TP1 : Chiffrement Symétrique

But du TP

- Utiliser openssl pour chiffrer/déchiffrer
- Utiliser openssl pour générer des signatures

Présentation de OpenSSL

OpenSSL est une boîte à outils cryptographiques implémentant les protocoles SSL et TLS. Elle 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 :
   - le chiffrement et déchiffrement (DES, IDEA, RC2, RC4, Blowfish, ...)
   - la création de clefs RSA
   - le calcul d'empreintes (MD5, SHA, RIPEMD160, ...)
   - la création de certificats X509
   - la signature et le chiffrement de courriers (S/MIME)
   - la réalisation de tests de clients et serveurs SSL/TLS

Pour connaître toutes les fonctionnalités de openSSL :

```
man openssl
```

Syntaxe : openssl étant une boîte à outils, il faut penser à spécifier quelle sous-commande vous souhaitez utiliser, ainsi que ses options :

```
openssl <commande> <options>
```

Exercice 1: Chiffrement symétrique

De nombreux systèmes de chiffrement symétrique existent et **openssl** permet de les manipuler facilement. C'est la commande **enc** qui permet de chiffrer/déchiffrer avec

```
Openssl.
openssl enc <options>
```

Parmi les options, on doit indiquer le système de chiffrement à choisir dans la liste :

```
base64          Base 64
bf-cbc          Blowfish in CBC mode
bf              Alias for bf-cbc
bf-cfb          Blowfish in CFB mode
bf-ecb          Blowfish in ECB mode
bf-ofb          Blowfish in OFB mode
cast-cbc        CAST in CBC mode
...
```

Pour connaître la liste complète, consultez la section "Supported Ciphers" de la page `man` concernant *`enc`*: `man enc`

Remarque: base64 n'est pas un système de chiffrement, mais un codage des fichiers binaires avec 64 caractères ASCII. Ce codage est utilisé en particulier pour la transmission de fichiers binaires par courrier électronique.

Lab1: Chiffrement avec mot de passe

Q 1. Créez un fichier toto et chiffrez-le avec le système Blowfish en mode CBC. Le chiffrement se fait à l'aide d'une clef, préalablement générée depuis un mot de passe. Le fichier chiffré se nommera toto.chiffre.

```
$ echo "Je suis debutant(e)" > toto.cl //creation du fichier toto
$ openssl enc -bf-cbc -in toto -out toto.ch //choisir le mot de passe gelt
```

Q 2. Déchiffrez ce fichier et enregistrez le résultat dans un fichier toto.dechiffre. Vérifiez que l'opération s'est bien déroulée.

```
$ openssl enc -bf-cbc -d -in toto.ch -out toto.de//mettre le mot de passe gelt
```

Q 3. Chiffrez le fichier de votre choix avec le système de votre choix dans le mode de votre choix, puis déchiffrez-le.

```
$ echo "Je sais maintenant chiffrer" > fichier.cl
$ openssl enc -e -aes-256-cbc -in fichier.cl -out fichier.ch
```

- */* *enc* : on précise qu'on va utiliser un algorithme de chiffrement*
- *-e* : on chiffre un fichier*
- *-aes-256-cbc* : l'algorithme, ici AES*
- *-in* : le nom du fichier à chiffrer*
- *-out* : le nom de sortie du fichier chiffré*/*

```
$ openssl enc -d -aes-256-cbc -in fichier.ch -out fichier.de
```

Q 4.Tentez maintenant de déchiffrer un cryptogramme en utilisant un mauvais mot de passe. Comment réagit openssl ?

Q5. Le fichier *crypto1.chiffre* a été chiffré avec le système AES en mode CBC, la clef de 128 bits ayant été obtenue par mot de passe.
Le mot de passe codé en base 64 est `TWFzdGVyUiZT`. Trouvez la commande openssl appropriée et décodez le mot de passe.

Exemple sur le codage en Base64

Prenons le groupe de 3 caractères ASCII « *Hi!* ». Ci-dessous la première ligne indique en binaire l'équivalence de ces 3 octets. La transformation consiste comme on peut le voir, à séparer les bits pour former en sortie 4 groupes de 6 bits :
01001000 01101001 00100001 ⇔ 010010 000110 100100 100001
Les 4 groupes de 6 bits en sortie nous donnent les valeurs 18, 6, 36 et 33. Ainsi en suivant la correspondance de la table indexée nous obtenons les 4 caractères « *SGkh* ».
Avec la commande base64 sous Linux :

```
$ echo -n 'Hi!' | base64
 SGkh
```

Avec OpenSSL

Encodage d'un fichier texte en base64 :

```
$ openssl enc -base64 -in monfichier.txt -out monfichier.b64
```

Décodage d'un fichier texte en base64 :

```
$ openssl enc -d -base64 -in monfichier.b64 -out monfichier.txt
```

On met le code TWFzdGVyUiZT en base 64 dans le fichier nommé code*base64,* puis on le décode dans un fichier *motPasse*.

```
$ openssl enc -d -base64 -in base64code -out motPasse
$ cat motPass  // résultat ……. est le mot de passe
```

Q 6. Déchiffrez enfin crypto1.chiffre.

```
$ openssl enc -d -aes-128-cbc -in crypto1.chiffre -out crypto1.dechiffre
$ cat crypto1.dechiffre
```

Lab2: Chiffrement avec clef explicite

Il est également possible d'utiliser une clef dite explicite, autrement dit non générée depuis un mot de passe classique. Il faut pour cela fournir une clef et un vecteur d'initialisation tous deux exprimés en hexadécimal.

Q 1. Chiffrez le fichier toto avec une clef explicite. Vous utiliserez le système Blowfish en mode CBC avec un vecteur d'initialisation de 64 bits et une clef de 128 bits.
Pour chiffrer le fichier toto avec une clef explicite, il faut utiliser les options –K et –iv :

1. –K suivi de la clef exprimée en hexadécimal
2. –iv suivi du vecteur d'initialisation exprimé en hexadécimal

L'exemple qui suit montre la commande pour chiffrer toto avec Blowfish en mode CBC avec un vecteur d'initialisation de 64 bits exprimé par 16 chiffres hexadécimaux, et une clef de 128 bits exprimée par 32 chiffres hexadécimaux.

```
$ openssl enc -bf-cbc -in toto -out toto.chiffre -iv 0123456789ABCDEF -K
0123456789ABCDEF0123456789ABCDEF
$openSSL enc -bf-cbc -d -in toto.chiffre  -out toto.dechiffre -iv
0123456789ABCDEF -K 0123456789ABCDEF0123456789ABCDEF
```

Q 2. Chiffrez le fichier crypto2 (à générer par vous-même) avec le système Blowfish en mode OFB, en utilisant le vecteur d'initialisation et la clef de votre choix. Le fichier chiffré se nommera crypto2.chiffre

```
$ openssl enc -bf-ofb -in crypto2 -out crypto2.chiffre -iv 00001111AAAABBBB
-K AAAAAAAABBBBBBBBCCCCCCCCDDDDDDDDEEEEEEEE
```

Q 3. Déchiffrez ensuite le cryptogramme obtenu.

```
$ openssl enc  -bf-ofb -d   -in crypto2.chiffre -out crypto2.dechiffre -iv
00001111AAAABBBB -K AAAAAAAABBBBBBBBCCCCCCCCDDDDDDDD
```

Q 4. Chiffrez le fichier clair crypto1 avec le même système, la même clef et le même vecteur d'initialisation que dans la question qui précède.

```
$ openssl enc -bf-ofb -in crypto1.chiffre -out crypto1.dechiffre -iv
00001111AAAABBBB -K AAAAAAAABBBBBBBBCCCCCCCCDDDDDDDDEEEEEEEE
```