COURS DU SÉCURITÉ INFORMATIQUE
Série de TD/TP N°3

L'utilisateur étant un maillon souvent faible de la chaîne de sécurité, il est crucial d'être capable de reconnaître qu'une communication est correctement sécurisée. De la même façon, un administrateur doit également être capable de sécuriser ses systèmes d'information. Durant ce TP nous allons donc apprendre à :

- La création de clés RSA, DSA (signature)
- Créer un certificat pour un serveur et le faire signer par une autorité de certification (CA)
- Créer un CA et signer des certificats de clients
- Configuration avancée d'Apache/SSL

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)

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 RSA : clés, chiffrement, déchiffrement, signatures

Attention : dans ce TP vous allez devoir générer beaucoup de fichiers et les utiliser. Ne jamais superimposer des infos dans un fichier déjà créé sans être sûr que c ?est ce que vous voulez !

Le chiffrement et les signatures RSA demandent l ?utilisation d ?une paire de clés : une clé publique et une clé privée. Il faut les générer tout d ?abord.

3.1 Génération d ?une paire de clés

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

$ openssl genrsa -out <fichier> <taille>

où *fichier* est un nom de fichier de sauvegarde de la clé, et *taille* est 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 maCle.pem, on tape la commande

$ openssl genrsa -out maCle.pem 1024

Le fichier obtenu est un fichier au format PEM (Privacy Enhanced Mail, format en base 64), dont voici un exemple :

```
$ cat maCle.pem
——BEGIN RSA PRIVATE KEY——
MIICXAIBAAKBgQCveVjLltevTC5kSAiTYjHMVuAR80DHMLWCp3BOVZ49eXwraXxO
7AfKWpA5g0wFZgZNERIfFYaCnvaQDQA+9BRIfsSSr3oSw0My5SD6eg15v0VmJmvP
d8LgBypJHbr6f5MXWqntvzp0Qvg6ddeNpUIrqkkh4uDfHFDWqyrkQUCvKwIDAQAB
AoGANchUrfnq28DWy0fE0R+cscvC292Z8jN8vrIBWxEk8iSlKU0om6v+a0g8wlP6
3gC6V66uxjY7xxdf7SD+/UykVl4PGFymhLtywSdGlgec3tLgBtV3ytJFilAVDBij
LzQwUegCO4zt1JWYc6vvaVdNyQSaGIIeYGsNDWEYlOtDSlkCQQDVRn9JS15G8p+H
4Z0PbU9ZQg2L1u9/SD/kELVe3Kx1fdHulxH0v8V2AgPdXA29Nhi+TxUtC+V8CMc2
KXmAvFsHAkEA0qBDmjHMDPwcGaqbQ2lymYQIGlZ5TLQFA98Dey2uE+CB6pmS/e/Z
ilu1IaasuE3vBzXfB/JU7DUkV++JQ7TtvQJBAL2s5dUch2sXqlOhjhpDP/eE7CE6
9WLAsbm2Nmd4YJRZYtQLXPfLeeSapC9BCCMHsnfGQ3H9i4mFEQ6VUi7w1Q8CQAQa
pVaS09QI8Y86eM4GdvowzWud9b0d4N8jcFDtIfA3NrDYjzmte8KraMsgEUuCET9F
uHPSL/9uRagE/dq44s0CQCMQU4PMqkMtwzCFsV8ZqLmkDPn1binIAwRLYFcsQRDt
gTi6rycz3Pk1hCVzBfyMd8zwqpwKmR5FoOXuJEv+mVg=
——END RSA PRIVATE KEY——
```

3.2 Visualisation des clés RSA

La commande rsa permet de visualiser le contenu d ?un fichier au format PEM contenant une paire de clés RSA.

```
$ openssl rsa -in <fichier> -text -noout
```

L ?option -text demande l ?affichage décodé de la paire de clés. L ?option -noout supprime la sortie normalement produite par la commande rsa.

Par exemple :

```
$ openssl rsa -in maCle.pem -text -noout
Private-Key : (1024 bit)
modulus :
00 :af :79 :58 :cb :96 :d7 :af :4c :2e :64 :48 :08 :93 :62 :
31 :cc :56 :e0 :11 :f3 :40 :c7 :30 :b5 :82 :a7 :70 :4e :55 :
9e :3d :79 :7c :2b :69 :7c :4e :ec :07 :ca :5a :90 :39 :83 :
4c :05 :66 :06 :4d :11 :12 :1f :15 :86 :82 :9e :f6 :90 :0d :
00 :3e :f4 :14 :48 :7e :c4 :92 :af :7a :12 :c3 :43 :32 :e5 :
20 :fa :7a :0d :79 :bf :45 :66 :26 :6b :cf :77 :c2 :e0 :07 :
2a :49 :1d :ba :fa :7f :93 :17 :5a :a9 :ed :bf :3a :74 :42 :
f8 :3a :75 :d7 :8d :a5 :42 :2b :aa :49 :21 :e2 :e0 :df :1c :
50 :d6 :ab :2a :e4 :41 :40 :af :2b
```

$ openssl rsa -in maCle.pem -text -noout
publicExponent : 65537 (0x10001)
privateExponent :
35 :c8 :54 :ad :f9 :ea :db :c0 :d6 :cb :47 :c4 :d1 :1f :9c :
b1 :cb :c2 :db :dd :99 :f2 :33 :7c :be :b2 :01 :5b :11 :24 :
f2 :24 :a5 :29 :4d :28 :9b :ab :fe :6b :48 :3c :c2 :53 :fa :
de :00 :ba :57 :ae :ae :c6 :36 :3b :c7 :17 :5f :ed :20 :fe :
fd :4c :a4 :56 :5e :0f :18 :5c :a6 :84 :bb :72 :c1 :27 :46 :
96 :07 :9c :de :d2 :e0 :06 :d5 :77 :ca :d2 :45 :8a :50 :15 :
0c :18 :a3 :2f :34 :30 :51 :e8 :02 :3b :8c :ed :d4 :95 :98 :
73 :ab :ef :69 :57 :4d :c9 :04 :9a :18 :82 :1e :60 :6b :0d :
0d :61 :18 :94 :eb :43 :4a :59
prime1 :
00 :d5 :46 :7f :49 :4b :5e :46 :f2 :9f :87 :e1 :9d :0f :6d :
4f :59 :42 :0d :8b :d6 :ef :7f :48 :3f :e4 :10 :b5 :5e :dc :
ac :75 :7d :d1 :ee :97 :11 :f4 :bf :c5 :76 :02 :03 :dd :5c :
0d :bd :36 :18 :be :4f :15 :2d :0b :e5 :7c :08 :c7 :36 :29 :
79 :80 :bc :5b :07
prime2 :
00 :d2 :a0 :43 :9a :31 :cc :0c :fc :1c :19 :aa :9b :43 :69 :
72 :99 :84 :08 :1a :56 :79 :4c :b4 :05 :03 :df :03 :7b :2d :
ae :13 :e0 :81 :ea :99 :92 :fd :ef :d9 :8a :5b :b5 :21 :a6 :
ac :b8 :4d :ef :07 :35 :df :07 :f2 :54 :ec :35 :24 :57 :ef :
89 :43 :b4 :ed :bd
exponent1 :
00 :bd :ac :e5 :d5 :1c :87 :6b :17 :aa :53 :a1 :8e :1a :43 :
3f :f7 :84 :ec :21 :3a :f5 :62 :c0 :b1 :b9 :b6 :36 :67 :78 :
60 :94 :59 :62 :d4 :0b :5c :f7 :cb :79 :e4 :9a :a4 :2f :41 :
08 :23 :07 :b2 :77 :c6 :43 :71 :fd :8b :89 :85 :11 :0e :95 :
52 :2e :f0 :d5 :0f
exponent2 :
04 :1a :a5 :56 :92 :d3 :d4 :08 :f1 :8f :3a :78 :ce :06 :76 :
fa :30 :cd :6b :9d :f5 :bd :1d :e0 :df :23 :70 :50 :ed :21 :
f0 :37 :36 :b0 :d8 :8f :39 :ad :7b :c2 :ab :68 :cb :20 :11 :
4b :82 :11 :3f :45 :b8 :73 :d2 :2f :ff :6e :45 :a8 :04 :fd :
da :b8 :e2 :cd
coefficient :
23 :10 :53 :83 :cc :aa :43 :2d :c3 :30 :85 :b1 :5f :19 :a8 :
b9 :a4 :0c :f9 :f5 :6e :29 :c8 :03 :04 :4b :60 :57 :2c :41 :
10 :ed :81 :38 :ba :af :27 :33 :dc :f9 :35 :84 :25 :73 :05 :
fc :8c :77 :cc :f0 :aa :9c :0a :99 :1e :45 :a0 :e5 :ee :24 :
4b :fe :99 :58

3.3 Chiffrement d ?un fichier de clés RSA

Il n ?est pas prudent de laisser une paire de clés en clair (surtout la partie privée). Avec la commande rsa, il est possible de chiffrer une paire de clé. Pour cela trois options sont possibles qui précisent l ?algorithme de chiffrement symétrique à utiliser : -des, -des3 et -idea.

```
$ openssl genrsa -out maCle.prem 2048
writing RSA key
```

Une phrase de passe est demandée deux fois pour générer une clé symétrique protégeant l ?accès à la clé.

Exercice 1.

Avec la commande cat observez-le contenu du fichier maCle.pem. Utilisez à nouveau la commande rsa pour visualiser le contenu de la clé.

3.4 Exportation de la partie publique

La partie publique d ?une paire de clés RSA est publique, et à ce titre peut être communiquée à n ?importe qui. Le fichier maCle.pem contient la partie privée de la clé, et ne peut donc pas être communiqué tel quel (même s ?il est chiffré). Avec l ?option -pubout on peut exporter la partie publique d ?une clé.

```
$ openssl rsa -in maCle.prem -pubout -out maClepublique.pem
```

Exercice 2.

Question 1. Notez le contenu du fichier maClePublique.pem. Remarquez les marqueurs de début et de fin.

Question 2. Avec la commande rsa visualisez la clé publique. Attention vous devez préciser l ?option -pubin, puisque seule la partie publique figure dans le fichier maClePublique.pem.

3.5 Chiffrement/déchiffrement de données avec RSA

On peut chiffrer des données avec une clé RSA. Pour cela on utilise la commande *rsautl*

```
$ openssl rsautl -encrypt -in <fichier_entree> -out <fichier_sortie> -inkey <cle> -pubin
```

où

- ⋆ *fichier_entree* est le fichier des données à chiffrer. Attention, le fichier des données à chiffrer ne doit pas avoir une taille excessive (ne doit pas dépasser 116 octets pour une clé de 1024 bits).
- ⋆ *cle* est le fichier contenant la clé RSA. Si ce fichier ne contient que la parte publique de la clé, il faut rajouter l ?option -pubin.
- ⋆ *fichier_sortie* est le fichier de données chiffré.

Pour déchiffrer on remplace l ?option -encrypt par -decrypt. Le fichier contenant la clé doit évidemment contenir la partie privée.

Exercice 3.

Chiffrez le fichier de votre choix avec le système symétrique de votre choix. Chiffrez la clé ou le mot de passe utilisé(e) avec la clé publique de votre destinataire (demandez-lui sa clé publique si vous ne l ?avez pas). Envoyez-lui le mot de passe chiffré ainsi que le fichier chiffré.

Demandez à un autre collègue (ou binôme) qu ?il vous envoie le fichier contenant sa clé publique à 1024 bits. Utilisez la clé pour chiffrer un message à choix et envoyez à votre collègue le chiffré. L ?instruction pour déchiffrer est :

$ openssl rsautl -decrypt -in <fichier du chiffrement> -inkey <fichier avec la cle de dechiffrement> -out <fichier de sortie>

Envoyez le chiffré à un autre collègue et recevez le chiffré d ?un de vos collègues. Essayez de déchiffrer son chiffré (qui n ?est pas chiffré avec votre clé publique) avec la clé privée.

3.6 Signature de fichiers

Les clés RSA ne sont pas juste utiles pour chiffrer des messages, sinon également pour calculer des signatures sur des messages. Rappel : pour signer un message avec un schéma à clé publique (comme RSA) on utilise la clé publique ou la clé privée ?

Choisissez un texte clair de taille petite. Calculez une signature RSA pour ce message en utilisant l ?instruction :

$ openssl rsautl -sign -in <fichier en entree> -inkey <fichier de la clé> -out <fichier de sortie>

Envoyez à un de vos collègues les données suivantes : le fichier contenant la clé utilisée pour la vérification des signatures, le message que vous avez signé et le fichier contenant la signature. Vous allez recevoir les mêmes donnés. Vérifiez la validité de la signature en utilisant l ?instruction :

$ openssl rsautl -verify -in <signature> -pubin -inkey <fichier de la cle>

Si la vérification s ?est bien passée, on affichera le texte clair auquel corresponde la signature. Sinon, le résultat sera une erreur.

4 Certificat avec OpenSSL

Une autorité de certificat AC est une entité qui signe des certificats numériques. Un exemple d'AC bien connu est Let's Encrypt. De nombreux sites Web sur Internet utilisent des certificats pour leurs connexions HTTPS qui ont été signés par Verisign.

Or si vous êtes votre propre autorité, personne ne vous fera confiance. En fait ceci n'est utile que dans deux cas. Le premier pour faire des tests sans devoir payer (relativement cher). Le deuxième pour avoir des certificats SSL sur un parc de clients que vous pouvez maîtriser. Par exemple sur un intranet pour pouvez exiger des clients qu'ils installent votre autorité.

4.1 Générer et faire certifier un certificat pour votre serveur web

Nous allons utiliser l'ordinateur serveur pour configurer et sécuriser Apache. Apache est un serveur HTTP très configurable et l'un des plus utilisé pour héberger les sites web sur Internet. Toute la configuration de Apache se trouve dans le répertoire /etc/apache2. Ce repertoire est organisé de la façon suivante :

— Activez le module apache en tapant :

```
$ sudo a2enmod ssl
```

— Après avoir activé SSL, vous devrez redémarrer le serveur Web pour que le changement soit reconnu :

```
$ sudo service apache2 restart
```

— Commençons par créer un sous-répertoire dans la hiérarchie de configuration d'Apache pour placer les fichiers de certificats que nous allons créer :
NB : Vous pouvez utiliser la clé et le certificat généré dans le TP précèdent :

```
$ sudo mkdir /etc/apache2/ssl
```

— Maintenant que nous disposons d'un emplacement pour placer notre clé et notre certificat, nous pouvons les créer en une seule étape en tapant :

```
$ sudo openssl req -x509 -nodes -days 365 -newkey rsa :2048 -keyout /etc/apache2/ssl/apache.key -out /etc/apache2/ssl/apache.crt
```

— La partie des questions ressemble à ceci :

```
Country Name (2 letter code) [AU] :MA
State or Province Name (full name)
[Some-State] :Casablanca
Locality Name (eg, city) [] :Casablanca City
Organization Name (eg, company) [Internet Widgits Pty Ltd] :AIAC
Organizational Unit Name (eg, section) [] :Genie Informatique
Common Name (e.g. server FQDN or YOUR name) [] :your_domain.com
Email Address [] :your_email@domain.com
```

— En fin de compte, ça ressemblera à quelque chose comme ça. Les entrées ont été modifiées à partir du fichier original :

```
<IfModule mod_ssl.c>
    <VirtualHost _default_ :443>
        ServerAdmin admin@example.com
        ServerName your_domain.com
        ServerAlias www.your_domain.com
        DocumentRoot /var/www/html
        ErrorLog $APACHE_LOG_DIR/error.log
        CustomLog $APACHE_LOG_DIR/access.log combined
        SSLEngine on
        SSLCertificateFile /etc/apache2/ssl/apache.crt
        SSLCertificateKeyFile /etc/apache2/ssl/apache.key
        <FilesMatch "(cgi|shtml|phtml|php)$">
            SSLOptions +StdEnvVars
        </FilesMatch>
        <Directory /var/www/html>
            SSLOptions +StdEnvVars
            DirectoryIndex index.php
            AllowOverride All
            Order allow,deny
            Allow from all
        </Directory>
        BrowserMatch "MSIE [2-6]"
        nokeepalive ssl-unclean-shutdown
        downgrade-1.0 force-response-1.0
        BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown
    </VirtualHost>
</IfModule>
```

— Sauvegardez et quittez le fichier lorsque vous avez terminé. Maintenant que nous avons configuré notre serveur virtuel SSL, nous devons l'activer.

```
$ sudo a2ensite default-ssl.conf
```

— Vous devez ensuite redémarrer Apache pour charger notre nouveau fichier d'hôte virtuel :

```
$ sudo service apache2 restart
```

Exercice 4.

Générer votre propre certificat et mettre votre site localhost en HTTPS.