Exercice : La cryptographie symétrique et asymétrique

Exercice 1

OpenSSL est un logiciel de cryptographie qui possède de nombreux algorithmes de chiffrement. OpenSSL est disponible sur les systèmes **windows/linux/mac** et s'utilise en ligne de commandes.

Voici les commandes que nous allons utiliser :

Pour encoder :

```
openssl enc -e -encodage -in mon-fichier.txt
openssl enc -e -encodage -in mon-fichier.txt -out mon-fichier-crypte.txt
```

Pour décoder :

```
openssl enc -d -encodage -in mon-fichier.txt
openssl enc -d -encodage -in mon-fichier-crypte.txt -out mon-fichier-decrypte.txt
```

Parmi les nombreux encodages, on s'intéresse aux algorithmes de chiffrement **base64** et **aes-256-ecb** qui nécessite une clef (mot de passe).

1) Créer un fichier texte contenant un message court.

2) Chiffrer votre message avec l'algorithme base64 puis envoyer le à un camarade pour qu'il le déchiffre.

3) **a)** Chiffrer votre message avec l'algorithme aes-256-ecb avec une clef de votre choix.

b) Envoyer votre message à un camarade en lui communiquant la clef.

4) **a)** Créer un nouveau message court.

b) Chiffrer votre message avec l'algorithme aes-256-ecb en utilisant la clef de votre choix.

c) Chiffrer votre clef avec l'algorithme base64.

d) Envoyer le message et la clef à un camarade pour qu'il de déchiffre.

5) Sur le web, dans le code source des pages html, il arrive d'avoir des images encodées avec l'algorithme de chiffrement base64. Le source de l'image de la page html ne contient pas une url mais le code encodé avec l'algorithme base64.

Nous allons faire une petite démonstration.

a) Créer une page html contenant une image au format jpg :

```
<!Doctype html>
<html lang="fr">
<body>
<img src='chemin/vers/mon/image' />
</body>
</html>
```

b) Chiffrer votre image avec **openssl** en utilisant l'algorithme base64.

c) Copier et coller votre code chiffré dans votre image comme ci-dessous :

```
<!Doctype html>
<html lang="fr">
<body>
<img src=' votre-code-chiffré-en-base64' />
</body>
</html>
```

Exercice 2

On rappelle que le OU exclusif se note par un accent circonflexe en python : 4 XOR 5 = 4^5.

1) Calculer en Python 154^45. Expliquer la valeur obtenue.

2) Calculer en Python $A\hat{}z$. Expliquer la valeur obtenue et donner le caractère ASCII associé.

Exercice 3

1) Écrire une fonction qui chiffre avec un OU exclusif. Elle aura 2 paramètres, le mot à chiffrer et la clef de chiffrement.

2) Chiffrer le mot **BONJOUR** avec la clef **12xy**.

3) Déchiffre votre mot chiffré et vérifier qu'on obtient bien le mot BONJOUR.

Exercice 4

On donne la fonction qui permet de chiffrer un message **m** avec une clef de chiffrement **c**. Les arguments **m** et **c** sont en binaire. Cette fonction utilise le OU exclusif (un circonflexe en Python) et renvoie un message chiffré en binaire. En voici le code sans les commentaires :

```
1 def chiffrer_message(m,c):
2     return bytes([m[i] ^ c[i % len(clef)] for i in range(len(m))])
```

Remarque : en Python, une donnée en binaire est accessible comme pour les chaines de caractères qui sont traitées comme les listes. On accède à un octet de la donnée en précisant sa position entre crochets. La valeur renvoyée est exprimée en nombre décimal.

La fonction peut être utilisée pour déchiffre un message grâce à la propriété du OU exclusif :

$$A \oplus B = C \iff A \oplus C = B$$

On donne également 4 autres fonctions : **chiffrer**, **dechiffrer**, **enregistre_crypte** et **enregistre_crypte**.

— **chiffrer** prend en argument la clef de chiffrement et chiffre le contenu du fichier **clair.txt**. La fonction renvoie une valeur binaire.

— **dechiffrer** prend en argument la clef de chiffrement et déchiffre le contenu du fichier **crypte.txt**. La fonction renvoie une valeur binaire.

— **enregistre_crypte** prend en argument un message en binaire et l'enregistre dans le fichier **crypte.txt**.

— **enregistre_decrypte** prend en argument un message en binaire et l'enregistre dans le fichier **decrypte.txt**.

Le programme principal se décompose en 4 temps :

— **Temps 1 :** on chiffre un message.

— **Temps 2 :** on enregistre le message chiffré.

— **Temps 3 :** on déchiffre le message chiffré.

— **Temps 4 :** on enregistre le message déchiffré.

1) Créer un message enregistré dans le fichier **clair.txt**.

2) Compléter le script python du fichier **crypto.py** disponible sur l'ENT.

3) Lancer votre programme, puis vérifier la création des fichiers **crypte.txt** et **decrypte.txt**.

4) Contrôler que le fichier **clair.txt** et **decrypte.txt** sont identiques.