

**TD/TP n°3**  
**Principes et Algorithmes de Cryptographie**  
**TP OpenSSL**

**1. Présentation de OpenSSL**

- Protocole SSL

Le protocole SSL (Secure Socket Layer) a été développé par la société Netscape Communications Corporation pour permettre aux applications client/serveur de communiquer de façon sécurisée. TLS (Transport Layer Security) est une évolution de SSL réalisée par l'IETF.

La version 3 de SSL est utilisée par les navigateurs tels Netscape et Microsoft Internet Explorer depuis leur version 4.

SSL est un protocole qui s'intercale entre TCP/IP et les applications qui s'appuient sur TCP. Une session SSL se déroule en deux temps

1. une phase de poignée de mains (handshake) durant laquelle le client et le serveur s'identifient, conviennent du système de chiffrement et d'une clé qu'ils utiliseront par la suite.
2. la phase de communication proprement dite durant laquelle les données échangées sont compressées, chiffrées et signées.

L'identification durant la poignée de mains est assurée à l'aide de certificats X509.

- OpenSSL

OpenSSL est une boîte à outils cryptographiques implémentant les protocoles SSL et TLS qui offre

1. une bibliothèque de programmation en C permettant de réaliser des applications client/serveur sécurisées s'appuyant sur SSL/TLS.

2. une commande en ligne (OpenSSL) permettant :

- la création de clés RSA, DSA (signature)
- la création de certificats X509
- le calcul d'empreintes (MD5, SHA, RIPEMD160, ...)
- le chiffrement et déchiffrement (RSA, DES, IDEA, RC2, RC4, Blowfish, ...)
- la réalisation de tests de clients et serveurs SSL/TLS
- la signature et le chiffrement de courriers (S/MIME)

Pour connaître toutes les fonctionnalités de openssl : man openssl.

La syntaxe générale de la commande openssl est

\$ openssl <commande> <options>

(le \$ est le prompt du shell)

Dans le texte qui suit, les commandes invoquant openssl supposent que cette commande est dans votre PATH.

## 2. Opérations basiques de OpenSSL

Vous pourrez utiliser les instructions suivantes :

- `$openssl genrsa -out <fichier_rsa.priv> <size>` : génère la clé privée RSA de size bits. La valeur de size : 512, 1024, etc.
- `$openssl rsa -in <fichier_srsa.priv> -des3 -out <fichier.pem>` : chiffre la clé privée avec l'algorithme DES3. Vous pouvez utiliser DES, 3DES, ou IDEA, etc.
- `$openssl rsa -in <fichier_rsa.priv> -pubout -out <fichier_rsa.pub>` : stocke la partie publique dans un fichier à part (création de la clé publique associée à la clé privé dans *fichier.pem*)
- `$openssl enc <-algo> -in <fichier.txt> -out <fichier.enc>` : chiffre fichier.txt avec l'algorithme spécifié (`openssl enc --help` pour avoir la liste des possibilités ou bien `openssl list-cipher-commands`) en un fichier.fic.
- `$openssl enc <-algo> -in <chiffre> -d -out <resultat>` : pour le décryptage
- `$openssl dgst <-algo> -out <sortie> <entree>` : pour hacher un fichier. algo est l'algorithme de hachage (sha, sha1, dss1, md2, md4, md5, ripemd160).
- `$openssl rand -out <clé.key> <numbits>` : pour générer un nombre aléatoire sur numbits (utiliser l'option –base 64 pour la lisibilité).
- `$openssl aes-256-cbc -in <fichier.txt> -out <fichier.enc> -e -k clé.key` : pour chiffrer un fichier avec l'algorithme AES utilisant une clé symétrique.
- `$openssl aes-256-cbc -in <fichier.txt> -out <fichier.enc> -d -k clé.key` : pour déchiffrer un fichier avec l'algorithme AES utilisant une clé symétrique.
- `$openssl rsautl -encrypt -pubin -inkey rsa.pub -in fic.txt -out fic.enc`: chiffrer un fichier fic.txt en un fichier fic.enc
- `$openssl rsautl -decrypt -inkey rsa.priv -in fic.enc -out fic.dec` : déchiffrer dans un fichier fic.dec
- `$openssl rsautl -sign -inkey rsa.priv -in fic.txt -out fic.sig`: pour générer une signature fic.sig pour le fichier fic.txt
- `$openssl rsautl -verify -pubin -inkey rsa.pub -in fic.sig`: pour la vérification d'une signature

## 3. Chiffrement/Déchiffrement avec OpenSSL

### Exercice 1 : Téléchargement et installation

Télécharger OpenSSL pour Windows à partir du site web : <http://osmansalem.free.fr/enseignement/SSIC.html>. Installer l'outil avec la documentation. Changer le répertoire de travail dans le terminal à l'endroit où vous avez installé l'outil, et vérifier que vous êtes bien dédans avec `$openssl version` et `$openssl version -a`.

1. Quelle est la version d'openssl installée ?
2. Lister tous les algorithmes de cryptographie dans l'outil openssl ? comment l'obtenir ?
3. Quel est la signification des options suivantes :
  - i. aes-192-cbc
  - ii. aes-256-ecb
  - iii. cast5-cfb
  - iv. des-ede3-ofb

- v. rc4-40
4. Quel est la signification de base64-encoding?  
\$ openssl enc -base64 -in file.txt ou bien openssl enc -base64 -in file.txt -out file.txt.enc
  5. Avez-vous vraiment besoin (vous et le récepteur) de rappeler de l'algorithme de chiffrement ?
  6. Choisir un fichier dans votre répertoire et générer le digest en SHA1 et MD5.

## **Exercice 2**

1. Générez-vous votre privée avec une longueur de 1024 bits.
2. Dériver la clé publique associée à la clé privée.
3. Générer une clé de 32bit comme une clé de cryptage symétrique pour une utilisation ultérieur.
4. Créer un fichier cigale.txt contenant un message secret.
5. Chiffrer le fichier cigale.txt avec l'algorithme AES 256 bits.
6. Chiffrer la clé de la session (étape 3) avec l'algorithme RSA, en utilisant la clé publique de votre binôme.
7. Générer un digest (hashage) pour le fichier cigale.txt avec md5 et sha1.
8. A quoi sert le digest en général ?
9. Envoyer la clé chiffrer à votre binôme (via : FTP, Telnet, scp, email, clé usb, etc.).
10. Envoyer le fichier chiffré à votre binôme, ainsi que sa signature.
11. Traiter les fichiers reçus par votre binôme (déchiffrer la clé et le fichier), après la vérification de la signature. Extraire le message original.