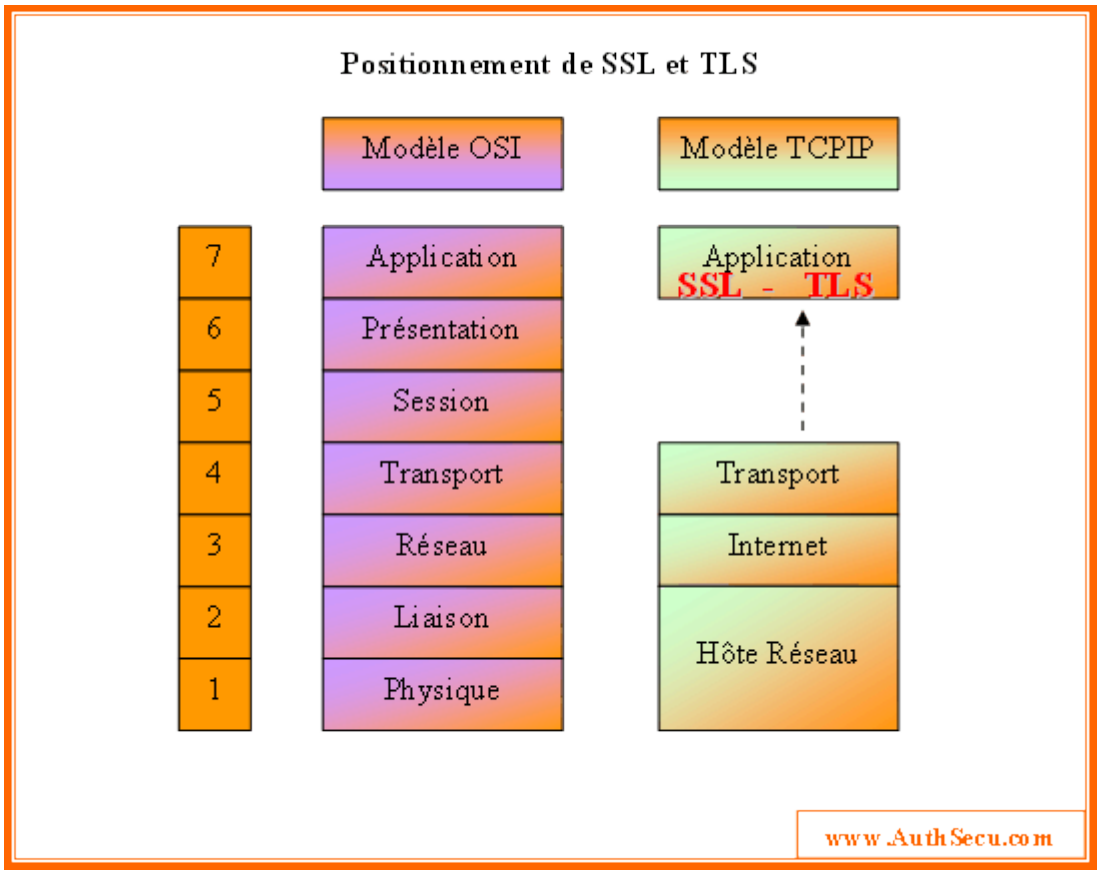
PROTOCOLE SSL ET TLS

1 – Généralités du protocole SSL et TLS

1.1 – Positionnement des protocoles SSL et TLS

SSL signifie Secure Sockets Layer et son équivalent actuel TLS signifie Transport Secured Layer. Ils sont tous les deux des protocoles situés entre le niveau Transport et Application.



SSL et TLS se comportent en effet comme une couche intermédiaire supplémentaire, car ils sont indépendants du protocole utilisé au niveau application. Cela signifie donc qu'il peut aussi bien être employé pour sécuriser une transaction web, l'envoi ou la réception d'email, etc. SSL et TLS sont donc transparents pour l'utilisateur et ne nécessitent pas l'emploi de protocoles de niveau Application spécifiques.

1.2 – Objectifs et moyens mis en oeuvre

SSL et TLS proposent les fonctionnalités suivantes :

- Authentification – Le client doit pouvoir s'assurer de l'identité du serveur. Depuis SSL 3.0, le serveur peut aussi demander au client de s'authentifier. Cette fonctionnalité est assurée par l'emploi de certificats.
- Confidentialité – Le client et le serveur doivent avoir l'assurance que leur conversationne pourra pas être écoutée par un tiers. Cette fonctionnalité est assurée par un algorithme de chiffrement.
- Identification et intégrité – Le client et le serveur doivent pouvoir s'assurer que les messages transmis ne sont ni tronqués ni modifiés (intégrité), qu'ils proviennent bien de l'expéditeur attendu. Ces fonctionnalités sont assurées par la signature des données

SSL et TLS reposent donc sur la combinaison de plusieurs concepts cryptographiques, exploitant la fois le chiffrement asymétrique et le chiffrement symétrique.

SSL et TLS se veut en outre évolutif, puisque le protocole est indépendant des algorithme de cryptage et d'authentification mis en oeuvre dans une transaction. Cela lui permet de s'adapter aux besoins des utilisateurs et aux législations en vigueur. Cela assure de plus une meilleure sécurité, puisque le protocole n'est pas soumis aux évolutions théoriques de la cryptographie (Si un chiffrement devient obsolète, le protocole reste exploitable en choisissant un chiffrement réputé sûr).

1.3 – Fonctionnement

Le protocole SSL et TLS se décompose en deux couches principales (quatre en réalité) :

- SSL et TLS Handshake Protocol choisit la version de SSL et TLS qui sera utilisée, réalise l'authentification par l'échange de certificats et permet la négociation entre le client et le serveur d'un niveau de sécurité au travers du choix des algorithmes de cryptage. C'est le protocole de configuration de la transaction.
- SSL et TLS Record Protocol encapsule et fragmente les données. C'est le protocole de transmission des données.

Dans une première phase, le client et le serveur vont effectuer la négociation an de configurer la transaction et d'échanger les clés de chiffrement. Puis ils effectueront l'échange de données proprement dit.

1.4 – Plan de ce document

Ce document présente dans un premier temps l'évolution de SSL depuis sa

création jusqu'à l'apparition du protocole actuel TLS et ses différentes perspectives d'avenir. Il replace aussi SSL dans le contexte des différentes solutions existantes de sécurisation des transactions sur le réseau. Puis nous rappelleront les concepts cryptographiques mis en oeuvre. Nous détaillerons ensuite le fonctionnement technique des protocoles SSL et TLS. Enfin, nous évoquerons les limites de ces protocoles en matière de sécurité.

2 – Présentation de SSL et TLS

2.1 – Historique

Voici l'historique des protocoles SSL et TLS par sortie de version :

•SSL 1.0
1994
Netscape

•SSL 2.0
Février 1995
Netscape
The SSL Protocol Version 2.0

•SSL 3.0
Novembre 1996
Netscape
The SSL Protocol Version 3.0

•TLS 1.0
Janvier 1999
IETF
RFC 2246

•Extensions TLS
Juin 2003
IETF
RFC 3546

•Extensions TLS
Avril 2006
IETF
RFC 4366

2.2 – Perspectives actuelles

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 l'origine uniquement sécuriser les transactions entre un client et un serveur web (HTTP), la spécification a été connue de faon 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 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, updatant la précédente, est sortie en Avril 2006.

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

2.3 – Contexte technique

Comme il a été mentionné ci-dessus, SSL et TLS sont des protocoles transparents pour l'utilisateur, situés entre les couches Application et Transport. De nombreux protocoles peuvent donc exploiter SSL et TLS, tels HTTP (HTTPS), LDAP (LDAPS), etc.

Cependant, si SSL et TLS est transparent au niveau des protocoles, il ne l'est pas au niveau des applications qui l'exploitent. Celles ci nécessitent donc individuellement des aménagements pour prendre en compte SSL et TLS. L'une des faiblesses de SSL et TLS est de donc disposer d'un nombre encore relativement réduit d'implémentations.

2.3.1 – Implémentations de SSL et TLS

- Implémentations dans les navigateurs web
  La majeure partie des implémentations de SSL et TLS se trouve dans les navigateurs et serveurs web. Le serveur Apache, notamment, peut exploiter SSL grâce une implémentation basée sur OpenSSL.
- OpenSSL

Implémenté en C, OpenSSL est une boîte à outils de chiffrement comportant deux bibliothèques (une de cryptographie générale et une implémentant le protocole SSL), ainsi qu'une commande en ligne. OpenSSL supporte SSL 2.0, SSL 3.0 et TLS 1.0. OpenSSL est distribué sous une licence de type Apache.

•GnuTLS
Le projet GnuTLS propose une implémentation du protocole TLS conforme aux spécifications de l'IETF. GnuTLS supporte TLS 1.1, TLS 1.0, SSL 3.0 et les extensions TLS. Il permet l'authentification via les certificats X509 et PGP. A la différence d'OpenSSL, GnuTLS est compatible avec les licences GPL.

2.3.2 – Certification

Ainsi que nous le verrons dans la présentation des concepts cryptographiques, le principal risque associé l'utilisation de SSL et TLS est le détournement des certificats. An de parer ce risque, il importe de se doter d'autorités de certifications,destinées valider ou invalider les certificats. Deux philosophies s'affrontent.

•La certification X.509
La certification X.509 repose sur le principe d'autorités centralisées, dont le rôle est détenir jour une liste des certificats valides. Elle doit aussi révoquer les certificats expirés, douteux, etc. L'utilisateur se réfère cette autorité chaque fois qu'il veut contrôler la validité d'un certificat. Ces autorités sont organisées hiérarchiquement, de faon que la plus haute soit une autorité de confiance maximale. Le rôle d'une autorité supérieure est de valider les autorités qui dépendent d'elle.

•La certification PGP
La certification PGP (Pretty Good Privacy) est basée sur la philosophie Les amis de mes amis sont mes amis . En effet, la validité des certificats se transmet de pair pair : si un utilisateur valide ou invalide un certificat, il transmet l'information aux utilisateurs avec lesquels il est directement en relation (il s'agit donc d'utilisateurs de confiance). De proche en proche,la certification se transmet.
Les inconvénients majeurs de PGP, sont d'une part la difficulté invalider un certificat s'il a été au préalable reconnu par beaucoup d'utilisateurs et d'autre part le problème savoir si les réseaux de confiance peuvent être suffisamment fiables.

Les protocoles SSL et TLS ont choisi de se baser sur les certifications X.509.

2.3.3 – SSL et TLS par rapport aux autres solutions

D'autres protocoles permettent d'assurer la sécurité sur le réseau. Bien que proposant des fonctionnalités concurrentes de SSL et TLS, ils sont plutôt considérés comme complémentaires.

•SSH

SSH est un protocole de niveau application qui propose une alternative sécurisée aux utilitaires classiques (rlogin, rsh, telnet) qui n'offrent pas de confidentialité. La possibilité d'exploiter un mécanisme de tunneling rend SSH, comme SSL et TLS compatible avec les autres protocoles de niveau application déjà existant. Tout comme SSL et TLS, SSH assure l'authentification des machines, la confidentialité et l'intégrité des données. Il assure aussi l'authentification des utilisateurs par mot de passe.
SSH souffre de faiblesses par rapport SSL et TLS : il n'intègre pas la notion de certificats X509 v3, nécessite l'installation d'une application cliente spécifique (pas de transparence).De plus, la notion de tunneling reste difficile appréhender.
Cependant, SSH est moins vulnérable que SSL et TLS en matière d'identification du client. En effet, la protection du certificat sur un poste client ne peut pas toujours être correctement assurée.

•IPSec
IPSec fournit un mécanisme de sécurisation au niveau de la couche réseau (IP). Il est utilisé notamment pour la mise en oeuvre de réseaux privés virtuels (VPN). Les fonctionnalités d'IPSec sont l'authentification des machines, la confidentialité et l'intégrité des transactions.
Son implémentation indissociable de la prochaine version du protocole IP, IPv6, entre en concurrence avec les fonctionnalités de confidentialité et d'intégrité de SSL et TLS. Elle offre en outre une sécurisation du réseau dans sa globalité et non des applications au cas par cas.
Cependant, IPSec ne peut assurer l'authentification des utilisateurs, ce qui pose le problème de la fiabilité des postes individuels. De plus, étant encore une technologie jeune,il se pose les problèmes de manque de recul et d'interopérabilité.
A ce jour, donc, les fonctionnalités de sécurité d'IPSec et IPv6 sont vues comme un important complément la sécurité offerte par SSL et TLS.

•SET/3D-Secure
Basé sur SSL et TLS, les protocoles SET (aujourd'hui obsolète) et 3D-Secure proposent une authentification validée par un tiers. Ces protocoles sont principalement destinés aux applications de paiement en ligne (ils sont développés par des institutions bancaires). Si SSL et TLS et SET/3D-Secure assurent chacun un haut degré de confidentialité, seul le SET permet une pleine identification réciproque des deux parties grâce un tiers de confiance,en l'occurrence la banque du vendeur. Ainsi, elle assure le vendeur que la carte est bonne et qu'elle n'a pas été volée et le client qu'aucune utilisation malveillante ne sera faite de ces informations.

On voit ici que, quoique souffrant de limitations, l'univers SSL et TLS est vaste et stable.

2.4 – Les ports et applications utilisant SSL et TLS

Voici la liste des applications exploitant SSL et TLS avec leurs ports

TCP associés :

•Protocole : NSIIOPS
Port TCP : 261
Description : IIOP Name Service sur SSL et TLS

•Protocole : HTTPS
Port TCP : 443
Description : HTTP sur SSL et TLS

•Protocole : DDM-SSL
Port TCP : 448
Description : DDM-SSL

•Protocole : SMTPS
Port TCP : 465
Description : SMTP sur SSL et TLS

•Protocole : NNTPS
Port TCP : 563
Description : NNTP sur SSL et TLS

•Protocole : SSHELL
Port TCP : 614
Description : SSL Shell

•Protocole : LDAPS
Port TCP : 636
Description : LDAP sur SSL et TLS

•Protocole : FTPS-DATA
Port TCP : 989
Description : FTP données sur SSL et TLS

•Protocole : FTPS
Port TCP : 990
Description : FTP controle sur SSL et TLS

•Protocole : TELNETS
Port TCP : 992
Description : Telnet sur SSL et TLS

•Protocole : IMAPS
Port TCP : 993
Description : IMAP4 sur SSL et TLS

•Protocole : IRCS
Port TCP : 994
Description : IRC sur SSL et TLS

•Protocole : POP3S
Port TCP : 995
Description : POP3 sur SSL et TLS

3 – Aspects cryptographiques

Les concepts mis en oeuvre dans SSL et TLS reposent sur des notions de cryptographie usuelles. SSL et TLS utilise en effet des méthodes de chiffrement symétrique et asymétrique pour assurer ses diverses fonctions : authentification, signature, intégrité et identification.

Sans aborder les algorithmes exploités, il est néanmoins intéressant de se pencher sur les bases cryptographiques qui ont donné naissance aux protocoles.

Mathématiquement, les protocoles d'échange de messages, appelés cryptosystèmes, sont représentés sous forme d'un quintuplet

$$\Gamma = (M, C, K, E, D)$$

M représente l'ensemble des messages clairs
C l'ensemble des messages codés
K l'ensemble des clés
E l'ensemble des fonctions de chiffrement (c'est dire un ensemble de la forme :

$$(e_k)_{k \in K})$$

D l'ensemble des fonctions de déchiffrement (c'est dire un ensemble de la forme :

$$(d_k)_{k \in K})$$

3.1 – Chiffrement symétrique ou à clé secrète

Le chiffrement symétrique, dit aussi clé secrète est la forme la plus ancienne de cryptage. Elle consiste utiliser une valeur courte (la clé) pour rendre un message inintelligible aux tierces parties.

Elle est dite symétrique car cette même clé permet ceux qui en ont connaissance de déchiffrer le message et ainsi d'accéder son contenu.

Mathématiquement, un cryptosystème clef secrète est donc déni par la condition :

$$d_k(e_k(m)) = m$$
c'est à dire $m = d_k(c)$ et $c = e_k(m)$
avec $m \in M, c \in C, k \in K, e_k \in E, d_k \in D$

Concernant l'implémentation d'un protocole de sécurité, le chiffrement symétrique est intéressant car il est simple mettre en oeuvre et requiert un faible temps de calcul. Il valide la contrainte de confidentialité,aussi longtemps que la clé reste secrète.

Cependant, il présente un inconvénient majeur : la difficulté de protéger le secret d'une clé. En effet, au moment ou les deux parties échangent leur clé secrète, ils ne peuvent pas s'assurer que celle ci n'est pas interceptée par un tiers. Cette solution s'avère donc, seule, insuffisante.

3.2 – Chiffrement asymétrique ou clé publique

3.2.1 – Chiffrement

Le chiffrement asymétrique, dit aussi clé publique découle de découvertes théoriques relativement récentes dans le domaine mathématique. Il repose sur l'existence de fonctions mathématiques difficiles inverser, sauf en exploitant une brèche secrète.

La clé du cryptosystème est ici un couple (kprivate, kpublic). kprivate est connue du seul récipiendaire des messages et kpublic est universellement connue.

Les fonctions de codage et de décodage deviennent alors respectivement ekpublic et dkpublic, kprivate, c'est dire qu'il est possible toute personne de coder un message, mais que seul le détenteur de la clé privée (récipiendaire du message) est capable de le décoder.

On l'appelle donc asymétrique, car il est facile pour un utilisateur quelconque de coder un message, mais difficile de décoder ce message. Le chiffrement asymétrique permet donc de s'assurer de l'identité du destinataire.

3.2.2 – Signature

La signature permet de s'assurer de l'identité de l'expéditeur d'un message. Les méthodes de signature, exploitent en réalité un protocole inverse de celui du chiffrement.

L'expéditeur va coder un message donné avec sa clé privée. Lui seul en est capable. En revanche, n'importe qui peut vérifier que le message a été codé par l'expéditeur, en le décodant avec la clé publique et en vérifiant qu'il correspond bien au message d'origine.

En effet, on utilise pour la signature une fonction brèche secrète telle que :

$$e(d(m)) = d(e(m)) = m$$

Comme seul l'expéditeur souhaité connais d(), le récipiendaire peut s'assurer que la communication reçue en vérifiant que :

$$e(c) = e(d(m)) = m$$

Cependant, si ce procédé identifie l'expéditeur, il possède l'inconvénient de rendre le message transmis déchiffrable par tous.

3.2.3 – Une première limite

Il est donc théoriquement possible d'utiliser un empilement de deux cryptosystèmes, l'un de signature clé publique, l'autre de chiffrement clé publique : ainsi, les deux parties s'identifient l'une et l'autre, chacune connaissant la clé publique de leur interlocuteur.

Cependant, les algorithmes a clé secrète sont gourmands en ressources, tant pour le codage que le décodage et c'est pourquoi cette solution ne pouvait être adoptée dans le protocole SSL et TLS. Le cryptage asymétrique a donc été associé aux étapes critiques du protocole : l'échange des clés secrètes, permettant l'emploi sécurisé de chiffrements symétriques (beaucoup plus légers) et, sous une forme allégée, la signature et la vérification de l'intégrité des données.

3.2.4 – Signature et hachage

Le principe de signature et de vérification d'intégrité est simplifié par l'emploi préalable d'une fonction de hachage. Une fonction de hachage calcule le résumé d'un texte. Ce résumé doit être sens unique, pour éviter de reconstituer le message initial connaissant seulement le résumé. Il doit être très sensible, c'est à dire qu'une petite modification du message entraîne une grande modification du résumé. En expédiant un message accompagné de son résumé (on dit aussi son haché), on peut s'assurer de l'intégrité du message, en recalculant le résumé l'arrivée.

En associant hachage et signature, il devient possible d'assurer les fonctionnalités de contrôle d'intégrité et d'identification de l'expéditeur de faon simple est performante. La signature devient facile calculer (car le hacher est beaucoup plus court que le message),et son intégrité, ainsi que celle du message, deviennent interdépendantes. C'est ce procédé qui sera utilisé dans SSL et TLS, sous le nom de signature MAC.

3.3 – Apparition de la notion de Certification

3.3.1 – L'attaque par le milieu

Un problème reste cependant : la fiabilité du système décrit ci-dessus repose sur la confiance accordée aux clés publiques. Or celles-ci doivent elles même se faire connaître des participants une transaction. Ainsi, toute la sécurité vole en éclat si un tiers malintentionné est capable de diffuser, l'insu des participants, des clés publiques contrefaites.

- Cas normal
  - 1 – Alice et Bob échangent leur clé publique. Charles peut les lire, il connaît donc kapublicet kbpublic.
  - 2 – Si Alice veut envoyer un message Bob, elle chiffre ce message avec kbpublic. Bob le déchiffre avec kbprivate.
  - 3 – Charles, qui ne possède que kbpublic, ne peut pas lire le message.
- Attaque, Admettons maintenant que Charles soit en mesure de modifier les échanges entre Alice et Bob.
  - 1 – Bob envoie sa clé publique Alice. Charles l'intercepte et renvoie Alice sa propre clé publique kcpublic en se faisant passer pour Bob.
  - 2 – Lorsque Alice veut envoyer un message a Bob, elle utilise donc, sans le savoir, la clé de Charles.
  - 3 – Alice chiffre le message avec la clé publique de Charles et l'envoie celui qu'elle croit être Bob.
  - 4 – Charles intercepte le message, le déchiffre avec sa clé privée kcprivate et peut lire le message.
  - 5 – Puis il chiffre nouveau le message avec la clé publique de Bob kbpublic, après l'avoir éventuellement modifié.
  - 6 – Bob déchiffre son message avec sa clé privée et ne se doute de rien puisque cela fonctionne.

Ainsi, Alice et Bob sont chacun persuadés d'utiliser la clé de l'autre, alors qu'ils utilisent en réalité tous les deux la clé de Charles.

3.3.2 – Se prémunir contre l'attaque par le milieu

On peut se prémunir contre cette attaque de diverses faons. Les deux principales sont la méthode des réseaux de confiance, la méthode de certification par un tiers de confiance. On trouve aussi des méthodes d'échange direct (main propre, téléphone) et d'authentification par mot de passe.

3.3.3 – La certification X.509

Le principe de la certification par une autorité repose sur la confiance accordée des organismes centraux. L'entité souhaitant obtenir une certification s'adresse une autorité,en lui fournissant sa clé publique. L'autorité, après avoir vérifié l'identité du demandeur, va fournir un certificat, auquel il adjoint sa propre signature : celle-ci permet alors de s'assurer que le certificat a bien été émis par une autorité compétente.

L'organisme de certification fait alors office de tiers de confiance. Le protocole retenu pour la certification sous SSL et TLS est X.509 v3.

Les certificats racine (c'est dire émanent d'autorités hautement ables) sont implémentés directement dans les navigateurs Web. Le problème l'heure actuel reste surtout l'absence de mise jour des certificats, en cas de compromission (de la clé privée, par exemple).

Structure d'un certificat X.509

- •Version
- •Numéro de série
- •Algorithme de signature du certificat
- •Signataire du certificat
- •Validité (dates limite)
  - •Pas avant
  - •Pas après
- •Détenteur du certificat
- •Informations sur la clé publique
  - •Algorithme de la clé publique
  - • Clé publique
- •Identifiant unique du signataire (Facultatif)
- •Identifiant unique du détenteur du certificat (Facultatif)
- •Extensions (Facultatif)
  - •Liste des extensions…

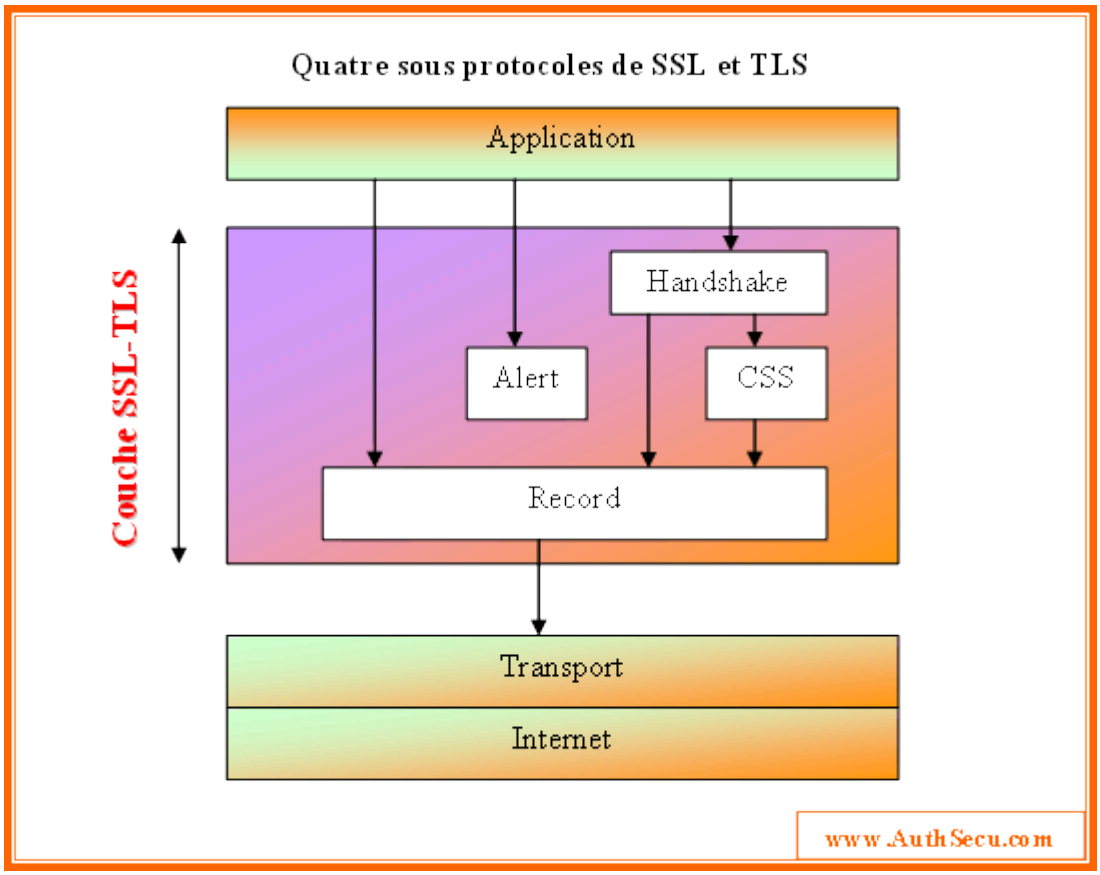
4 – Le protocole SSL et TLS

Le protocole SSL et TLS est subdivisé en quatre sous protocoles :

- •Handshake Protocol – Ce protocole négocie les paramètres de cryptage qui seront l'oeuvre lors de la connexion.
- •Change Cipher Spec Protocol – Ce protocole annonce la n du protocole de négociation.
- •Alarm Protocol – C'est le protocole de signalement d'erreurs et d'alertes.

•Record Protocol – Ce protocole se place entre les autres et la couche 4. C'est lui qui assure le rôle de communication de SSL et TLS.

Ils s'organisent comme présenté dans le schéma suivant :



Commençons par le protocole de négociation, car c'est le premier utilisé.

4.1 – Le protocole Hanshake

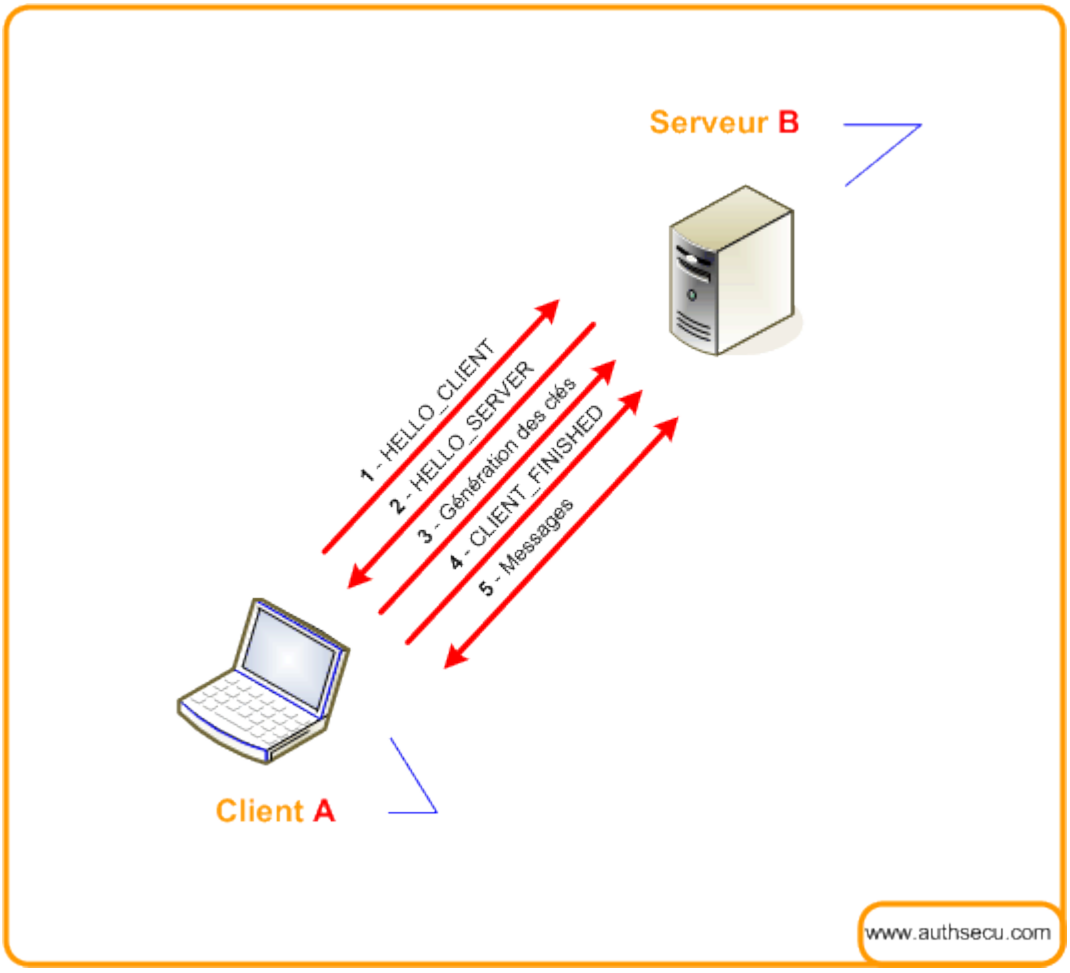
Il permet au client et au serveur de s'authentifier mutuellement, de négocier les algorithmes de chiffrement, de négocier les algorithmes de MAC (Message Authentification Code) et enfin de négocier les clés symétriques qui vont servir au chiffrement.

4.1.1 – Format d'échange

Chaque message échangé entre le client et le serveur contient trois champs :

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

4.1.2 – Détail des échanges



1 – Le client envoie un message HELLO_CLIENT, en clair, au serveur. Ce message contient :

- Version – La plus haute version de SSL que puisse utiliser le client.
- Random – Un horodatage de 32 bits et une valeur aléatoire de 28 octets générée par le client. Le nombre obtenu va servir la signature des messages.
- Session ID – Un nombre, qui identifie la connexion. Un zéro signifie la volonté du client d'établir une nouvelle connexion sur une nouvelle session. Un autre nombre signifie la volonté de changer les paramètres ou de créer une nouvelle connexion sur la session existante.
- CipherSuite – Une liste, par ordre décroissant de préférence, des algorithmes que supporte le client. Il s'agit des algorithmes d'échange de clé et de chiffrement.
- Compression Method – Liste, par ordre décroissant de préférence, des algorithmes de compression supportés par le client.

Puis le client attend une réponse du serveur

2 – Le serveur répond au client en clair : HELLO_SERVER. Le message contient :

- •Version – La plus haute version de SSL que puisse utiliser le client.
- •Random – Un horodatage de 32 bits et une valeur aléatoire de 28 octets générée par le client.
- •Session ID – L'Identifiant de la session qui débute.
- •CipherSuite – La séquence d'algorithmes choisis pour la session. Le serveur sélectionne la première suite qu'il connait dans la liste transmise par le client.
- •Compression Method – La méthode de compression qui va être utilisée.

Maintenant que les algorithmes sont choisis, le serveur va s'authentifier auprès du client. Il envoie pour cela son ou ses certificats (X.509) au client. Il peut pendant cette étape demander un certificat au client. Le client vérifie l'authenticité du serveur : si cette authenticité est mise en doute, la transaction est interrompue.

3 – Génération des clés de chiffrement symétrique. Le client génère de son côté une préclé qui servira a produire les clés utilisées
par la suite. Cette clé est envoyée au serveur,cryptée a l'aide de sa clé publique. A l'aide de cette pré clé, le serveur et le client génèrent quatre clés pour la session :

- •Server write mac secret – utilisée dans la signature des messages du serveur.
- •Client write mac secret – pour les messages du client.
- •Server write key – pour chiffrer les données émises par le serveur.
- •Client write key – pour le client.

Ces clés ne sont pas échangées. Si besoin est, le serveur vérifie l'authenticité du client.

4 – Le client envoie le message CLIENT_FINISHED au serveur. Ce message est chiffré et signé a l'aide des clés ci-dessus. Cela signifie qu' partir de maintenant, le client communique de cette manière.

5 – Le serveur procède de même. Ces messages sont dénis par le sous protocole Change Cipher Spec (c'est d'ailleurs tout ce que définit ce protocole).

4.2 – Le protocole Change Cipher Spec

Ce protocole contient un seul message : change_cipher_spec. Il est envoyé par les deux parties la n du protocole de négociation. Ce message transite chiffré par l'algorithme symétrique précédemment négocié.

4.3 – Le protocole Alarm

Ce protocole spécifie les messages d'erreur que peuvent s'envoyer clients et

serveurs. Les messages sont composés de deux octets. Le premier est soit warning soit fatal. Si le niveau est fatal, la connexion est abandonnée. Les autres connexions sur la même session ne sont pas coupées mais on ne peut pas en établir de nouvelles. Le deuxième octet donne le code d'erreur.

Les erreurs fatales sont :

- Unexpected_message – indique que le message n'a pas été reconnu
- Bad_record_mac – signale une signature MAC incorrecte
- Decompression_failure – indique que la fonction de décompression a reçu une mauvaise entrée
- Handshake_failure – impossible de négocier les bons paramètres
- Illegal_parameter – indique un champ mal formaté ou ne correspondant rien.

Les warnings sont :

- Close_notify – annonce la fin d'une connexion
- No_certificate – répond une demande de certificat s'il n'y en a pas
- Bad_certificate – le certificat reçu n'est pas bon (par exemple, sa signature n'est pas valide)
- Unsupported_certificate – le certificat reçu n'est pas reconnu
- Certificate_revoked – certificat révoqué par l'émetteur
- Certificate_expired – certificat expiré
- Certificate_unknown – pour tout problème concernant les certificats et non listé ci-dessus.

4.4 – Le protocole Record

Ce protocole chapeaute les autres protocoles de SSL et TLS, en fournissant une interface unifiée pour la transmission des données.

4.4.1 – Rôle

- Encapsulation – Permet aux données SSL et TLS d'être transmises et reconnues sous une forme homogène.
- Confidentialité – Assure que le contenu du message ne peut pas être lu par un tiers : les données sont chiffrées en utilisant les clés produites lors de la négociation.
- Intégrité et Identité – Permet de vérifier la validité des données transmises, grâce aux signatures MAC : cette signature est elle aussi générée l'aide des clés produites lors de la négociation.

4.4.2 – Processus d'encapsulation

•Segmentation – Les données sont découpées en blocs de taille inférieure 16 384 octets

•Compression – Les données sont compressées en utilisant l'algorithme choisi lors de la négociation. A partir de SSL 3.0, il n'y a plus de compression.

•Signature MAC (0, 16 ou 20 octets) – Une signature des données est générée l'aide de la clé MAC. Comme elle exploite une fonction de condensation, on parlera en réalité de HMAC (Hashed MAC). Elle est calculée de la manière suivante :

```
HASH(
    write mac secret
    | pad_2
    | HASH(
        write mac secret
        | pad_1
        | numéro de ce message
        | protocole pour ce message
        | longueur de ce message
        | données compressées
        )
    )
```

La fonction de condensation HASH() est soit MD5 soit SHA-1.

Pad_1 et pad_2 sont des mots de remplissage (pad_1 = 0x36 et pad_2 = 0x5C (répétés 40 fois pour SHA-1 et 48 fois pour MD5))

•Chiffrement – Le paquet obtenu est chiffré l'aide de la fonction dénie lors de la négociation et de la write key. Les algorithmes actuellement possibles pour le chiffrement sont 3-DES 168, IDEA 128, RC4 128, Fortezza 80, DES 56, DES-40, RC4-40, RC2-40.
Ajout de l'en tête (5 octets) – L'entête SSL est ajoutée et le paquet est passé à la couche inférieure. Il contient :

•Content-Type (1 octet) – Indique le type de paquet SSL et TLS contenu dans l'enregistrement :

•0x20 – Paquet de type Change Cipher Spec

•0x21 – Paquet de type Alert

•0x22 – Paquet de type Handshake

•0x23 – Paquet de type Application Data : ce type correspond aux données effectives de la transaction SSL.

•Major Version (1 octet), Minor Version (1 octet) – Numéros de versions principal et secondaire du protocole SSL et TLS utilisé. Par exemple, le protocole TLS1.0 sera identifié avec la paire (0x03,0x01), car TLS 1.0 est considéré comme la version 3.1 de SSL.

(Compressed) Length (2 octets) – Taille (compressée s'il y a lieu)

du fragment SSL et TLS.

4.4.3 – Réceptions des paquets

A la réception des paquets, le destinataire effectue les opérations suivantes :

- 1 – Vérification de l'entête SSL
- 2 – Déchirage du paquet
- 3 – Vérification du champ HMAC (en appliquant la même fonction que ci-dessus aux données déchiffrées puis en comparant le résultat au HMAC reçu)
- 4 – Décompression des données
- 5 – Réassemblage des parties

Si au cours de ces vérifications se passe mal, une alarme est générée.

5 – Faiblesses et attaques envisageables

5.1 – Limites

Les navigateurs n'ont pas 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.

La relation de confiance est dénie par la liste pré installée des autorités de certification :Les navigateurs du commerce sont livrés avec de nombreuses clés publiques pré installées (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 laquelle 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.

Ce problème prend toute son ampleur quand il s'agit de certificats révoqués. En effet, le protocole SSL et TLS (1.0) ne prévoit pas l'obtention de la liste des certificats révoqués (CRL, Certificates Revoked List) par l'autorité de certification avant d'utiliser un certificat (ou périodiquement). Un certificat ainsi signé par une autorité peut être révoqué (utilisation frauduleuse du certificat, clé divulguée…) sans que l'utilisateur n'en soit informé.

On peut alors imaginer l'attaque suivante :

5.2 – Un scénario d'attaque

Un utilisateur malintentionné peut créer un site portant un nom similaire un autre, connu, par exemple www.goggle.com. Cet utilisateur obtient alors un

certificat pour son site et peut ainsi piéger d'autres utilisateurs par fishing, ou autre. Dans ce cas, même si l'autorité révoque le certificat, le site pirate continuera d'être perçu comme sûr par les navigateurs internet.

6 – Conclusion

Les caractéristiques de SSL sont donc :

- L'indépendance vis à vis des couches inférieures et supérieures
- Le fonctionnement en mode Client/Serveur
- L'assurance aux deux parties d'une transaction authentifiée (certificats), privée (cryptage) identifiée et intègre (MAC).

L'âge de SSL lui confère une maturité certaine et d'une longue expérience. Malgré ses défaillances au niveau de l'authentification, son utilisation est très répandue et cela prouve sa robustesse. Son évolutivité, ainsi que son évolution lui promettent un bel avenir.