

heig-vd

Haute Ecole d'Ingénierie et de Gestion  
du Canton de Vaud

Applications et Services Internet

# Laboratoire SSL avec JSSE

RAPPORT DE LABORATOIRE

Romain de Wolff  
Simon Hintermann

IL2008

22 janvier 2008

## Table des matières

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Utilisation du serveur web</b>	<b>1</b>
<b>3</b>	<b>Clé publique générée</b>	<b>1</b>
<b>4</b>	<b>Réponses aux questions</b>	<b>2</b>
4.1	En examinant le certificat du serveur, comment pouvez-vous en déduire que celui-ci est auto-signé? . . . . .	5
4.2	Pourquoi serait-il intéressant de faire signer le certificat par une entité comme Verisign? . . . . .	5
<b>5</b>	<b>Observations</b>	<b>5</b>
5.1	Connexion sans l'authentification client . . . . .	6
5.2	Connexion avec l'authentification client activée . . . . .	6
5.3	Cipher utilisé . . . . .	7
5.4	Principales différences entre SSL v3 et TLS v1 . . . . .	9
5.5	Le message <i>Certificate Verify</i> . . . . .	9
5.6	Configuration du serveur web avec système d'authentification mais sans chiffrement de données . . . . .	9
<b>6</b>	<b>Conclusion</b>	<b>10</b>
<b>7</b>	<b>Références</b>	<b>10</b>
<b>8</b>	<b>Annexe</b>	<b>11</b>

## 1 Introduction

```
dasdsad
// test valeur
if (condif) {
    instructions( check );
}
```

Le but de ce laboratoire est de modifier le serveur web créé en Java durant un laboratoire précédant et d'y implémenter le protocole SSL/TLS. Notre serveur web doit se lancer dans deux mode différent : un mode avec authentification du client nécessaire et un autre sans.

## 2 Utilisation du serveur web

Le lancement du serveur s'effectue à l'aide de la ligne de commande. Pour lancer le serveur avec authentification du client sur le port 443 (port par défaut de SSL) il faut utiliser la commande suivante :

```
java WebServer 1 443
```

Le "1" permet de dire que l'on active l'authentification du client. Pour le rendre facultatif, on utilisera un "0" à la place, comme le montre la commande suivante :

```
java WebServer 0 443
```

Notons que pour lancer le serveur sur le port 443 comme montré ci dessus il faut exécuter la commande en tant qu'administrateur.

## 3 Clé publique générée

Voici la clé publique qui nous a été generée par Keytool :

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBujCCASMAQAwEjELMAkGA1UEBhMCQ0gx CzA JBgNVBAGTA1ZEMREwDwYDVQQHEwhMYXVzYW5u
ZTEZMBcGA1UEChMqd3d3LlRBR0FEQVJULmNvbTEcMBoGA1UECMTUGVyc29uYWwgV2ViIFNlcnZl
cjESMBAGA1UEAxMJMTI3LjAuMCAxMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCjcLrCV1/h
50CuSHjNevhTrRS0bCQ1oCN27c3hTLdDbLVjDNqUJgziTXpowFTUXmM/hrbKwVzM5+I4krwx/6dW
oVVhaGywXkQwN4mQ2rgFvkdM8xIpPKfyVMTLYRQLfd89qLYC8C0SUR3MqzuNRpT71nlalRB9A6Mg
IIx53mf8UQIDAQABoAAwDQYJKoZIhvcNAQEEBQADgYEAht/hvwHdT1Qb5ZPe6EmBbMJe6VozqQT
yzaA2q6+4Y+FzuQ0PT7oePyg22e6HTiEtXRiNGiCXVlceeNxKYFBGoBGVSGOHvauWFGRntErntQ
X7vYKW5XCjHfEpsMwKsj42b4zMFn743IT/LmiC/NsghW3q+UD7AUslld4+XaX68=
-----END NEW CERTIFICATE REQUEST-----
```

## 4 Réponses aux questions

Lors de la connexion sur notre serveur HTTPS à l'aide du navigateur Firefox, le serveur nous affiche une alerte comme le montre la figure 1

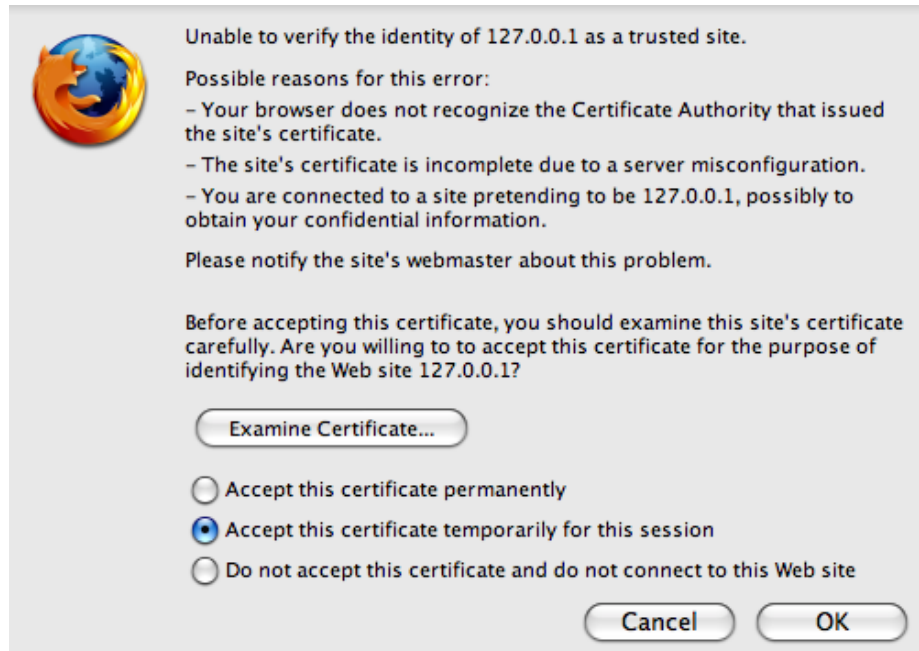


FIG. 1 – Alerte affichée lors de la connexion sur le site sécurisé.

Nous acceptons ce certificat et nous allons voir le site s'afficher. On remarque que le site est sécurisé grâce à l'icone représentant un cadenas (an bas à droite dans Firefox) que l'on peut voir sur la figure 2.

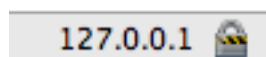


FIG. 2 – Icone dans la barre des tâches du navigateur Firefox.

En cliquant sur le cadenas on peut afficher les informations relatives à la sécurité et donc du certificat que l'on a accepté. La figure 3 nous montre à quoi ressemble cette fenêtre.

La figure 3 nous montre les informations sur le certificat et nous dit que nous faisons confiance au CA mentionné. En cliquant sur le bouton "View" on obtient les informations sur le certificat, exactement les information que nous avons introduites lors de la création à l'aide de *Keytool*. La figure 4 nous montre cette fenêtre.



FIG. 3 – Information sur la sécurité du site.

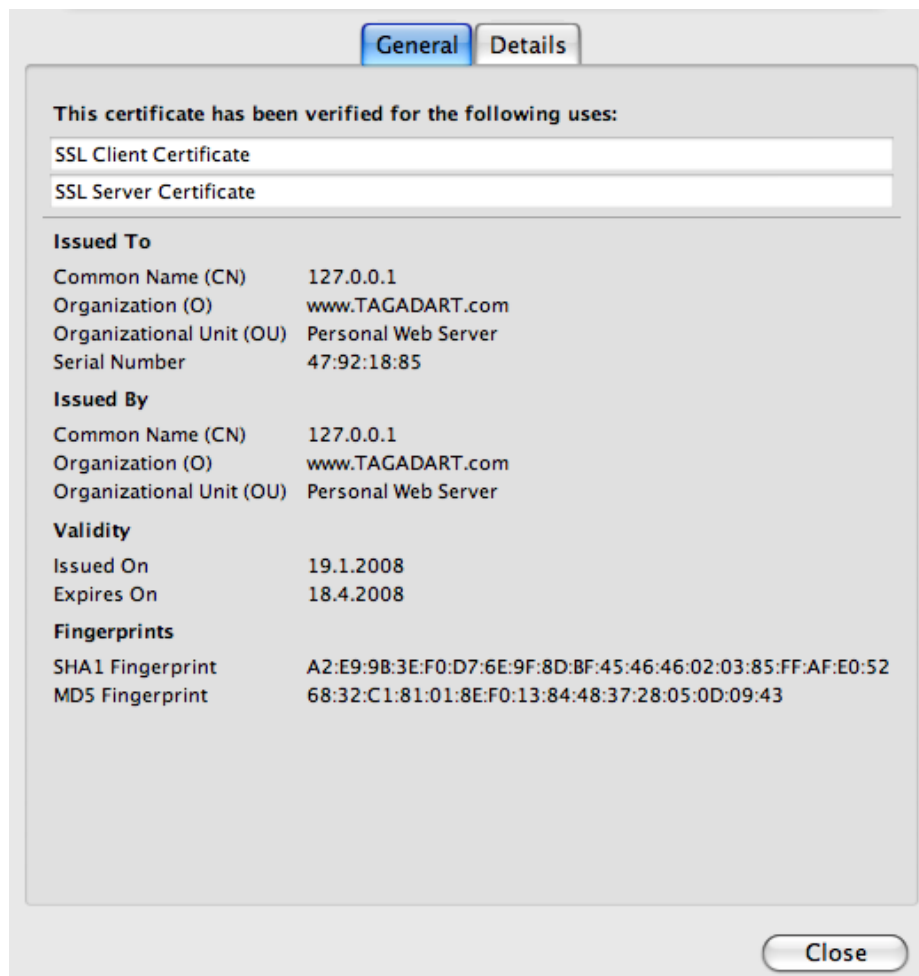


FIG. 4 – Affichage détaillée des informations sur le certificats que nous avons créé.

#### 4.1 En examinant le certificat du serveur, comment pouvez-vous en déduire que celui-ci est auto-signé ?

On le voit bien sur la figure 4 : les champs “Issued To” (distribué à) et “Issued By” (distribué par) sont identiques. On sait dès lors que le certificat est auto-signé. Certain navigateur affiche même directement l’information “Certificat Auto-Signé”. C’est le cas notamment de Safari.

#### 4.2 Pourquoi serait-il intéressant de faire signer le certificat par une entité comme Verisign ?

Verisign est une autorité de certification (CA) : elle émet des certificats qu’elle vend. Ces certificats sont réputés fiables. L’avantage d’avoir un certificat d’une CA reconnue est que les utilisateurs qui se connectent sur le site peuvent, grâce à la renommée de Verisign, savoir que le site utilise une encryption de qualité. C’est donc pour des questions de sécurité et de véracité que nous avons avantage à utiliser les services offerts par une société comme Verisign.

L’utilisateur sera donc mis en confiance et n’aura plus d’avertissement du navigateur comme quoi le certificat est douteux.

## 5 Observations

Nous allons établir deux connexions avec le serveur web différentes et les comparer. La première sans authentification du client et la seconde avec. L’annexe 1, “Communication SSL” illustre les échanges entre le client et le serveur. Des numéros y figurent auxquelles nous ferons référence lors de nos explications.

Nous allons suivre le déroulement partiel des deux types de connexions. Les captures du logiciel Wireshark sont incluses afin de bien voir ce qu’il se passe lors de la connexion.

Les premiers messages échangés (1) font partie du *3 way handshake*. C’est à ce moment que le navigateur web (dans notre exemple Firefox) établit une connexion avec le serveur. Ces messages ne sont pas propres à SSL mais il est intéressant de les mentionner ici.

Le client envoie un message `Client Hello` (2) qui contient la liste des algorithmes et la dernière version de SSL qu’il supporte.

Le serveur effectue les choix de *cipher* ainsi que la version de SSL, la plus récente possible, qu’ils utiliseront pour communiquer.

Le dernier message identique échangé est `Certificate` que le serveur envoie au client. Il est optionnel lorsque l’authentification du client n’est pas nécessaire mais indispensable si l’on désire authentifier le client.

Les étapes suivantes changent en fonction du système d’authentification choisi.

## 5.1 Connexion sans l'authentification client

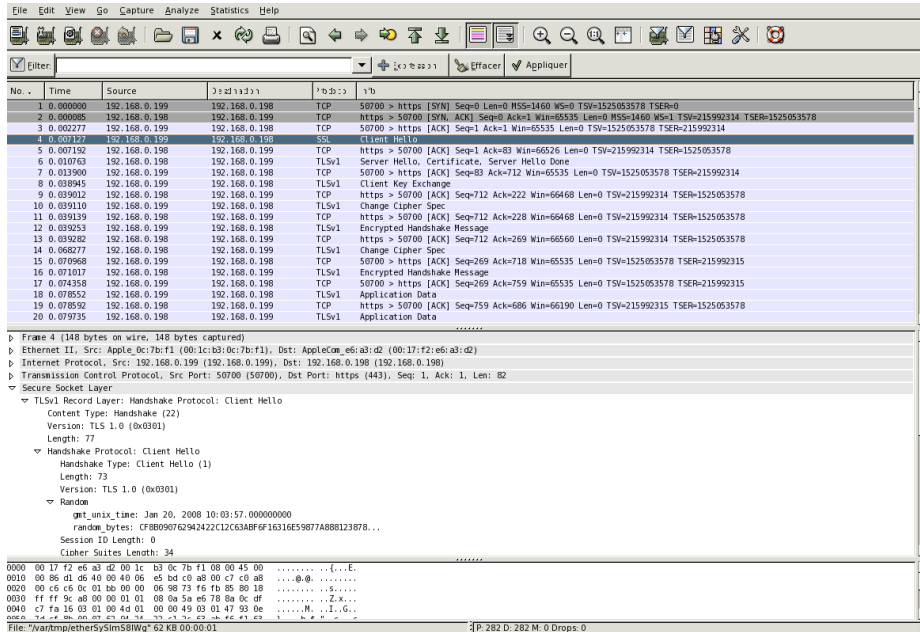


FIG. 5 – Logiciel de capture de paquet Wireshark, communication client-serveur sans authentification du client.

Si l'authentification du client n'est pas nécessaire, le message suivant est Server Hello Done (5), envoyé par le serveur. Il permet d'indiquer au client que les négociations initiales sont terminées de son côté.

Dans les étapes suivantes, le client va échanger des informations pour permettre de créer les clés symétriques (6, Client Key Exchange). Une fois la clé symétrique partagée, le client demande de passer mode sécurisé (7 et 8, Change Cipher Spec). Le serveur envoie aussi (8) un message Change Cipher Spec qui va permettre finalement de communiquer de manière chiffrée. C'est les message de type Application Data qui contiennent les données chiffrées transmises entre le client et le serveur.

## 5.2 Connexion avec l'authentification client activée

Si l'authentification du client est demandée par le serveur, ce dernier va envoyer au client un message Certificate Request (4). Le serveur demande au client de lui envoyer son certificat afin de pouvoir décider si il pourra se connecter ou non.

Le client envoie alors son certificat (9) que le serveur peut vérifier.

Les autres message échangé sont identiques que la version qui ne demande pas d'authentification à l'utilisateur. La négociation et le changement pour passer en mode chiffré se font de la même manière.



File Edit View Go Capture Analyze Statistics Help

Filter

No.	Time	Source	Destination	Protocol	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	49563 > https [SYN] Seq=0 [TCP CHECKSUM INCORRECT] Len=0 MSS=15344 WS=3 TSV=74320970 TS=0
2	0.000000	127.0.0.1	127.0.0.1	TCP	https > 49563 [SYN, ACK] Seq=1451 Win=65535 [TCP CHECKSUM INCORRECT] Len=0 MSS=15344 WS=3
3	0.000001	127.0.0.1	127.0.0.1	TCP	49563 > https [ACK] Seq=1451 Win=524280 [TCP CHECKSUM INCORRECT] Len=0 TSV=74320970 TS=0
4	0.000098	127.0.0.1	127.0.0.1	TCP	[TCP Dup ACK 2#1] Seq=0 > 49563 [ACK] Seq=1 Ack=1 Win=524280 [TCP CHECKSUM INCORRECT] Len=0
5	0.000680	127.0.0.1	127.0.0.1	SSL	Client Hello
6	0.000705	127.0.0.1	127.0.0.1	TCP	https > 49563 [ACK] Seq=125 Win=524280 [TCP CHECKSUM INCORRECT] Len=0 TSV=74320970 TS=0
7	0.000710	127.0.0.1	127.0.0.1	TLSv1	Server Hello, Certificate, Certificate Verify, Change Cipher Spec, Server Hello Done
8	0.000972	127.0.0.1	127.0.0.1	TCP	49563 > https [ACK] Seq=125 Ack=1083 Win=524280 [TCP CHECKSUM INCORRECT] Len=0 TSV=74320970 TS=0
9	0.344898	127.0.0.1	127.0.0.1	TLSv1	Certificate, Client Key Exchange, Certificate Verify, Change Cipher Spec, Encrypted Handshake
10	0.344951	127.0.0.1	127.0.0.1	TCP	https > 49563 [ACK] Seq=1083 Ack=1664 Win=524280 [TCP CHECKSUM INCORRECT] Len=0 TSV=74320970 TS=0
11	0.356693	127.0.0.1	127.0.0.1	TLSv1	Change Cipher Spec
12	0.356688	127.0.0.1	127.0.0.1	TCP	49563 > https [ACK] Seq=1664 Ack=1089 Win=524280 [TCP CHECKSUM INCORRECT] Len=0 TSV=74320970 TS=0
13	0.356809	127.0.0.1	127.0.0.1	TLSv1	Encrypted Handshake Message
14	0.356844	127.0.0.1	127.0.0.1	TCP	49563 > https [ACK] Seq=1664 Ack=1126 Win=524280 [TCP CHECKSUM INCORRECT] Len=0 TSV=74320970 TS=0
15	0.357109	127.0.0.1	127.0.0.1	TLSv1	Application Data
16	0.357109	127.0.0.1	127.0.0.1	TCP	49563 > https [ACK] Seq=1126 Ack=2090 Win=524280 [TCP CHECKSUM INCORRECT] Len=0 TSV=74320970 TS=0
17	0.357623	127.0.0.1	127.0.0.1	TLSv1	Application Data
18	0.357645	127.0.0.1	127.0.0.1	TCP	49563 > https [ACK] Seq=2090 Ack=1162 Win=524280 [TCP CHECKSUM INCORRECT] Len=0 TSV=74320970 TS=0
19	0.357847	127.0.0.1	127.0.0.1	TLSv1	Application Data
20	0.357872	127.0.0.1	127.0.0.1	TCP	49563 > https [ACK] Seq=2090 Ack=1162 Win=524280 [TCP CHECKSUM INCORRECT] Len=0 TSV=74320970 TS=0
21	0.357927	127.0.0.1	127.0.0.1	TLSv1	Application Data
22	0.357940	127.0.0.1	127.0.0.1	TCP	49563 > https [ACK] Seq=2090 Ack=1229 Win=524280 [TCP CHECKSUM INCORRECT] Len=0 TSV=74320970 TS=0
23	0.357984	127.0.0.1	127.0.0.1	TLSv1	Application Data
24	0.358007	127.0.0.1	127.0.0.1	TCP	49563 > https [ACK] Seq=2090 Ack=1252 Win=524280 [TCP CHECKSUM INCORRECT] Len=0 TSV=74320970 TS=0
25	0.358050	127.0.0.1	127.0.0.1	TLSv1	Application Data
26	0.358063	127.0.0.1	127.0.0.1	TCP	49563 > https [ACK] Seq=2090 Ack=1296 Win=524280 [TCP CHECKSUM INCORRECT] Len=0 TSV=74320970 TS=0
27	0.361018	127.0.0.1	127.0.0.1	TLSv1	Application Data

↳ Frame 7 (1138 bytes on wire, 1138 bytes captured)

↳ Null/Loopback

↳ Internet Protocol, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)

↳ Transmission Control Protocol, Src Port: https (443), Dst Port: 49563 (49563), Seq: 1, Ack: 125, Len: 1082

Source port: https (443)

Destination port: 49563 (49563)

Sequence number: 1 (relative sequence number)

[Next sequence number: 1083 (relative sequence number)]

Acknowledgment number: 125 (relative ack number)

Header Length: 32 bytes

```

0000  00 00 00 00 00 04 6e 68 64 04 00 04 06 00 00  ....E..hag...
0010  7f 00 01 01 7f 00 00 01 01 bb 1 c 90 1a c5 74 2f  ...f...f...
0020  45 46 4a 28 1b ff 01 02 63 00 00 01 00 00 00  ...E...f...
0030  2c 4b 7c 02 2c 4b 70 16 03 01 04 35 02 00 00  ...;...f...
0040  03 01 03 47 93 29 63 49 7c a0 c0 fa ce 7b 5b d7  ...;...f...

```

l0d: <live capture in progress> File: Var/tmp/ethergidUyqH8Pab 65 KB
21p: 381 D: 381 M: 0

### 5.3 CIPHER utilisé

Dans la classe `WebServer.java`, nous pouvons changer la liste des *ciphers* utilisé par notre serveur grâce à la constante `CIPHERSUITES`.

heig-vd  
Haute Ecole d'Ingénierie et de Gestion  
du Canton de Vaud

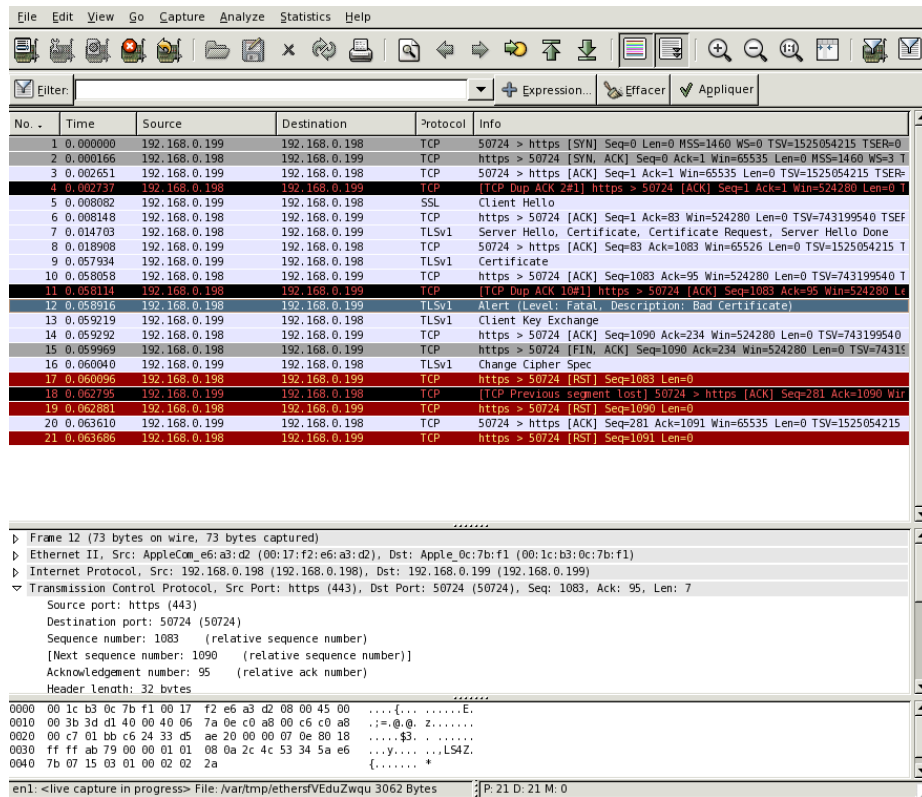


FIG. 7 – Logiciel de capture de paquet Wireshark, communication client-serveur avec authentification du client et erreur (le client ne possède pas le certificat).

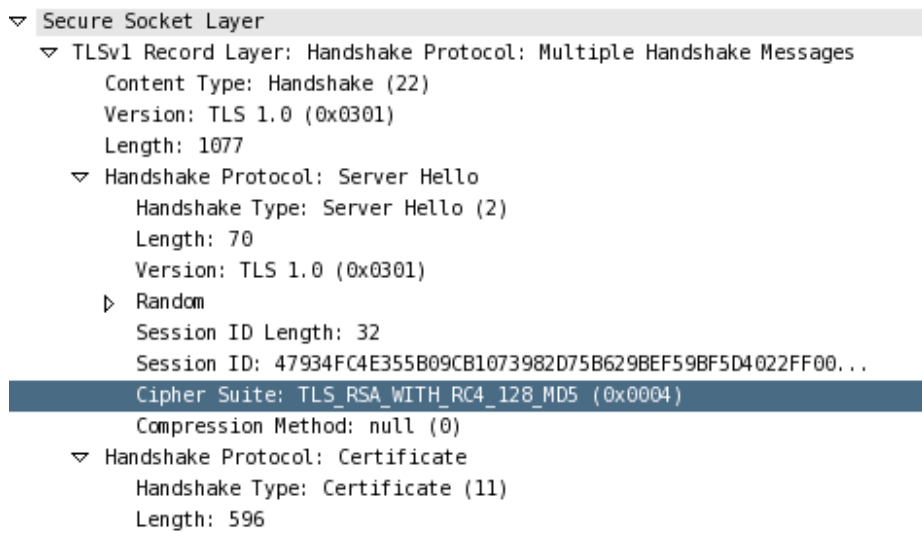


FIG. 8 – Selection du cipher utilisé lors de la communication SSL entre le client et le serveur

## 5.4 Principales différences entre SSL v3 et TLS v1

Pour force l'utilisation de l'une ou l'autre des spécifications, on peut modifier l'objet `SSLContext` comme cela :

```
SSLContext sslContext = SSLContext.getInstance("TLS");
```

Ici on a choisit TLS, mais on peut aussi choisir `SSLv3`. On a récupéré ces informations grâce aux méthodes `getEnabledProtocols()` et `getEnabledCipherSuites()`. Nous les affichons dans la console dont voici un extrait :

```
*****
Enabled Session Creation ? false
Protocols actifs:
- SSLv2Hello
- SSLv3
- TLSv1
Ciphers actifs:
- SSL_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_RSA_WITH_RC4_128_SHA
- SSL_RSA_WITH_RC4_128_MD5
*****
```

Selon la documentation que nous avons trouvé sur internet, il semble que les différences entre `SSLv3` et TLS ne soient pas grandes. On note quand même que ces deux protocole ne sont pas compatible dans les deux sens, mais que TLS soit compatible avec `SSL v3`.

## 5.5 Le message *Certificate Verify*

Le message `Certificate Verify` est utile lorsque l'on désire authentifier le client. Pour se faire, le client signe numériquement à l'aide de sa clé privée ceci afin que le serveur puisse vérifier son identité. Le serveur pourra la vérifier en utilisant la clé publique du client.

## 5.6 Configuration du serveur web avec système d'authentification mais sans chiffrement de données

Pour que notre serveur web effectue uniquement de l'authentification, il faut le lancer avec authentification du client et modifier l'objet serveur appartenant à la classe `SSLServerSocket`. La méthode `setEnabledSessionCreation()` permet d'effectuer cette action.

En effet, si la session n'est pas créée, on aura uniquement l'authentification qui sera effectuée. Le reste des communication ne sera pas chiffrés.

## 6 Conclusion

La découverte et l'implémentation du protocole SSL/TLS est passionnant. Nous n'avions jamais effectué de serveur sécurisé de cette manière. Le langage Java qui met à disposition JSSE nous facilite grandement la vie et rend la création d'un serveur web sécurisé relativement facile.

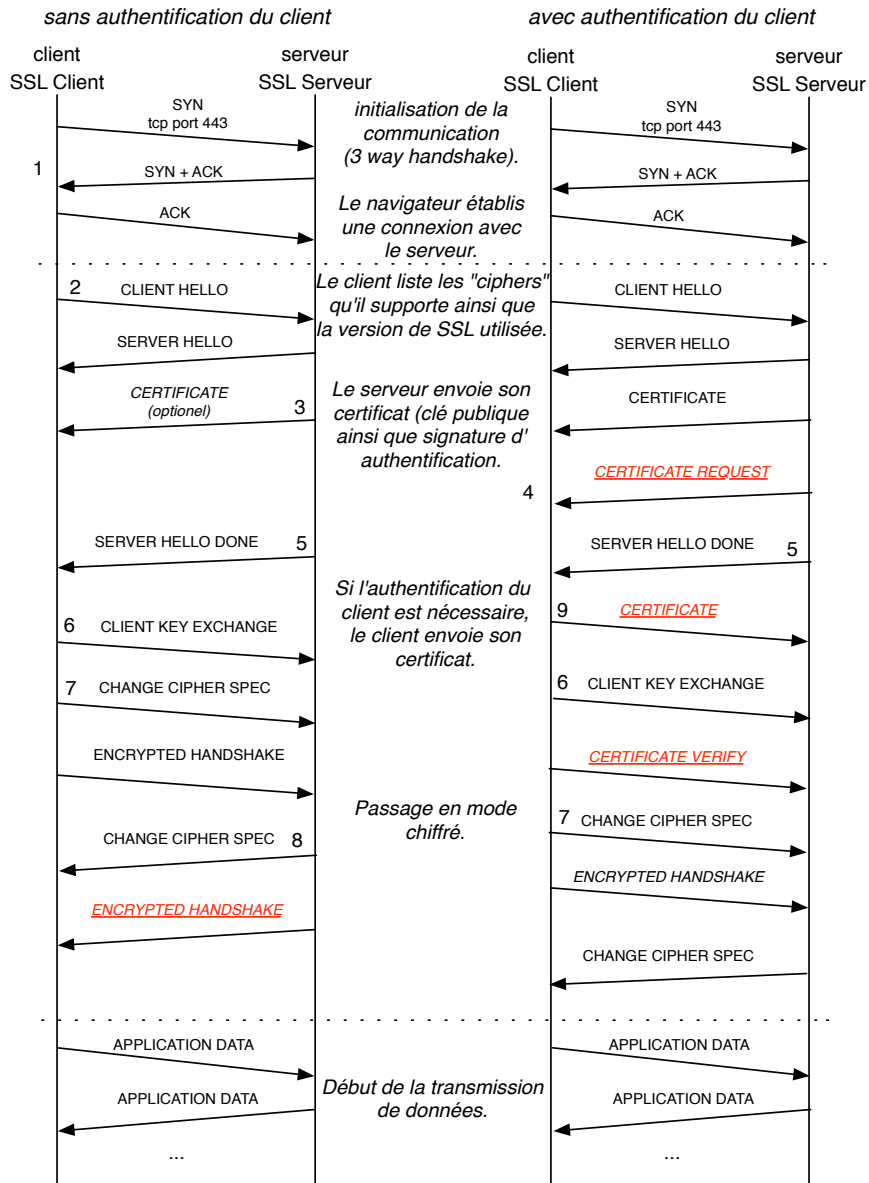
Après avoir lus la documentation et fait quelques tests, il nous fallut simplement du temps pour faire tout ce qui est demandé dans ce laboratoire.

## 7 Références

- <http://www.ietf.org/rfc/rfc2246.txt> TLS Specifications, IETF
- <http://wp.netscape.com/eng/ssl3/draft302.txt> SSL v3 Specifications, Netscape
- <http://java.sun.com/javase/6/docs/technotes/guides/security/jsse/JSSERefGuide.html>  
Java Secure Socket Extension (JSSE) - Reference Guide, Sun.com
- <http://java.sun.com/javase/technologies/security/> Java SE Security, Sun.com
- [http://fr.wikipedia.org/wiki/Transport\\_Layer\\_Security](http://fr.wikipedia.org/wiki/Transport_Layer_Security) TLS et SSL, Wikipedia.org
- <http://www.javaworld.com/javatips/jw-javatip115.html> Secure JavaMail with SSL, Java-World.com
- [http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.itame2.doc\\_5.1/ss7aumst18.htm](http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.itame2.doc_5.1/ss7aumst18.htm)  
The SSL Handshake, IBM.com
- <http://www.commentcamarche.net/crypto/ssl.php3> Cryptographie SSL, CommentCaMarche.net
- <http://www.computing.net/webdevel/wwwboard/forum/439.html> SSL vs TLS, Forum Computing.Net

## 8 Annexe

### Communication SSL Connexion correctement établie



Romain de Wolff - 20 janvier 2008