TP n°1

CHIFFREMENT

Objectif

Lors de ce TP, vous devrez récupérer le ficher aes-128-ecb.enc sur la page Moodle du cours.

La boîte à outils **OPENSSL** regroupe des systèmes cryptographiques et offre :

1. une commande **OPENSSL** permettant le chiffrement et le déchiffrement, la création de clefs, le calcul d'empreintes, la création de certificats électroniques, la signature et le chiffrement de courriers ainsi que la réalisation de tests de clients et serveurs SSL/TLS

2. une bibliothèque de programmation en C permettant de réaliser des applications client/serveur sécurisées s'appuyant sur SSL/TLS que nous verrons ultérieurement.

Pour connaître toutes les fonctionnalités de **OPENSSL** :

```
man openssl
```

La commande openssl étant une boîte à outils, il faut spécifier quel outil, ou sous-commande, vous souhaitez utiliser, selon le format suivant :

```
openssl <sous-commande> <options>
```

Pour plus d'informations sur chaque *sous-commande* de openssl : *man sous-commande*
De nombreux systèmes de chiffrement existent et openssl permet de les manipuler facilement. Pour lister les suites cryptographiques utilisables :

```
openssl ciphers -v
```

On peut restreindre la liste à ceux proposant un haut niveau de chiffrement (i.e. clefs de taille supérieure à 128 bits)

```
openssl ciphers -v 'HIGH'
```

Ou encore à ceux proposant un haut niveau de chiffrement pour un même algorithme :

```
openssl ciphers -v 'AES+HIGH'
```

La sous-commande enc qui permet de les utiliser pour chiffrer/déchiffrer avec openssl : *openssl enc <options>*

Merci de déposer sur Moodle, un compte-rendu de l'exercice 3 avant la fin du TP en précisant bien votre nom et prénom.

1. Exercice 1 : Fonction de HASH

1.) Effectuez un hachage du mot ***Bonjour*** en utilisant :
   a. l'algorithme MD5 ;
   b. l'algorithme SHA1.
2.) Effectuez un hachage de la phrase ***Bonjour je suis un hash*** *en utilisant*
   a. l'algorithme MD5 ;
   b. l'algorithme SHA1.
3.) Comparez le résultat des sorties MD5 puis des sorties SHA1, que pouvez-vous en conclure ?
4.) Encodez le mot ***Bonjour*** en Base64
5.) Encodez la phrase ***Bonjour je ne suis pas un hash*** en Base64
6.) Comparez le résultat des deux sorties, que pouvez-vous en conclure ?
7.) Pouvez-vous effectuer les opérations inverses afin de retrouver les phrases des questions 2 et 5 ?
8.) Que pouvez conclure sur le hachage et l'encodage ?

2. Exercice 2 : Chiffrement symétrique avec mot de passe et salage (salt)

1.) Chiffrez la phrase ***Chiffrement en BlowFish*** dans un fichier *result1-bf-cbc* en utilisant bf-cbc.
2.) Chiffrez à nouveau la même phrase avec le même mot de passe dans un fichier *result2-bf-cbc.*
3.) Comparez les fichiers result1-bf-cbc et result2-bf-cbc à l'aide de la commande xxd, que constatez- vous ?
4.) Vérifiez que le déchiffrement des 2 fichiers précédents donne bien le même résultat.
5.) Refaites les questions 1 à 4 en rajoutant l'option -nosalt lors du chiffrement. Que pouvez-vous conclure ?

3. Exercice 3 (Noté): Déchiffrement avec clef et bourrage (padding)

Le fichier *aes-128-ecb.enc* a été chiffré en AES mode ecb, la clef de chiffrement en base64 donnant :

```
Y2VjaWVzdHVuZWNsZWY=
```

1.) Déchiffrez le fichier *aes-128-ecb.enc* et affichez son contenu.
2.) Créez un fichier *secret* dont la taille (en octets) ne soit pas un multiple de 8.
3.) Chiffrez-le sans *grain de sel* avec le système Blowfish en mode CBC. Le fichier chiffré se nommera *secret.enc* .
4.) Déchiffrez ce fichier et enregistrez le résultat dans un fichier *secret.dec.* Vérifiez le contenu.
5.) Comparez la taille des trois fichiers (clair, chiffré et déchiffré). Que constatez-vous ?
6.) Déchiffrez maintenant votre fichier secret.enc en *secret2.dec* en utilisant l'option *-nopad..*
7.) Comparez à nouveau les tailles des trois fichiers obtenus.
8.) Visualisez le fichier *secret2.dec* avec nano ou vi puis avec la commande *xxd*. Qu'en déduisez-vous?

4. Exercice 4 : Bonus

1.) Illustrez une des faiblesses du mode ECB pour du chiffrement symétrique par bloc.
Utilisez l'algorithme de chiffrement BlowFish (bf-ecb), pour générer un résultat chiffré d'un texte contenant 16 caractères « a » avec les contraintes suivantes : aucun « padding » et aucun « salage »
2.) Affichez le résultat avec la commande *xxd*.
3.) Montrez que le mode ECB n'utilise pas de vecteur d'initialisation.
4.) Montrez que le mode CBC corrige la faille précédente en utilisant le même mot de passe et le même texte.
5.) Montrez que le mode CBC utilise un vecteur d'initialisation IV.
6.) Montrez qu'avec le même algorithme et même clef, un texte clair donnera toujours le même texte chiffré.
7.) Trouvez une solution qui permet d'éviter le phénomène précédent.
8.) Quel algorithme est le plus rapide entre des-cbc, aes-512-cbc et bf-cbc ?