

## ***T.P : ATELIER SECURITE*** Création certificat électronique

### **T.P 1l'outil OpenSSL**

#### **Objectifs :**

##### **Générales**

- Savoir exploiter les bibliothèques de sécurité en développement,
- Savoir intégrer les mécanismes de sécurité aux applications.

##### **Spécifiques**

- Savoir l'outil OpenSSL, :
- Savoir utiliser et manipuler l'outil OpenSSL, :
- Savoir Créer une IKP (Autorité de Certification "CA", des certificats X509, clés de chiffrement, etc...)
- Savoir utiliser les fonctions de Chiffrement/Déchiffrement (DES, IDEA, RC2, RC4, Blowfish, ...),
- Savoir Calculer des empreintes digitales (MD5, SHA).

#### **Travaux demandés :**

Télécharger l'outil OpenSSL, :

Installer l'outil OpenSSL, :

Voir le « help » de l'ensemble de compte fourni

Définir une Autorité de Certification "CA",

Calcul d'empreintes (MD5, SHA).

Créer des certificats X509,

Créer de clés de chiffrement,

Chiffrer/Déchiffrer (DES, IDEA, RC2, RC4, Blowfish, ...),

#### **Référence :**

1. <http://fr.wikipedia.org/wiki/OpenSSL>
2. <http://fr.wikibooks.org/wiki/OpenSSL>

OpenSSL for windows

<http://www.slproweb.com/products/Win32OpenSSL.html>

<http://gnuwin32.sourceforge.net/packages/openssl.htm>

<http://forum.odette.org/repository/Odette-CA-Tools/view>

Taper help pour afficher le help.

```

C:\OpenSSL-Win32\bin\openssl.exe
OpenSSL> help
openssl:Error: 'help' is an invalid command.

Standard commands:
ca                  ciphers            cms
crl                 dh                 ec
dhparam            dsa                 dsaparam
ecparam            enc                 engine
gendh              gendec             genpkey
nseq               ocsrp              passwd
pkcs7              pkcs8              prime
pkcsutl            rsa                 rsautl
s_time             smime              ts
spkac              verify
x509

Message Digest commands (see the 'dgst' command for more details)
md4                 md5                 mdc2
sha                 sha1                rmd160

Cipher commands (see the 'enc' command for more details)
aes-128-cbc         aes-128-ecb         aes-192-cbc         aes-192-ecb
aes-256-cbc         aes-256-ecb         base64              bf
bf-cbc             bf-ecb             bf-ofb              camellia-128-cbc
camellia-128-ecb   camellia-128-ecb   camellia-192-cbc   camellia-192-ecb
camellia-256-cbc   camellia-256-ecb   cast                cast-cbc
cast5-cbc          cast5-cfb          cast5-ecb          cast5-ofb
des                des-cbc            des-cfb            des-ecb
des-ede            des-ede-cbc        des-ede-cfb        des-ede-ofb
des-ede3           des-ede3-cbc       des-ede3-cfb       des-ede3-ofb
des-ofb            des3               desx                idea
idea-cbc           idea-cfb           idea-ecb           idea-ofb
rc2                rc2-40-cbc         rc2-64-cbc         rc2-cbc
rc2-cfb           rc2-ecb           rc2-ofb            rc4
rc4-40             rc4-cbc            rc4-ofb            seed
seed-cbc          seed-ofb           seed-cfb
OpenSSL>

```

## I. Génération des clés

### On génère la clé privée de notre autorité

Cette clé nous permettra de signer tous les certificats émis par notre AC !  
L'option des3 permet d'ajouter une passphrase pour crypter notre clé.

On peut générer une paire de clés RSA avec la commande *genrsa* de openssl.

```
OpenSSL> genrsa -algorithm <fichier> <taille>
```

où fichier est un nom de fichier de sauvegarde de la clé, et taille et la taille souhaitée exprimée en bits) du modulus de la clé.

Par exemple, pour générer une paire de clés de 1024 bits, stockée dans le fichier ca.key, on tape la commande

```
OpenSSL> genrsa -des3 -out AC / AC.key 1024
```

Une clé privée non cryptée ressemble à ça:

```
OpenSSL> genrsa -out AC / AC.key
```

## II. Créer de clés de chiffrement Chiffrer/Déchiffrer un fichier texte

Pour générer une clé privée RSA de longueur 1024 dans un fichier rsa.priv :

```
OpenSSL> genrsa -out rsa.priv
```

On peut ajouter du chiffrement avec l'option `-algorithme` qui peut être `-des`, `-des3`, etc. Puis extraire la clé publique RSA dans un fichier `rsa.pub` à partir de la clé privée :

```
OpenSSL> rsa -in rsa.priv -pubout -out rsa.pub
```

Voyons comment chiffrer un fichier `fic.txt` en un fichier `fic.enc` en utilisant la clef publique :

```
OpenSSL> rsautl -encrypt -pubin -inkey rsa.pub -in fic.txt -out fic.enc
```

Puis comment le déchiffrer dans un fichier `fic.dec` via la clef privée:

```
OpenSSL> rsautl -decrypt -inkey rsa.priv -in fic.enc -out fic.dec
```

Chiffrer avec la clef privée :

```
OpenSSL> rsautl -sign -inkey rsa.pem -in fichier.txt -out fic.dat
```

Déchiffrer via la clef publique:

```
OpenSSL> rsautl -verify -pubin -inkey rsa.pub -in fic.dat -out decod.txt
```

Soit un fichier donné *fichier\_nom\_eleve* (choisissez un fichier.txt qui contient des données textuelles).

Ecrire la commande qui permet de le chiffrer et produit ainsi un fichier *fichier\_nom\_eleve.enc*

```
Openssl> enc -in fichier.txt -out fichier.enc -e -des3 (c'est pour chiffrer)
```

```
Openssl> enc -in fichier.enc -out fichier.dec -d -des3 (c'est pour déchiffrer)
```

### III. Chiffrement asymétrique

#### Génération de clé privée/publique RSA :

Le format de sortie par défaut est du PEM (Privacy Enhanced Mail).

A l'aide de l'option `-outform` ou `-inform` on peut changer le format. Deux formats sont supportés par cette option PEM et DER.

Vous avez un fichier de configuration de OPENSSL qui s'appelle 'openssl.cnf' ou 'openssl.txt'. Vous pouvez le placer dans le répertoire BIN.

Pour créer la clé privée/publique, vous pouvez taper la commande suivante :

```
OpenSSL> genrsa -out key 1024
```

#### Ou bien, vous pouvez créer la clé en faisant les étapes suivantes :

\* créer un fichier nommé "rand.txt" contenant n'importe quoi ... (exemple : Bonjour)

Ce fichier « rand.txt » va aider l’algorithme RSA à créer votre clé privée/publique.

La clé privée sera stockée sur votre disque dur dans un fichier. Cette clé privée sera chiffrée par un algorithme symétrique (par exemple 3DES). Cet algorithme va chiffrer et protéger la clé privée grâce à une clé de chiffrement symétrique générée par le mot de passe (pass-phrase) que vous allez choisir et confirmer.

```
OpenSSL> genrsa -des3 -out key -rand rand.txt 1024
```

*Ici, vos clé privée/publique sont générées. Elles sont stockées sur votre disque dur dans le fichier key. Les clés ont une longueur chacune de 1024 bits.*

Pour la vérification des clés privée/publique RSA et les visualiser vous pouvez taper :

```
OpenSSL> rsa -in key
```

*Ou bien*

```
OpenSSL> rsa -in key -check
```

*Ou bien*

```
OpenSSL> rsa -in key -check -modulus
```

*Ou bien*

```
OpenSSL> rsa -in key -check -modulus -text
```

*Ou bien, utiliser ‘wordpad’ ou ‘notepad’ de windows pour la visualisation du contenu de la clé.(Nous préfèrons wordpad)*

Génération de la clé publique RSA :

```
OpenSSL> rsa -in key -pubout -out pubkey
```

*Pubkey est un fichier qui va contenir la clé publique. Il a été créé à partir du fichier key qui contient les clés privée/publique.*

*NB : Lorsque vous manipuler la clé publique, utiliser toujours l’option -pubin*

Vérification de la clé publique RSA :

```
OpenSSL> rsa -pubin -in pubkey -text
```

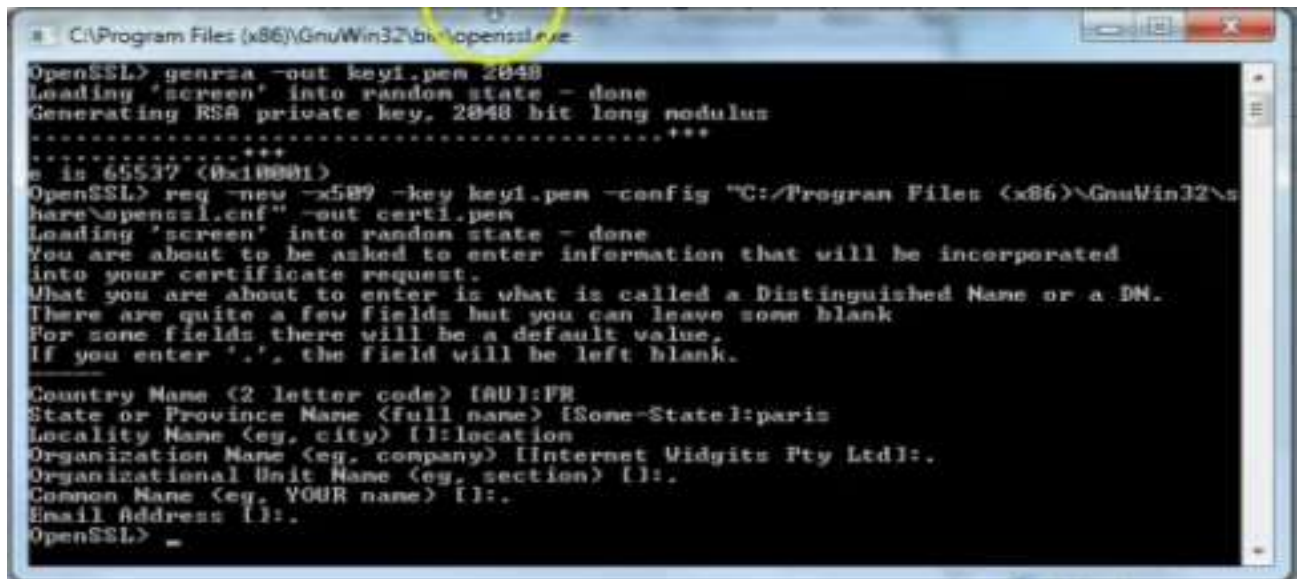
*Ou bien visualiser la clé par ‘wordpad’*

#### **IV. On crée le certificat auto-signé X509**

Le certificat est la clé publique de notre CA signé par... heu, notre CA. Je vous rappelle que pour que votre certificat soit *trust*, il faut normalement le faire

signer par une autre CA reconnu, ce qui vous permet de créer une CA intermédiaire.

```
OpenSSL> req -new -x509 -key.pem -config "C:\OpenSSL-Win32\bin\config.cnf" -out cert.pem
```



```
C:\Program Files (x86)\GnuWin32\bin>openssl.exe
OpenSSL> genrsa -out key1.pem 2048
Loading 'screen' into random state - done
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (b=10001)
OpenSSL> req -new -x509 -key key1.pem -config "C:/Program Files (x86)\GnuWin32\share\openssl.cnf" -out cert1.pem
Loading 'screen' into random state - done
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

-----
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:paris
Locality Name (eg, city) []:location
Organization Name (eg, company) [Internet Widgits Pty Ltd]:.
Organizational Unit Name (eg, section) []:.
Common Name (eg, YOUR name) []:.
Email Address []:.
OpenSSL> _
```