I - Use of openSSL to illustrate some operations of cryptography

1 - Presentation of openSSL

The SSL (Secure Socket Layer) Protocol has been developed by the Netscape Communications Corporation to allow the client/server applications to communicate in a protected way.
TLS (Transport Layer Security) is an evolution of SSL carried out by the IETF (Internet Engineering Task Force).
SSL is a protocol which is located between TCP/IP and the applications which are based on TCP. An SSL session proceeds in two times:

- a handshake phase during which the client and the server are identified, then they negotiate the system of ciphering and a key which they will use thereafter.
- the communication phase itself during which the exchanged data are compressed, ciphered and signed.
- The identification phase during which the handshake is ensured using X509 certificates.

openSSL is a cryptographic toolbox implementing SSL and TLS protocols. These protocols offer:

- a C programming library allowing to achieve protected client/server applications based on SSL/TLS.
- an in line control (openssl) allowing:
    - the creation of RSA, DSA keys (signature)
    - the creation of X509 certificates
    - the calculation of prints (MD5, SHA,…)
    - ciphering and deciphering (DES, IDEA, RC2, RC4,…)
    - the realization of tests of SSL/TLS clients and servers
    - the signature and the ciphering of mails (S/MIME)

2 - Some simple examples (see also appendix 2)

Symmetrical ciphering with RC4 algorithm:

The file obtained is quantified using an rc4 flow initialized thanks to a password. The password will be: **secret**.

Edit the file texte.clair according to (or copy-paste three lines of text from another document into texte.clair):

```
Je m'en allais, les poings dans mes poches crevées
Mon paletot aussi devenait idéal
J'allais sous le ciel, Muse ! et j'étais ton féal.
```

Openssl accepts this input file and creates an rc4-ciphered output file.

>openssl rc4 –in texte.clair –out texte.chiffre

(The password will be: **secret**.)

Do again the same operation by using the same password:

>openssl rc4 –in texte.clair –out texte.chiffre2

Compare texte.chiffre and texte.chiffre2, they are different, why?

More information on rc4

The password is not used directly as a key. It is used for generating the "salt" with the assistance of a random number. Thus two messages ciphered with the same password will not be identical!

To see these various parameters, let's cipher with the option – p twice and then compare:
>openssl rc4 –in texte.clair –out texte.chiffre -p

The ciphered file is deciphered by:

>openssl rc4 –d –in texte.chiffre –out texte.out

Check that texte.clair and texte.out are similar.

>diff texte.clair texte.out (in Linux)

>comp texte.clair texte.out (in Windows)

base 64 coding

Code texte.clair in base 64 into texte.base64

Then to decode texte.base64 into texte.clair3.

Why base 64 cannot be considered as an encryption algorithm?

3 - Hashing techniques

The functions of hashing are one-way functions (as the CRC) which make it possible to calculate a check code (print) related to the message to be transmitted. It is impossible (or at least very difficult) to modify the message so as to obtain the same print.
The two most used methods are md5 and SHA.
Example:

>openssl md5 texte.clair

Password in linux
The method of ciphering of the passwords in Linux is DES (a symmetrical algorithm). The password is not stored on the system. Instead of that, a print of the password is stored.
Extract of file shadow (in the etc/shadow directory):

```
Root :$1$3ku0ivnQ$SuYbwLu.9PvhrgzDhQer1. :13097 :0 :99999 :7 :::
```

Salt Print

The following control makes it possible to generate the print of the password **`toto`** with the salt **`3ku0ivnQ`**

>openssl passwd -1 - salt 3ku0ivnQ toto

Do the experience again with your own **`shadow`** file and the **`root`** password.

4 - Asymmetrical methods with a couple of public/private key.

The most usually used methods are:
Diffie-Hellman: Used to generate the same secret key both for the transmitter and the receiver without transmitting it.
RSA: (Rivest - Shamir - Adleman)
DSA: (Digital Signature Algorithm)

Generation of a couple of keys
Generate a key file including/understanding a private key of 1024 bits.

>Openssl genrsa –out cle.priv.pem 1024

This file contains the private key. Now let us generate the public key in a file .pub

>Openssl rsa –in cle.priv.pem –pubout –out cle.pub.pem

Applications of cryptography with a public key are numerous. Let us see how to cipher a text before transmitting it:

Cipher using the public key:

>openssl rsautl -inkey cle.pub.pem -pubin –in texte.clair -out texte.cryptpub -encrypt

It is necessary, for deciphering, to use the private key: (it is not necessary to specify that it is a private key: it is the default option.)

>openssl rsautl -inkey cle.priv.pem -in texte.cryptpub -decrypt

NB : Of course in an actual mailing application, we don’t encrypt and decrypt on the same station.

5 - Digital signature

Signing a document digitally consists in calculating a print and protecting this print by ciphering it with its own private key. Any receiver will be able by using the public key of the transmitter (authentication) to recover the print and thus to validate the document.

There are two stages:

- Calculation of print
- Ciphering of this print

>openssl md5 texte.clair > texte.emp

>openssl rsautl -sign -in texte.emp -inkey cle. priv.pem -out texte.sign

Application:

Recover from the ftp server (`ftp://ftp.gtrgrenoble.fr/TpL3/crypto`) the files **`texte_prof.clair`**, **`text_prof.sign`** and the public key of the teacher (**`cleprof.pub.pem`**)

The file **`text_prof.sign`** is the signature of the text **`texte_prof.clair`**

Thanks to the public key of the sender, the receiver can:

- decrypt the received hash,
- create his own version of the hash for the received information,
- compare the two hashes (received and calculated).

Decipher this signature using the public key of the teacher and check the print of the file **`texte_prof.clair`**.

You can use a control such as

`Openssl rsautl –verify –in text_prof.sign –inkey cleprof.pub.pem –pubin > resul.out`

Note:

In practice, it is more interesting to generate the signature in only one stage:

>openssl dgst –md5 –sign cle.priv.pem –out texte.sign texte.clair

ANNEXE 3 : EXEMPLE D'UTILISATION DE OPENSSL

Certaines clés utiles à ces exemples peuvent être téléchargée de ftp:172.16.6.100

`openssl rand –out cle_session.bin 16`	Génère une clé de session aléatoire de 16 octets et la range dans un fichier `cle_session.bin`
`openssl enc –aes256 –pass file:cle_session.bin –in message.txt –out message_crypte.bin`	Chiffre en AES (est-ce un algorithme symétrique ou assymétrique ?) 256 bits le fichier `message.txt` et le range dans `message_crypte.bin`
`openssl dgst –md5 –binary –out resume.bin message_crypte.bin`	Calcule un résumé MD5 de `message_crypte.bin` et le range dans `resume.bin`
`openssl rsautl –sign –inkey alice_rsa_pub_prive –in resume.bin –out signature.bin`	Signe `resume.bin` avec la clé privée d'Alice et le range dans `signature.bin` (que signifie `rsautl` ? Qu'est-ce que rsa ?)
`openssl rsautl –encrypt –pubin –inkey bob_rsa_pub.pem –in cle_session.bin –out cle_session_crypte.bin`	Chiffre avec RSA la clé de session `cle_session.bin` en utilisant la clé publique de Bob ; le résultat est rangé dans `cle_session_crypte.bin`
`tar zcvf message.tgz message_crypte.bin cle_session_crypte.bin signature.bin`	Crée une archive compressée `message.tgz` contenant les fichiers `message_crypte.bin`, `signature.bin` et `cle_session.bin` Quels sont les contenus des trois fichiers envoyés ? Pourquoi ont-ils été envoyés ?

`tar zxvf message.tgz`	Désarchive le fichier message.tgz
`openssl rsautl –verify –pubin –inkey alice_rsa_pub.pem –in signature.bin –out resume1.bin`	Déchiffre le résumé avec la clé publique de Alice pour retrouver le résumé
`openssl dgst –md5 –binary –out resume1.bin message_crypte.bin`	Calcule le résumé MD5 du message chiffré
`Diff -s resume.bin resume1.bin` (sous Windows `comp resume.bin resume1.bin`	Compare le résumé reçu et le résumé calculé
`openssl rsautl –decrypt -inkey bob_rsa_pub_priv.pem –in cle_session_crypte.bin –out cle_session.bin`	Décrypte la clé de session à l'aide de la clé privée de Bob (le mot de passe associé à la clé sera demandé)
`openssl enc –d –aes256 –pass file:cle_session.bin –in message_crypte.bin –out message.txt`	Décrypte le message avec la clé de session