TP3 : de Crypto avec OpenSSL

Objectif:

Ces travaux pratiques sont basés sur la suite logicielle OpenSSL.

Vous allez travailler dans le répertoire BIN d'OpenSSL pour ne pas être obligé de taper à chaque fois le chemin d'accès aux fichiers.

OpenSSL offre un mode commande où l'on peut interagir avec l'API. C'est ce mode que l'on utilisera tout simplement en tapant openssl (en en cliquant deux fois sur openssl.exe). Pour accéder à l'usage des différentes commandes et avoir le help (aide en ligne), il suffira lorsqu'on est sur openssl de taper le nom de la commande suivie de –help.

L'objectif de ces travaux pratiques est de vous familiariser avec les services de base de la sécurité. Notamment le chiffrement symétrique, le chiffrement asymétrique, le hachage, la signature numérique et sa vérification, et la certification.

Il vous ait demandé dans un premier temps de suivre pas à pas les exercices qui vous sont proposés et par la suite vous avez quartier libre pour les exercices qui vous feront plaisir.

Notation : toutes les commandes et arguments associés seront notés dans la suite en ***italique gras.***

`exemple : openssl> ` ***`enc -in messageclair -out messagechiffre -e -des3`***

1 CHIFFREMENT SYMETRIQUE

La commande qui vous permet d'utiliser le chiffrement symétrique est la commande *enc (en tapant **enc –help** vous aurez toutes les options associées)*

Question 1 :

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***

On peut alors vérifier que le message ***fichier_nom_eleve.enc*** est bien inintelligible. Il est chiffré.

Ce même message est transmis à votre camarade qui pourra également le déchiffrer.

On peut alors vérifier que le message ainsi déchiffré est bien identique à ***fichier_nom_eleve*** ...

Il vous est bien sûr de comprendre l'ensemble des manipulations associées.

Il vous est demandé de diversifier les algorithmes. D'utiliser également l'option –a pour produire un fichier chiffré lisible donc codé en base64 (lisible ne veut pas dire en clair).

Solution :

`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)`

Leur demander de diversifier leurs algorithmes. D'utiliser par exemple DES, RC2, RC4, AES, BF (BlueFich) , …

Question 2 :

Pourquoi quand on applique deux fois la commande ***enc*** une fois au fichier en clair et une fois au fichier chiffré on n'obtient pas un résultat en clair.

Réponse :

Au fichier chiffré on rajoute des entêtes qui ne font pas partie des données. Ainsi qu'on applique deux fois de suite la commande avec l'option ***–e*** il chiffre et les données et les entêtes. Quand on utilise l'option –d il applique la même fonction sauf qu'il extrait du chiffrement les données structurelles.

2 CHIFFREMENT ASYMETRIQUE

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 suivant es :

* 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'

CHIFFREMENT/DECHIFFREMENT DE DONNEES AVEC RSA

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

```
Openssl> rsautl -encrypt -in fichier_entrée -pubin -inkey clépublique
         -out fichier_sortie
```

où

- *fichier_entrée* 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).

- *clépublique* 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 obligatoirement contenir la partie privée.

3 SIGNATURE DE FICHIERS

Il n'est possible de signer que de petits documents. Pour signer un gros document on calcule d'abord une empreinte de ce document. La commande dgst permet de le faire.

```
Openssl> dgst -fonctionhachage  -out fichierempreinte    fichier_entrée
```

où fonction*hachage* est une fonction de hachage.

Avec `openssl`, plusieurs fonctions de hachage sont proposées dont

- MD5 (option `-md5`), qui calcule des empreintes de 128 bits,
- SHA1 (option `-sha1`), qui calcule des empreintes de 160 bits,
- RIPEMD160 (option `-ripemd160`), qui calcule des empreintes de 160 bits.

Fichierempreine : c'est le fichier empreinte du document. Il est de petite taille (128 bits, 160 bits, …)

Signer un document revient à signer son empreinte. Pour cela, on utilise l'option -sign de la commande rsautl

```
Openssl> rsautl -sign   -in fichierempreinte -inkey cléprivée  -out fichiersignature
```

Fichiersignature : c'est la signature numérique.

et pour vérifier la signature

```
openssl> rsautl -verify  -in fichiersignature –pubin -inkey clepublique
          -out fichierempreinte2
```

Il reste ensuite à vérifier que l'empreinte ainsi produite est la même que celle que l'on peut calculer avec.