certificat. Mais vous conservez la clé privée, qui vous perm et de signer, et de déchiffrer. Cette clé se doit d'être protégée (puisqu' elle est sensible). Or les navigateurs (dont Internet Explorer i.e. MSIE), qui sont les "clients" SSL-TLS les plus répandus n'obligent pas les utilisateurs à spécifier un mot de passe pour leurs clés privées. Et de fait, beaucoup d'utilisateurs n'en mettent pas. Ce qui peut constituer une grosse lacune de sécurité.

3.4.2.3. Problèmes intrinsèques au protocole :

Le système est fondé sur une seule paire de clés. Le principal problème que cela pose vient de la différence entre les contraintes qui pèsent sur les clés de signature/authentification et celles de chiffrement. Avoir la possibilité de sauvegarder en central les clés de chiffrement peut s'avérer très pratique (si un utilisateur oublie son mot de passe il n'est par exemple pas nécessaire de régénérer une paire de clés et de la certifier à nouveau, mais on peut simplement lui renvoyer ses clés, en l'obligeant à ressaisir un mot de passe). Si les services d'un tiers sont utilisés pour la sauvegarde d'une unique paire de clés (service offert par certaines autorités de certification ou Tiers de confiance), le client ne sera plus le seul à avoir accès à sa clé privée de signature et la non-répudiation n'est alors plus

Certes, gérer deux paires de clés est plus lourd que d'en gérer une seule. La sauvegarde des clés est importante là où celles-ci sont utilisées pour chiffrer des données stockées, comme des courriers électroniques par exemple. Si les clés ne sont pas sauvegardées et que l'accès aux clés est perdu, les données chiffrées n'ont plus aucune utilité (et si en essayant d'en briser la protection on les récupère quand même, c'est alors l'application crypto qui n'est d'aucune utilité). Ce défaut est un peu théorique parce que SSL, dans la pratique, ne chiffre que des flux. Pour un utilisateur "isolé", il est au contraire souhaitable qu'aucune de ses clés privées ne soit ailleurs qu'en sa possession. Mais il est probable que dans l'avenir un même certificat servira à plusieurs types d'opérations cryptographiques, auquel cas ce problème se posera.

Le protocole SSL ne prévoit pas de vérification systématique des CRL. Lorsqu'un serveur Web présente un certificat, le navigateur en vérifie sa validité ; cela consiste pour lui à :

- Vérifier que les dates de validité sont valides
- Vérifier que la signature appliquée au certificat est valide

Mais le protocole SSL n'impose pas qu'un certificat ne soit utilisé que suite à la consultation de la CRL qui lui correspond. Un serveur Web peut donc présenter aux navigateurs un certificat révoqué. Netscape 6 permet de vérifier automatiquement les CRL, grâce au protocole OCSP (Online Certificate Status Protocol, désactivé par défaut) La vérification manuelle des CRL est laborieuse et quasiment jamais faite. [9]

3.5. Conclusion :

SSH et SSL sont des couches de protocoles, qui s'entremettent dans certains niveaux du modèle OSI afin de fournir des fonctionnalités additionnelles de sécurisation utilisant une multitude d'algorithmes cryptographiques, symétriques ou asymétriques, que le modèle OSI ne peut pas fournir par défaut.

Chapitre 4 :
Sécurisation d'une communication dans un réseau d'ordinateurs

4.1. Introduction :

Afin de démontrer l'importance de la sécurisation des communications dans les réseaux d'ordinateurs, un exemple pratique devait être fait.

Pour cela, on a simulé une communication dans un réseau d'ordinateurs par un programme qui sert à envoyer un message d'un ordinateur à un autre via le réseau local. Pour simuler la sécurisation, on a intégré dans ce programme un algorithme de cryptographie afin de crypter le message à l'envoi et de le décrypter à la réception.

Pour simuler une attaque, on a utilisé un logiciel de capture de trames afin de capter la transmission de la donnée dans le réseau et intercepter le message envoyé.

Pour cela on a mis en place le programme avec le langage C++ et installé le logiciel de capture des trames WIRESHARK.

- Pour la programmation en C++, on a utilisé DEV-C++
- Pour la capture des trames, on a utilisé le logiciel WIRESHARK :

WIRESHARK est un analyseur de paquets réseau. Un analyseur de paquets réseau va essayer de capturer les paquets réseau afin d'afficher les données que ces derniers contiennent.

4.2. Mise en place du programme :

Le programme est basé sur l'envoi de messages d'un ordinateur à un autre en utilisant les sockets. Un socket nous permet d'ouvrir un port sur l'ordinateur afin d'envoyer ou de recevoir des données dans un réseau, ce port restera ouvert jusqu'à la fin de la transmission. Ceci nous permet d'échanger les données soit en mode connecté (protocole TCP) ou non connecté (protocole UDP). Dans cet exemple, on a utilisé le mode connecté.

Dans notre programme, on a mis en place un serveur qui démarre le socket et attend que le client se connecte. Une fois connecté, le client nous demande d'entrer le message à envoyer. Ce message est ensuite reçu par le serveur qui va l'afficher pour ensuite fermer le socket et ainsi la connexion.

4.3. Algorithme de sécurisation :

4.3.1. Algorithme de cryptage:

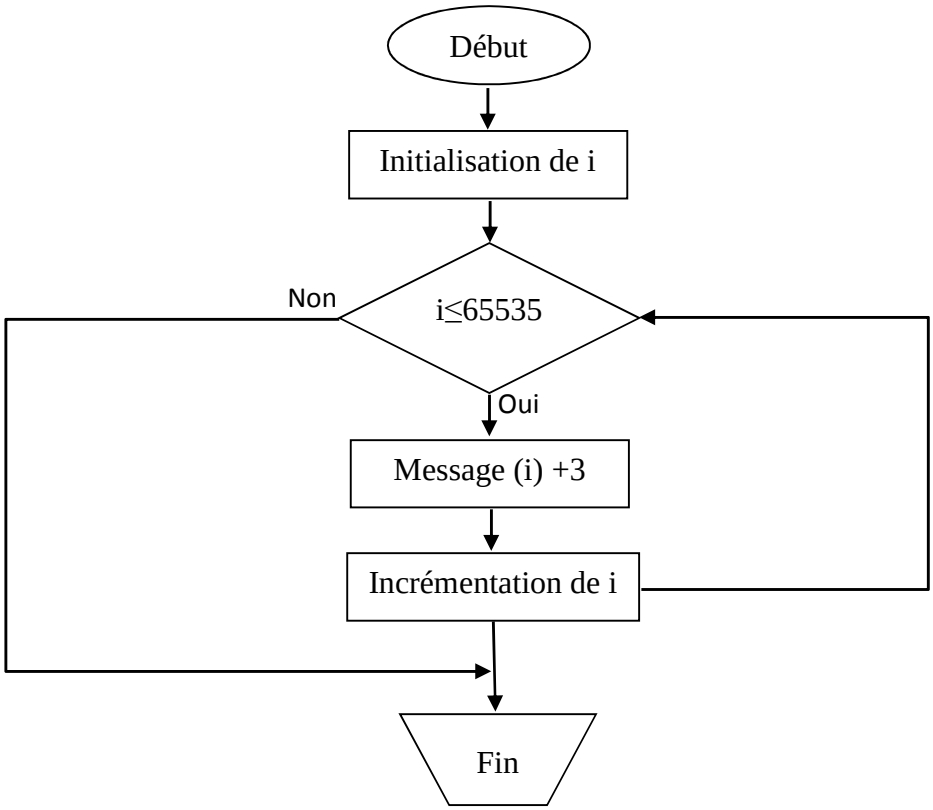


Figure 4.1 : Algorithme de cryptage de César

4.3.2. Algorithme de décryptage:

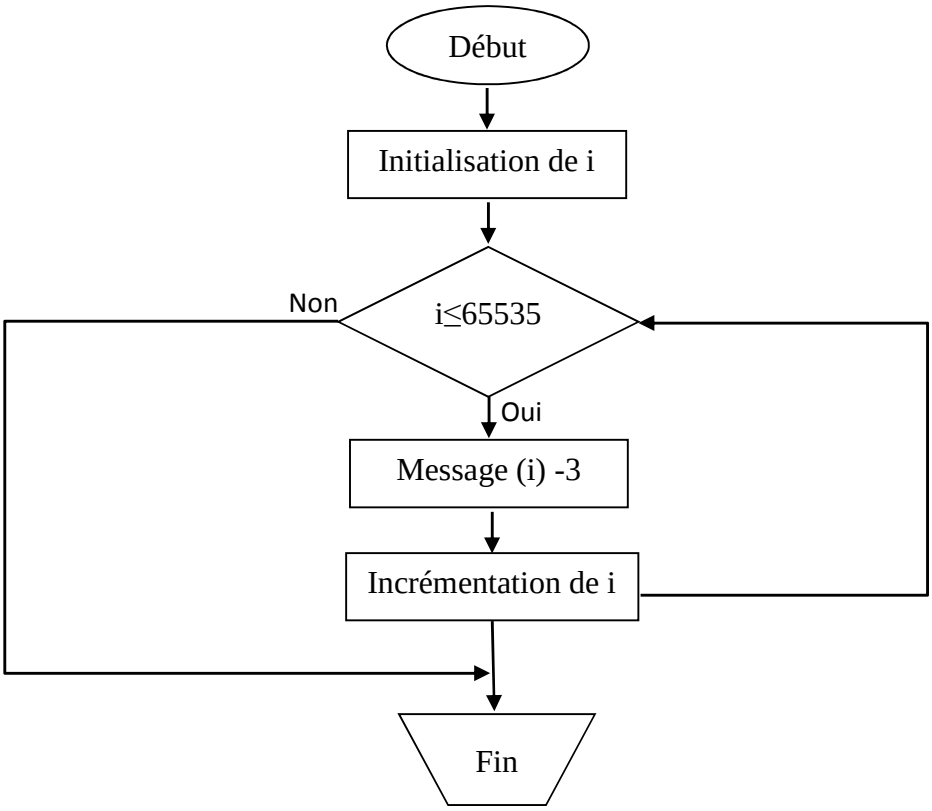


Figure 4.2 : Algorithme de décryptage de César

4.4. Cas non sécurisé :

Dans ce cas, lorsque l'on entre le message à envoyer dans le client, celui-ci l'envoie directement vers le serveur (en clair).

4.4.1. Algorithme du côté client :

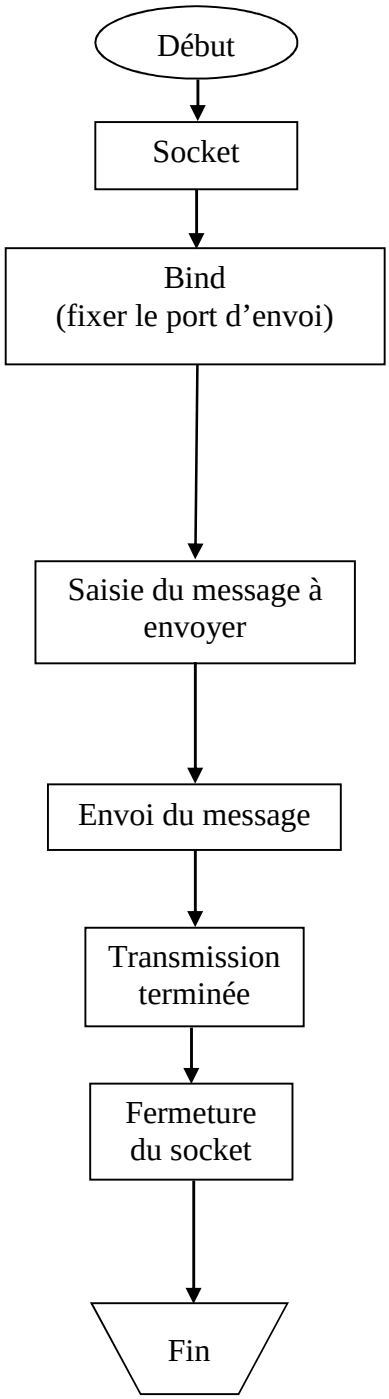


Figure 4.3 : Organigramme de l'envoi non sécurisé des données

4.4.2. Algorithme du côté serveur :

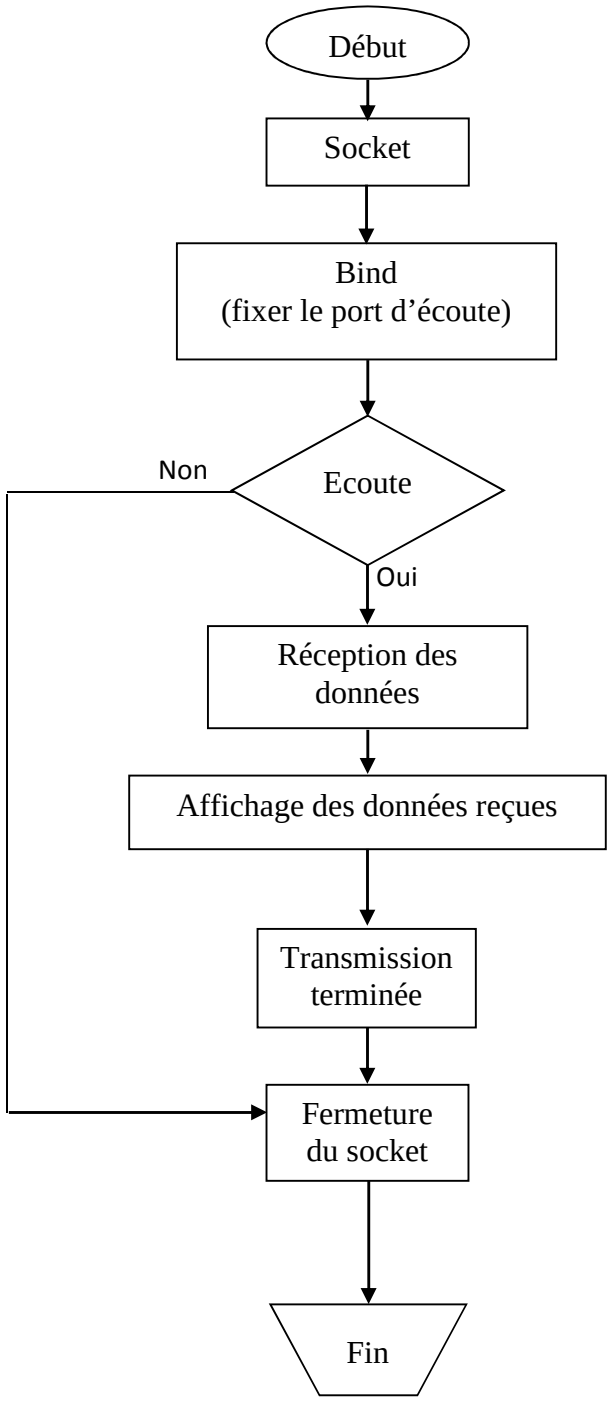


Figure 4.4 : Organigramme de la réception des données (cas non sécurisé)

4.5. Cas sécurisé :

Dans ce cas, une fois entré le message à envoyer, celui-ci passe d'abord par un algorithme afin qu'il soit crypté. Ensuite, il va être envoyé au serveur qui, de son côté va faire passer le message reçu (crypté) par un algorithme de décryptage afin d'obtenir le message clair.

4.5.1. Algorithme du côté client :

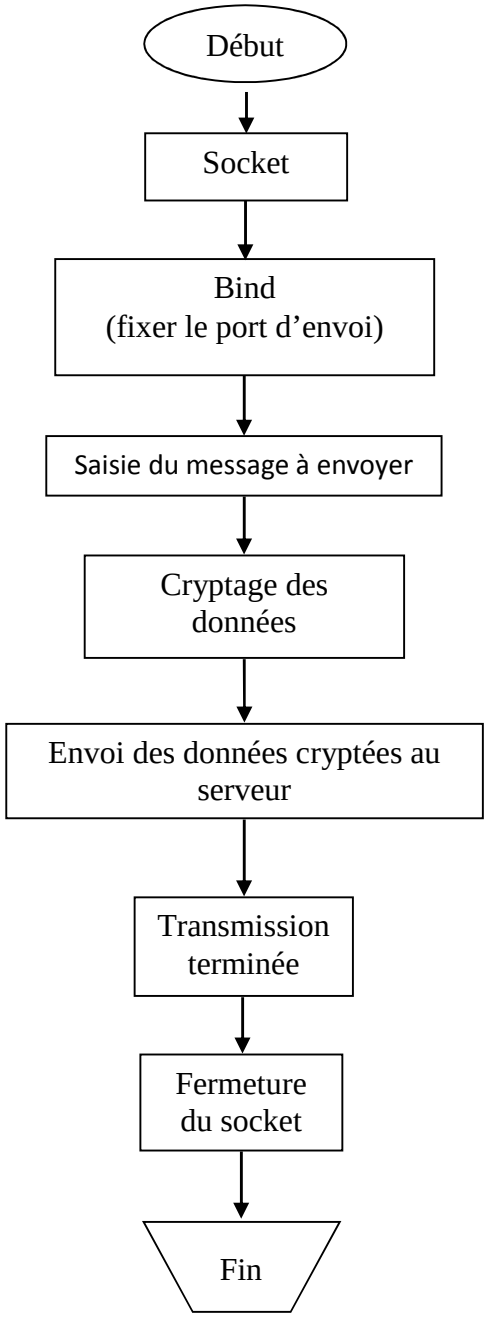


Figure 4.5 : Organigramme de l'envoi sécurisé des données

4.5.2. Algorithme du côté serveur :

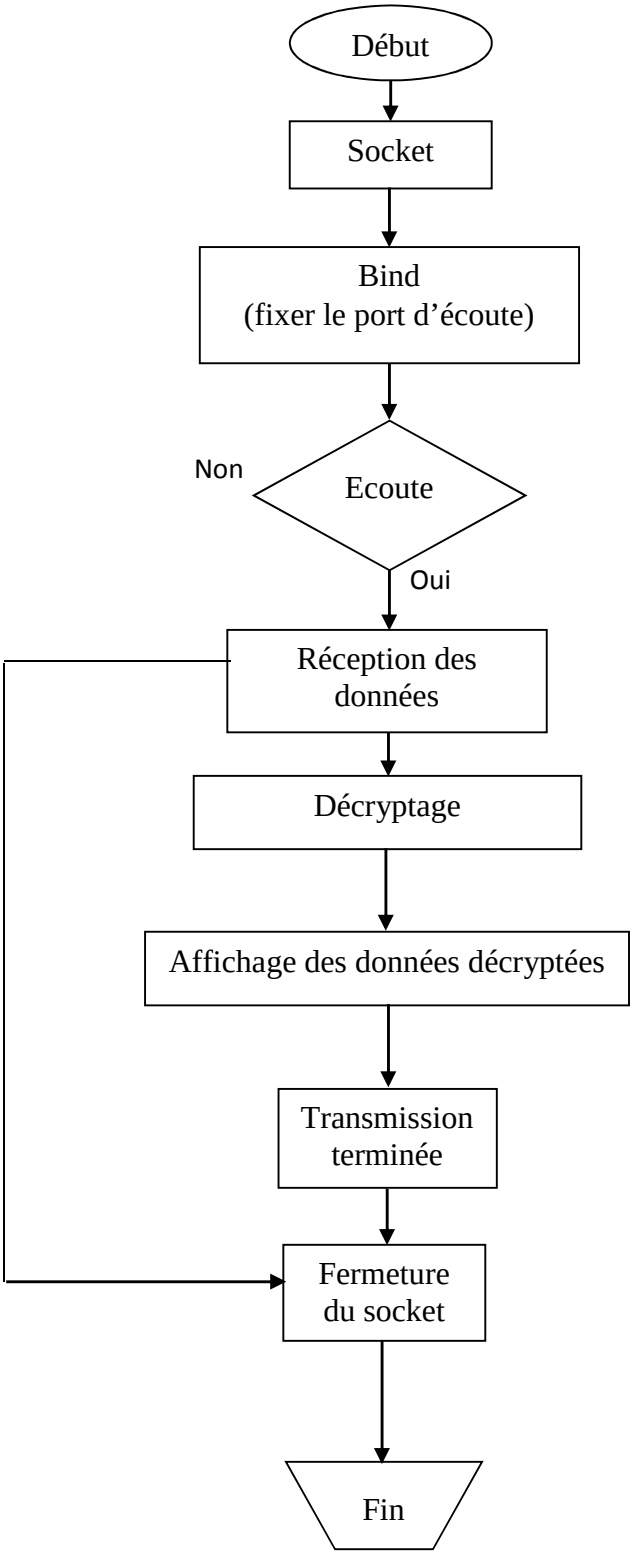


Figure 4.6 : Organigramme de la réception des données (cas sécurisé)

4.6. Exécution du programme :

4.6.1. Cas non sécurisé :

4.6.1.1. Envoi du message (coté client) :

Figure 4.7 : Envoi du message à partir du client

4.6.1.2. Réception du message (coté serveur) :

Figure 4.8 : Réception des données au niveau du serveur

4.6.1.3. Capture de la trame :

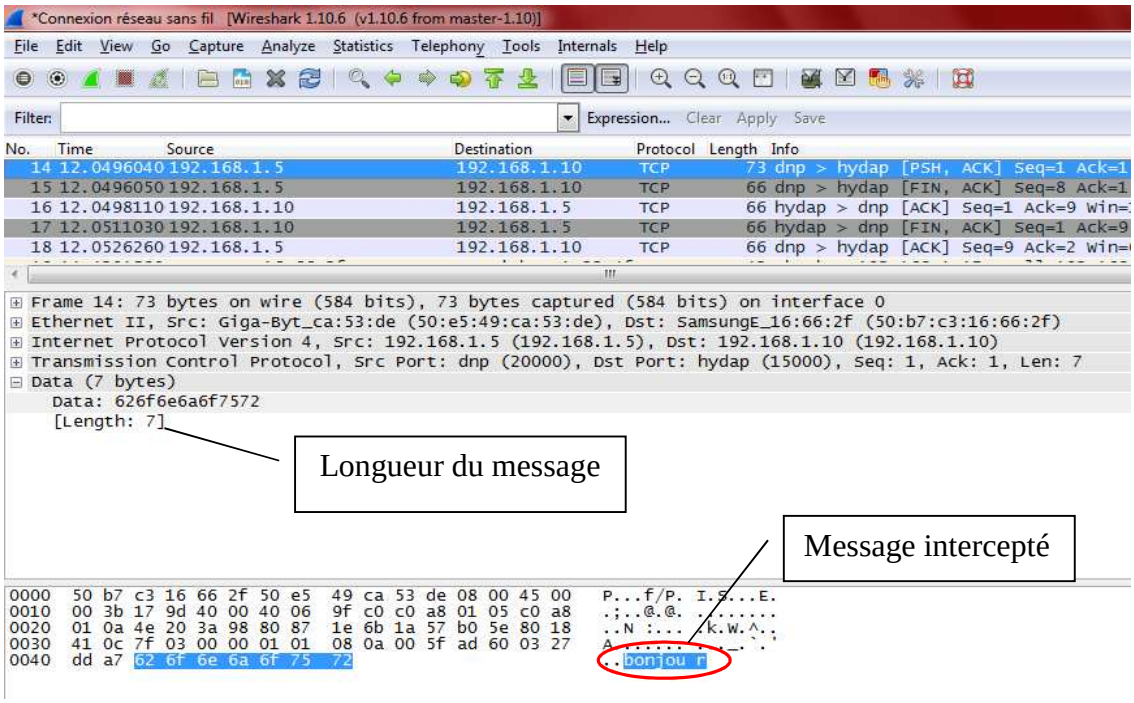


Figure 4.9 : Capture du paquet contenant le message envoyé

4.6.2. Envoi sécurisé du message :

4.6.2.1. Envoi du message (coté client) :

Figure 4.10 : Envoi des données cryptées

Afin de vérifier que le message crypté envoyé est le même qu'à la réception, le message crypté est affiché.

4.6.2.2. Réception du message (coté serveur) :

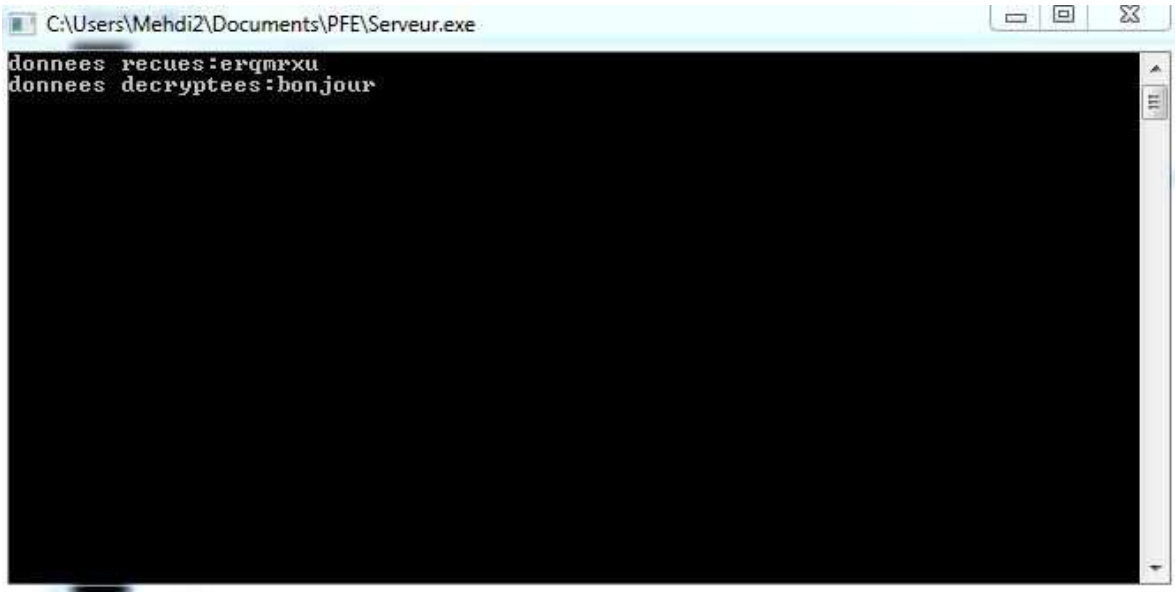


Figure 4.11 : Réception des données (cas sécurisé)

Afin de confirmer que le message reçu est le même envoyé, on a affiché le message crypté et ensuite on a décrypté celui-ci pour obtenir le message original.

4.6.2.3. Capture de la trame :

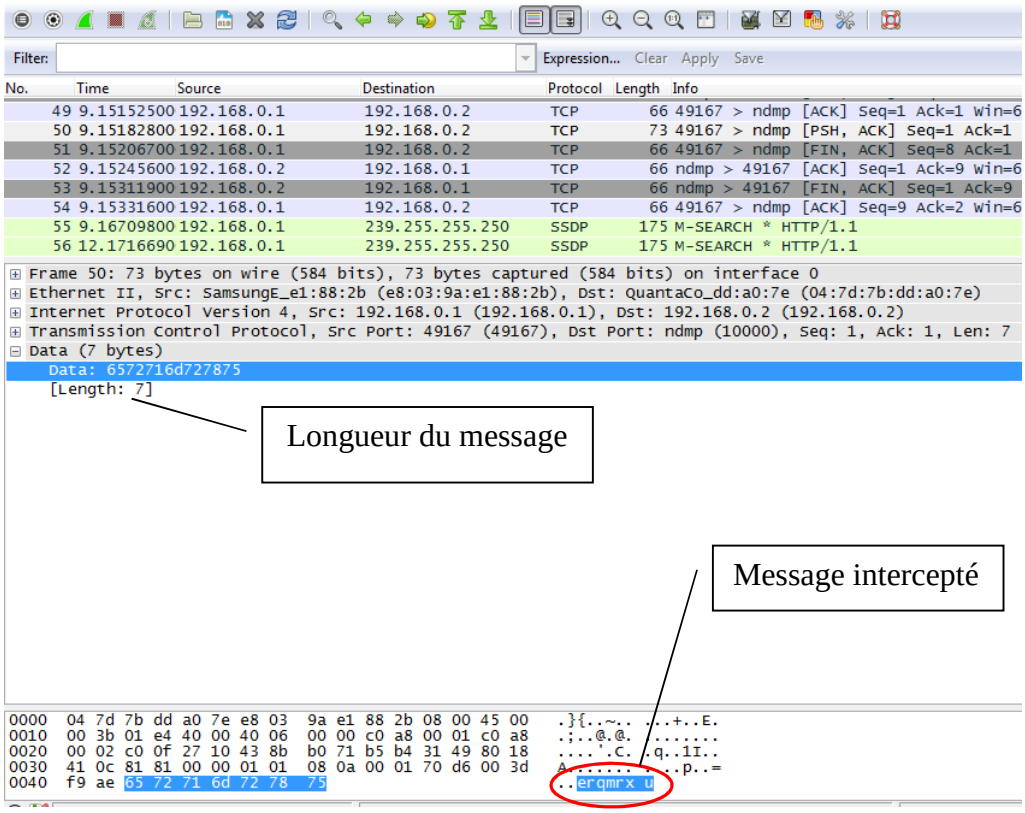


Figure 4.12 : Capture du paquet contenant le message envoyé (cas sécurisé)

Dans le cas sécurisé, on a pu capturer le paquet contenant le message mais celui-ci est incompréhensible (crypté), on doit donc justifier de la clé de cryptage pour comprendre (décrypter) le contenu du message.

4.7. Conclusion :

Si la communication n'est pas sécurisée, les données vont circuler en clair dans le réseau. Par conséquent, l'interception de la transmission nous permettra d'accéder directement aux informations contenues dans ces données.

Quand la communication est sécurisée, avant d'être envoyées dans le réseau, les données sont d'abord cryptées.

Au cas où la transmission est interceptée, on ne pourra obtenir les informations à partir des données que si l'on justifie de la clé qui nous permettra de les décrypter.

Conclusion générale

Conclusion générale :

L'étude de la sécurisation des communications dans les réseaux d'ordinateurs est l'objectif du travail qu'on a effectué. La sécurisation est nécessaire dans les réseaux d'ordinateurs, elle permet l'échange d'informations et de données même dans un réseau ouvert tels qu'internet.

Ce projet nous a permis de comprendre le réseau et son fonctionnement.

Il nous a aussi donné une idée sur la façon dont l'information circule entre les ordinateurs dans le réseau.

On a pu voir des méthodes de cryptographie symétriques et asymétriques utilisées dans le cryptage des données.

On a pu connaitre les méthodes de sécurisation les plus courantes et leur fonctionnement.

Enfin, on a mis en place un programme par lequel on a constaté l'importance de la sécurisation dans les réseaux d'ordinateurs.

On souhaite améliorer ce travail et mettre en place un programme qui pourra établir une connexion sécurisée avec SSL afin de garantir la sécurité des transferts de données.

Bibliographie

Bibliographie

[1] www.frame-ip.com/osi, 09/05/2014.

[2] Open Systems Networking TCP/IP and OSI, David M.Piscitello and A.Lyman Chapin, “table 2.2 cross reference of ISO/CCITT OSI standards”, p22

[3] ser-info-02.ec-nantes.fr/users/info3/weblog/1c2f7/Histoire_de_la_cryptographie.html, 09/05/2014

[4] http://www.cryptage.org, 09/05/2014

[5] http://wakaziva.pagesperso-orange.fr/crypto/2.htm, 07/05/2014

[6] http://www.bibmath.net/crypto/index.php?action=affiche&quoi=moderne/md5, 18/04/2014

[7] http://en.wikipedia.org/wiki/Secure_Shell, 11/03/2014

[8] Nouvelles technologies réseaux SSH – SSL – TLS, Stéphane Brinster, Guillaume Lecomte et Aymeric Bernard

[9] Thèse doctorat Ibrahim HAJJEH : Sécurité des échanges. Conception et validation d’un nouveau protocole pour la sécurisation des échanges. Soutenue le 7 décembre 2004, Ecole nationale supérieure des télécommunications, Paris

Annexe

Annexe

Programme C++ :

Cas sécurisé :

Côté client :

```
#include <winsock2.h>
#include <cstdio>
#include<conio.h>
#include<string.h>
#include <iostream>
#include <climits>
#include <cstdlib>
#include <conio.h>
#pragma comment(lib,"ws2_32.lib")
using namespace std;
inline int modulo (int m, int n)
{
   return m >= 0 ? m % n : ( n - abs ( m % n ) ) % n;
}
WSADATA initialisation_win32;
int tempo;
char buffer[65535];
char str[65535];
SOCKET identifiant;
SOCKADDR_IN information_sur_la_destination;
SOCKADDR_IN sa_loc;
main (int argc, char* argv[])
{
printf("\nBonjour, veuillez entrer votre message\n");
          string message;
          getline(cin, message);
   for (string::iterator it = message.begin(); it < message.end(); ++it) {
      if (isupper(*it)) {

             *it = 'A' + modulo(*it - 'A' + 3, 26);
      }
      else if (islower(*it)) {

             *it = 'a' + modulo(*it - 'a' + 3, 26);
      }
   }
   strcpy(str, message.c_str());
   cout <<  message << endl;
          WSAStartup(MAKEWORD(2,2),&initialisation_win32);
          identifiant=socket(AF_INET,SOCK_STREAM,0);
          tempo=1;
          setsockopt(identifiant,IPPROTO_TCP,TCP_NODELAY,(char
*)&tempo,sizeof(tempo));
```

```
        sa_loc.sin_family = AF_INET;
   sa_loc.sin_port = htons(20000);
   sa_loc.sin_addr.s_addr = inet_addr("192.168.1.3");
   bind(identifiant, (struct sockaddr *)&sa_loc, sizeof(struct sockaddr));
        information_sur_la_destination.sin_family=AF_INET;
        information_sur_la_destination.sin_addr.s_addr=inet_addr("197.205.127.129");
        information_sur_la_destination.sin_port=htons(15000);
        connect(identifiant,(struct
sockaddr*)&information_sur_la_destination,sizeof(information_sur_la_destination));
        strcpy(buffer,str);
        send(identifiant,buffer,strlen(buffer),0);
 shutdown(identifiant,2);
                closesocket(identifiant);
          WSACleanup();
                getch();
        }
```

Côté serveur :

```
#include <winsock2.h>
#include <cstdio>
#include<conio.h>
#include <string>
#include <climits>
#include <cstdlib>
#include <iostream>
#include <sstream>
#pragma comment(lib,"ws2_32.lib")
using namespace std;
inline int modulo (int m, int n)
{
   return m >= 0 ? m % n : ( n - abs ( m % n ) ) % n;
}
WSADATA initialisation_win32;
int erreur;
int tempo;
int nombre_de_caractere;
char buffer[65535];
char str[65535];
SOCKET id_de_la_socket;
SOCKET id_de_la_nouvelle_socket;
SOCKADDR_IN information_sur_la_source;
int main (int argc, char* argv[])
        {
   stringstream s;
   string message;
   WSAStartup(MAKEWORD(2,2),&initialisation_win32);
        id_de_la_socket=socket(AF_INET,SOCK_STREAM,0);
        tempo=1;
```

```
        setsockopt(id_de_la_socket,IPPROTO_TCP,TCP_NODELAY,(char
*)&tempo,sizeof(tempo));
        information_sur_la_source.sin_family=AF_INET;
        information_sur_la_source.sin_addr.s_addr=INADDR_ANY;
        information_sur_la_source.sin_port=htons(10000);
        erreur=bind(id_de_la_socket,(struct
sockaddr*)&information_sur_la_source,sizeof(information_sur_la_source));
        erreur=99;
        while(erreur!=0)
                erreur=listen(id_de_la_socket,1);
        tempo=sizeof(information_sur_la_source);
        id_de_la_nouvelle_socket=accept(id_de_la_socket,(struct
sockaddr*)&information_sur_la_source,&tempo);
        nombre_de_caractere=recv(id_de_la_nouvelle_socket,buffer,1515,0);
        buffer[nombre_de_caractere]=0;
        s << buffer;
   s >> message;
   for (string::iterator it = message.begin(); it < message.end(); ++it) {
      if (isupper(*it)) {

            *it = 'A' + modulo(*it - 'A' - 3, 26);
      }
      else if (islower(*it)) {

            *it = 'a' + modulo(*it - 'a' - 3, 26);
      }
   }
   strcpy(str, message.c_str());
   cout<<"donnees recues:"<<str<<endl;
   cout<<"donnees decryptees:"<<<<endl;
        shutdown(id_de_la_nouvelle_socket,2);
        closesocket(id_de_la_nouvelle_socket);
        closesocket(id_de_la_socket);
        WSACleanup();
        getch();
        }
```

Cas non sécurisé :

Côté client :

```
#include <winsock2.h>
#include <cstdio>
#include<conio.h>
#include<string.h>
#include <iostream>
#include <climits>
#include <cstdlib>
#include <conio.h>
#pragma comment(lib,"ws2_32.lib")
using namespace std;
WSADATA initialisation_win32;
int tempo;
char buffer[65535];
char str[65535];
SOCKET identifiant;
SOCKADDR_IN information_sur_la_destination;
SOCKADDR_IN sa_loc;
main (int argc, char* argv[])
{
printf("\nBonjour, veuillez entrer votre message\n");
        string message;
        getline(cin, message);
strcpy(str, message.c_str());
WSAStartup(MAKEWORD(2,2),&initialisation_win32);
        identifiant=socket(AF_INET,SOCK_STREAM,0);
        tempo=1;
        setsockopt(identifiant,IPPROTO_TCP,TCP_NODELAY,(char
*)&tempo,sizeof(tempo));
        sa_loc.sin_family = AF_INET;
   sa_loc.sin_port = htons(20000);
   sa_loc.sin_addr.s_addr = inet_addr("192.168.1.3");
   bind(identifiant, (struct sockaddr *)&sa_loc, sizeof(struct sockaddr));
        information_sur_la_destination.sin_family=AF_INET;
        information_sur_la_destination.sin_addr.s_addr=inet_addr("197.205.127.129");
        information_sur_la_destination.sin_port=htons(15000);
        connect(identifiant,(struct
sockaddr*)&information_sur_la_destination,sizeof(information_sur_la_destination));
        strcpy(buffer,str);
        send(identifiant,buffer,strlen(buffer),0);
  shutdown(identifiant,2);
                closesocket(identifiant);
            WSACleanup();
                getch();
        }
```

Côté serveur :

```
#include <winsock2.h>
#include <cstdio>
#include<conio.h>
#include <string>
#include <climits>
#include <cstdlib>
#include <iostream>
#include <sstream>
#pragma comment(lib,"ws2_32.lib")
using namespace std;
WSADATA initialisation_win32;
int erreur;
int tempo;
int nombre_de_caractere;
char buffer[65535];
SOCKET id_de_la_socket;
SOCKET id_de_la_nouvelle_socket;
SOCKADDR_IN information_sur_la_source;
int main (int argc, char* argv[])
	{
WSAStartup(MAKEWORD(2,2),&initialisation_win32);
	id_de_la_socket=socket(AF_INET,SOCK_STREAM,0);
	tempo=1;
	setsockopt(id_de_la_socket,IPPROTO_TCP,TCP_NODELAY,(char
*)&tempo,sizeof(tempo));
	information_sur_la_source.sin_family=AF_INET;
	information_sur_la_source.sin_addr.s_addr=INADDR_ANY;
	information_sur_la_source.sin_port=htons(10000);
	erreur=bind(id_de_la_socket,(struct
sockaddr*)&information_sur_la_source,sizeof(information_sur_la_source));
	erreur=99;
	while(erreur!=0)
		erreur=listen(id_de_la_socket,1);
	tempo=sizeof(information_sur_la_source);
	id_de_la_nouvelle_socket=accept(id_de_la_socket,(struct
sockaddr*)&information_sur_la_source,&tempo);
	nombre_de_caractere=recv(id_de_la_nouvelle_socket,buffer,1515,0);
	buffer[nombre_de_caractere]=0;
	cout<<"donnees recues:"<<buffer<<endl;
shutdown(id_de_la_nouvelle_socket,2);
	closesocket(id_de_la_nouvelle_socket);
	closesocket(id_de_la_socket);
	WSACleanup();
	getch();
	}
```

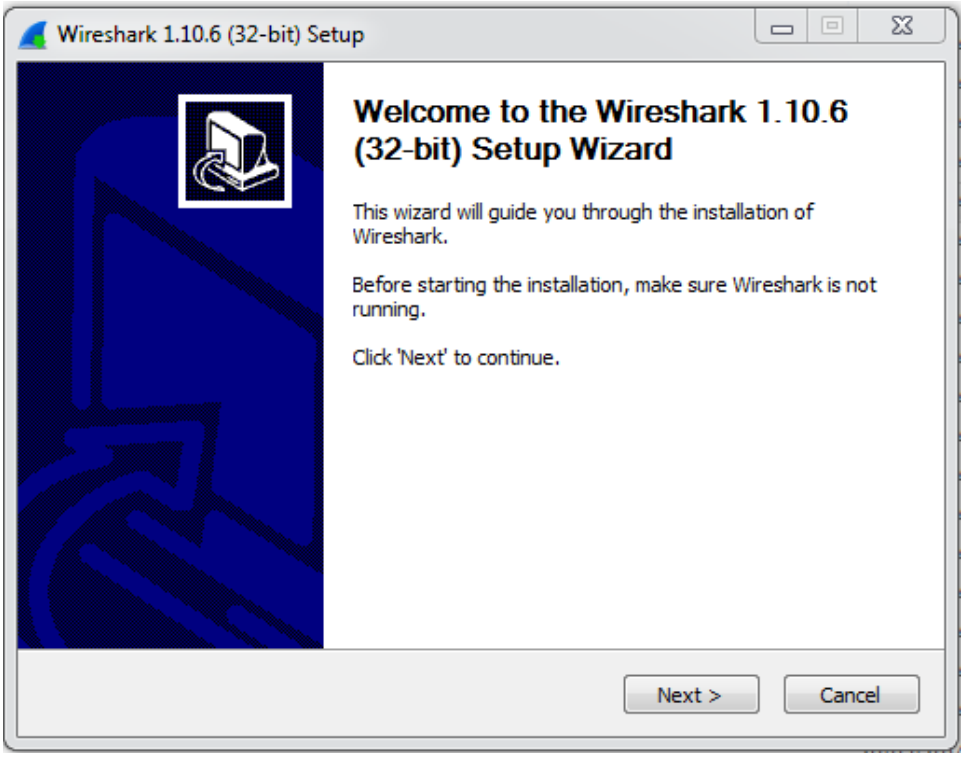
Mise en place du réseau d'ordinateurs :

Dans l'exemple d'application, on a utilisé un réseau de deux ordinateurs reliés directement par un câble réseau ou par l'intermédiaire d'un routeur, un switch filaire ou sans fil (wifi). Deux ordinateurs ont été utilisés, un comme client, et l'autre comme serveur.

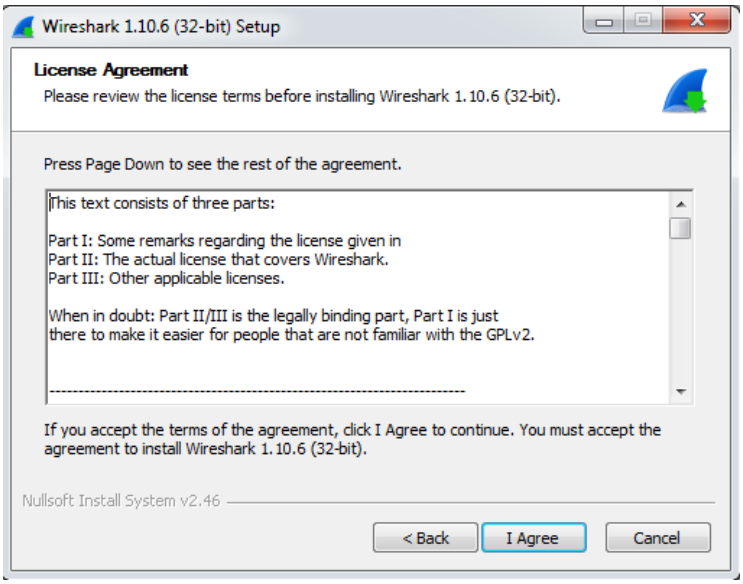
Wireshark :

Installation :

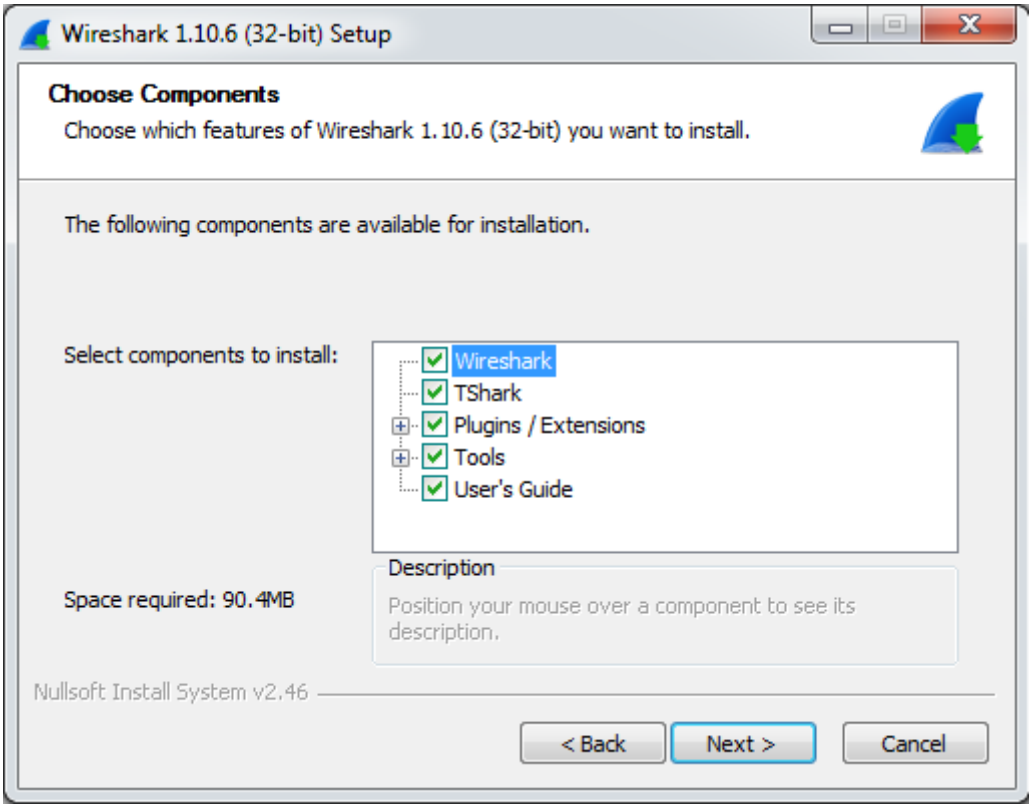
Executer le fichier d'installation de Wireshark, une fenêtre va s'afficher :



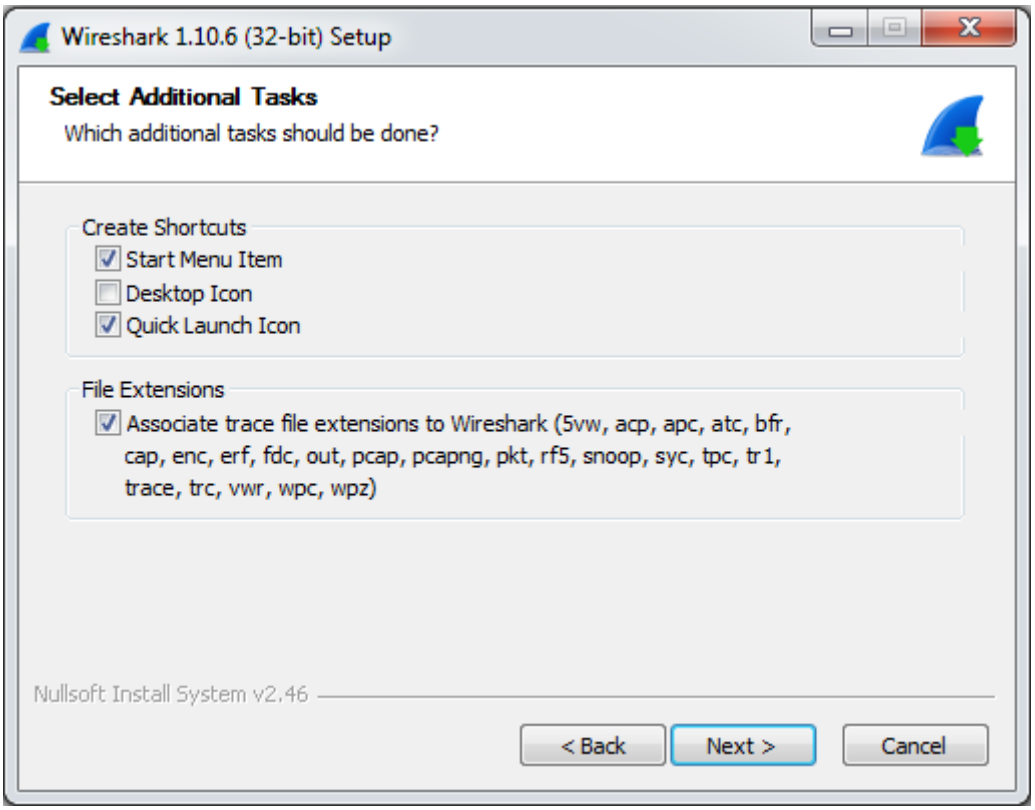
Appuyer sur Next, une autre fenêtre va s'afficher :



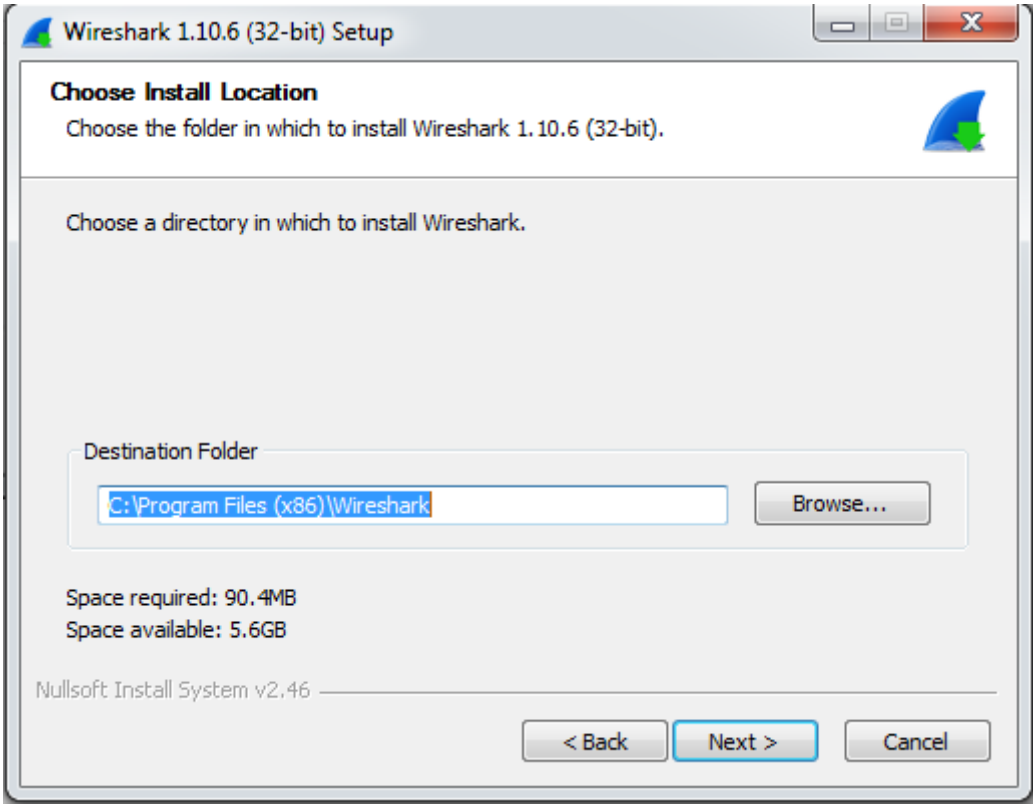
Appuyer sur I Agree, une autre fenêtre va s'afficher :



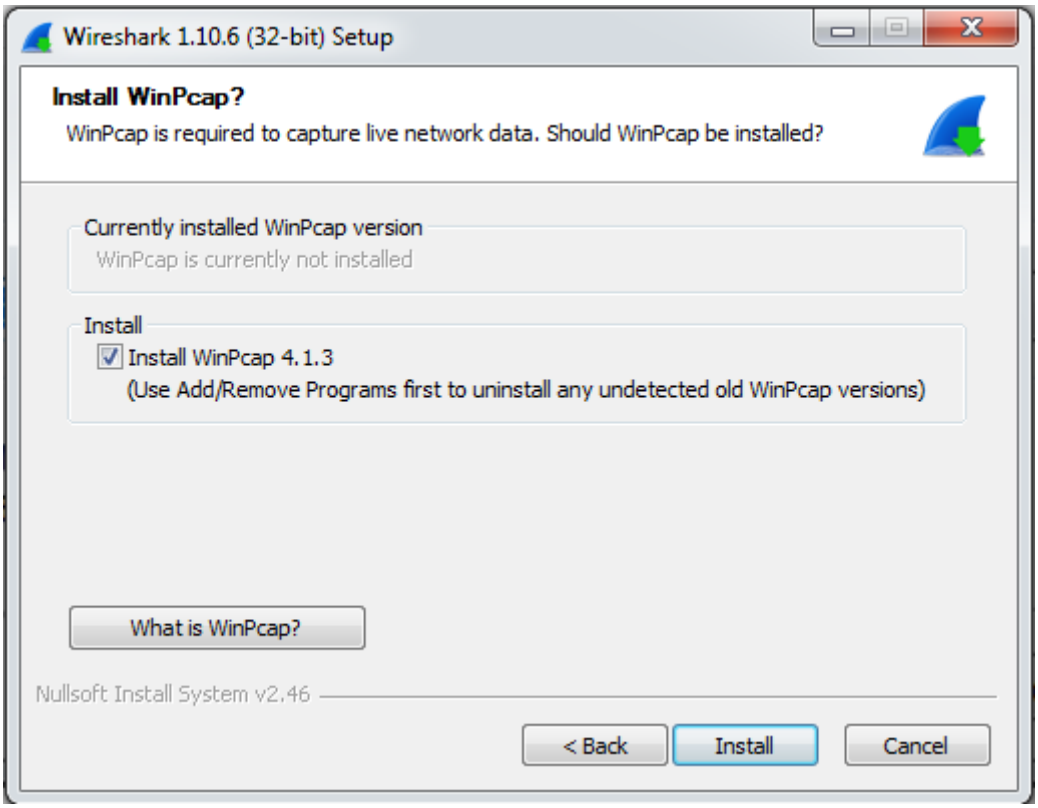
Appuyer sur Next :



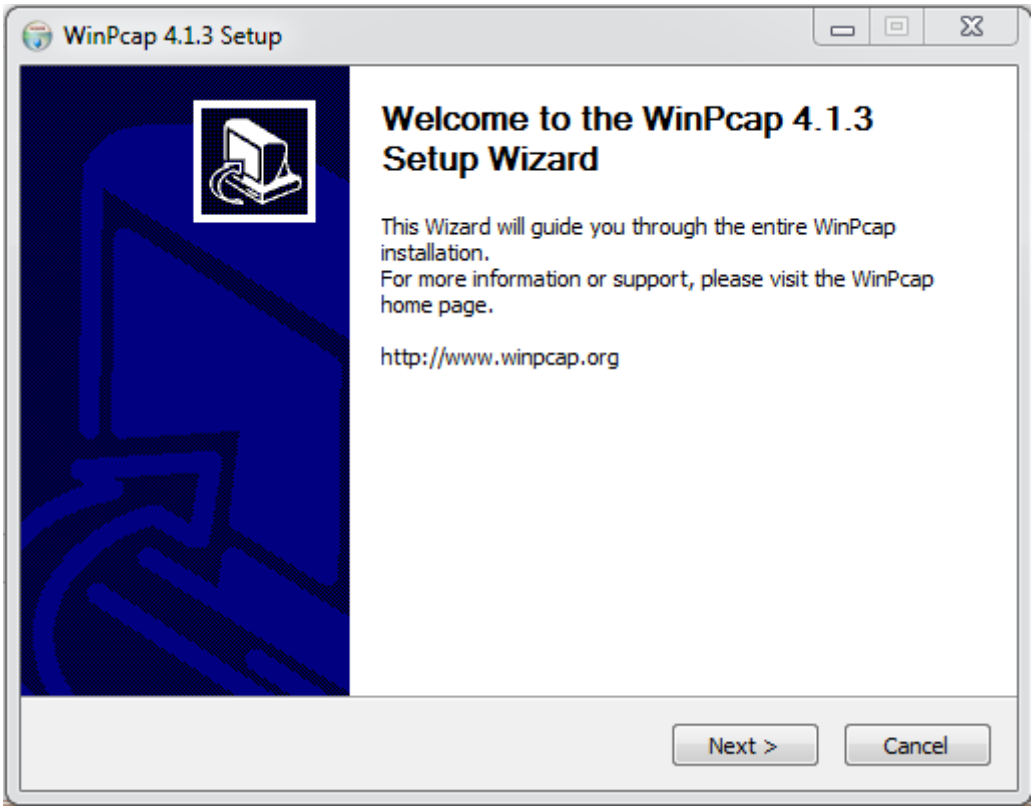
Appuyer sur Next :



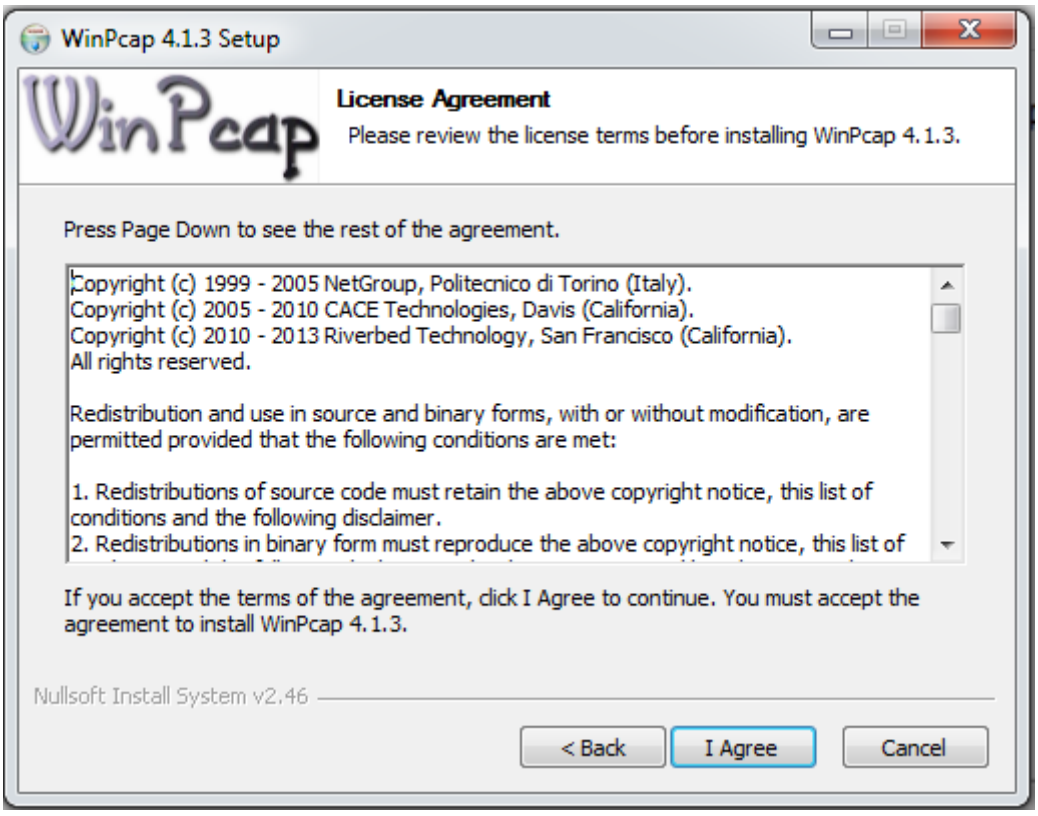
Appuyer sur Next :



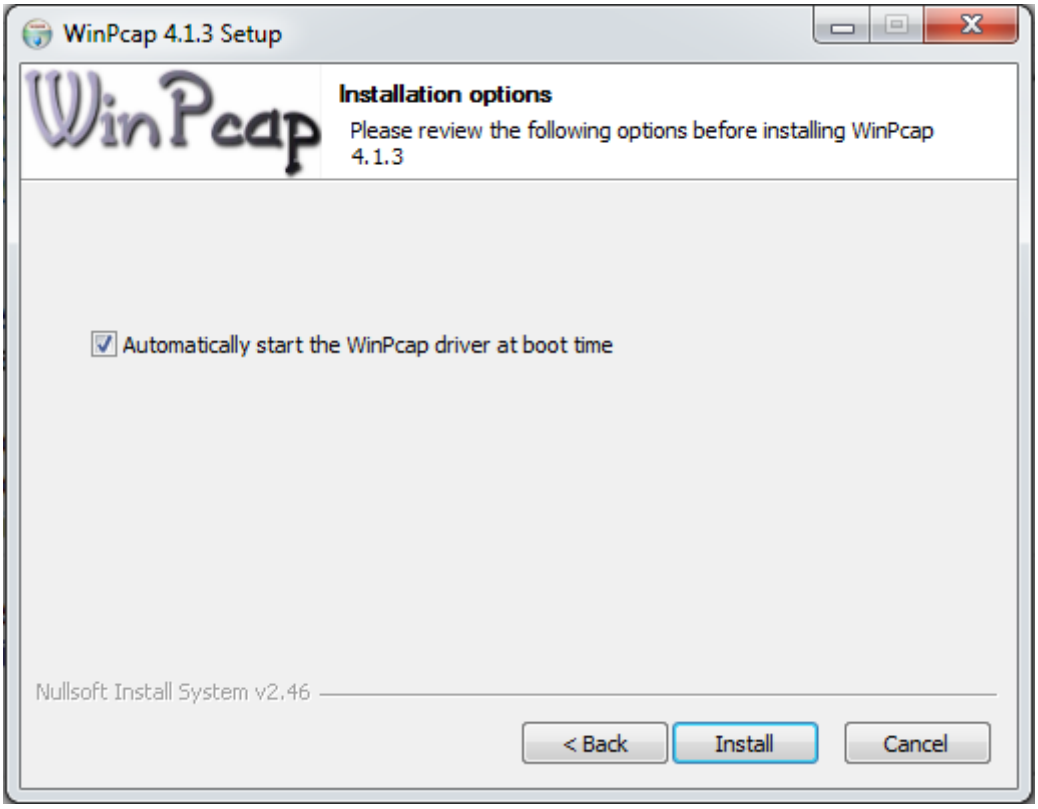
Appuyer sur Install, une autre fenêtre s'affiche (appuyer sur Next) :



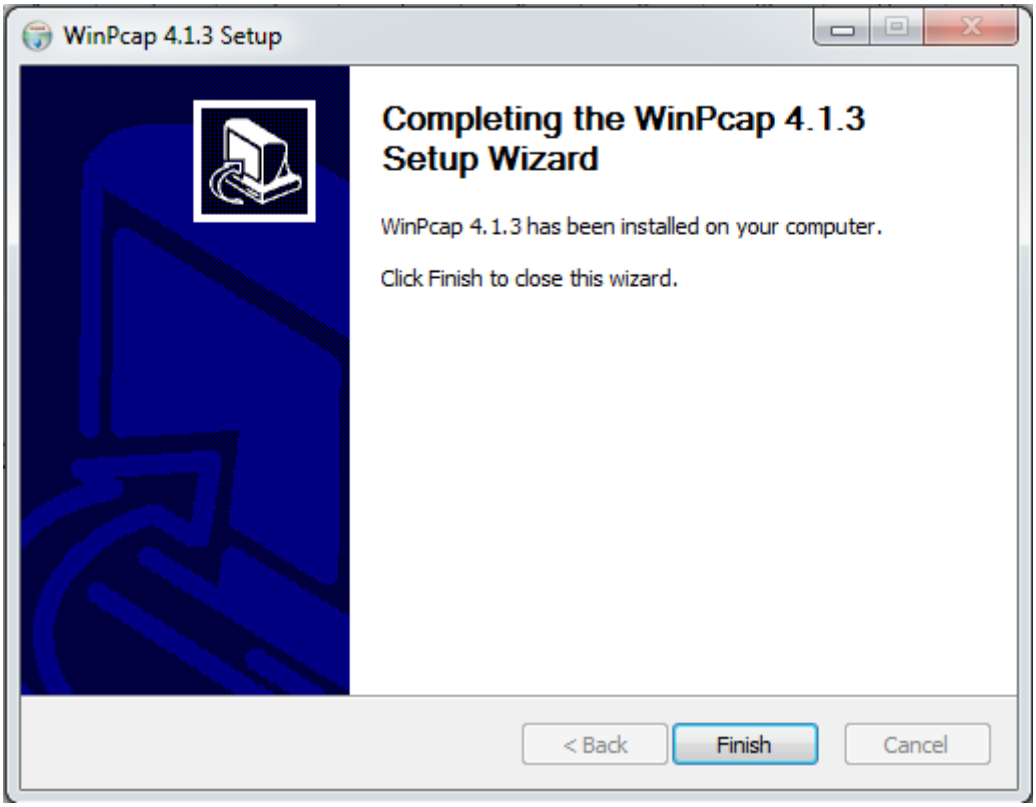
Une autre fenêtre s'affiche, appuyer sur I Agree :



Une autre fenêtre s'affiche, appuyer sur Install :



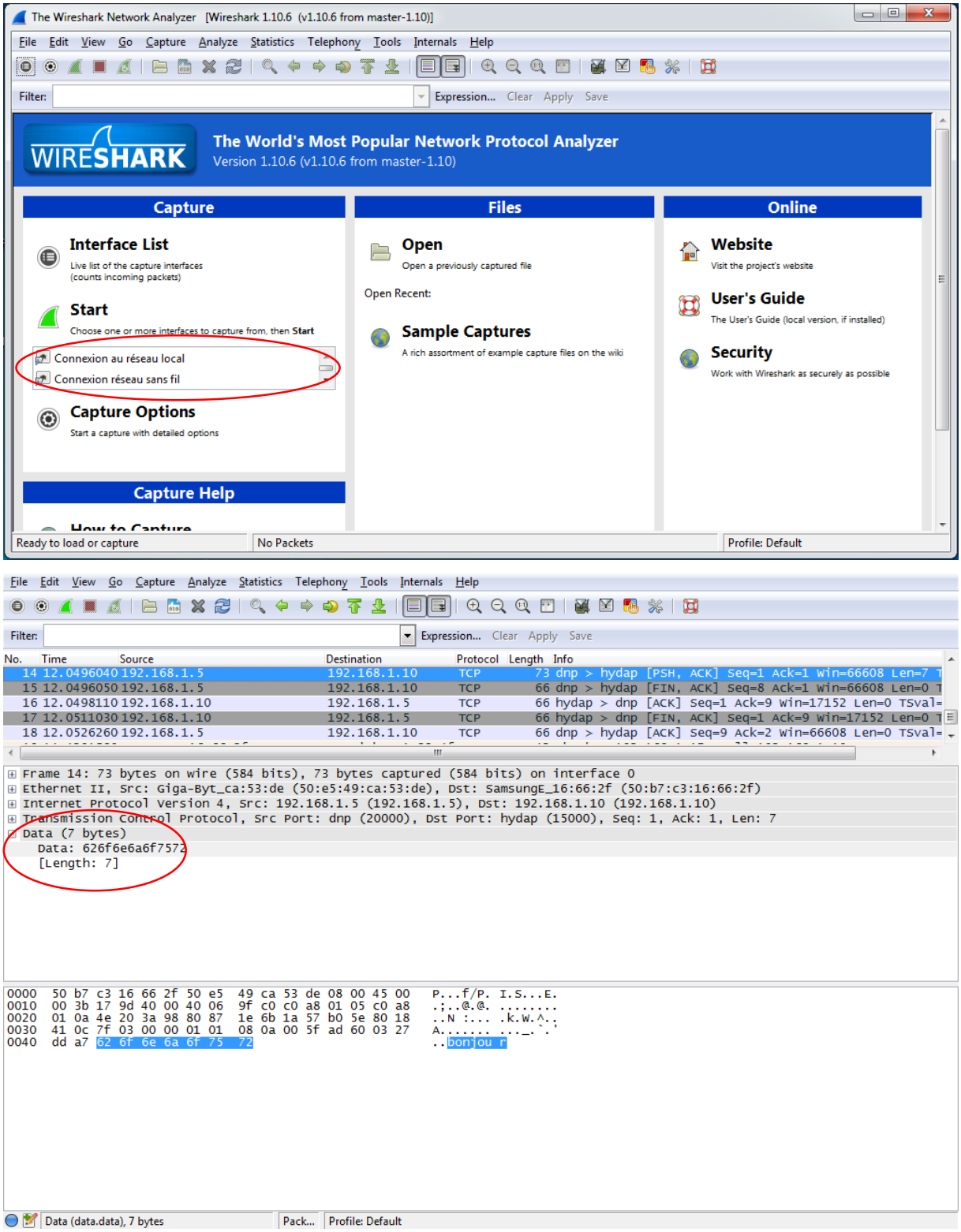
Une fois l'installation terminée, appuyer sur Finish :



Démarrage de Wireshark :

Double cliquer sur l'icône sur le bureau :

Dans la fenêtre qui va apparaitre, choisir le type de liaison (câble ou sans fil) et appuyer sur start.



Une fois le message envoyé, appuyer sur Stop

Vérifier sur les paquets capturés ceux qui ont les informations correspondantes à l'expéditeur et au destinataire du message pour y retrouver les données envoyées (Data).