Cryptographie

TD n°3 : Cryptographie asymétrique

Exercice 0

Si vous ne l'avez pas encore fait, implémentez un programme capable de **déchiffrer** les fichiers chiffrés par `chiffre_fichier.py`.

Exercice 1

Dans cet exercice, nous allons donner la possibilité à **plusieurs utilisateurs** de déchiffrer un fichier **sans utiliser de mot de passe**. En effet, les utilisateurs vont pouvoir utiliser des couples clé publique (pour chiffrer) / clé privée (pour déchiffrer) à la place.

1 Génération des bi-clés

Il vous faudra utiliser l'utilitaire `openssl` pour générer des paires de clés. Une fois que vous l'aurez éventuellement installé, utilisez les commandes suivantes pour générer une clé privée RSA de 2048 bits, qui sera stockée dans le fichier `key.pem` :

```
openssl genrsa -out key.pem 2048
```

Ensuite, vous pouvez extraire la clé publique via la commande suivante, et la stocker dans le fichier `key.pub.pem` :

```
openssl rsa -in key.pem -outform PEM -pubout -out key.pub.pem
```

Pensez aussi à générer un deuxième bi-clé (en choisissant des noms de fichier différents pour les nouvelles clés privée et publique).

2 Chargement des bi-clés

En utilisant le paquet Python `cryptography`, chargez la clé publique et la clé privée. Les primitives nécessaires sont présentes dans le sous-module de sérialization. Le format utilisé par OpenSSL s'appelle PEM, et on charge bien des **clés**, et non des certificats X.509. Avec toutes ces informations, vous devriez pouvoir trouver les fonctions à appeler !

3 Chiffrement de fichier avec les clés publiques de destinatires

Au lieu d'utiliser un mot de passe pour chiffrer et déchiffrer un fichier, on va utiliser les bi-clés à la place. L'idée est de mettre en place un chiffrement dit *hybride* :

- Pour chiffrer les données d'un fichier, on va générer une clé symétrique aléatoire, et **chiffrer cette clé** avec les **clés publiques** des destinataires.
- On stocke alors, dans le fichier chiffré, en plus du contenu chiffré et de l'IV éventuel, tous les chiffrés de la clé ainsi obtenus (un par correspondant).

4 Déchiffrer grâce à sa clé privée

Maintenant, chaque destinataire pourra déchiffrer le fichier :

- Il ou elle va essayer de déchiffrer la clé de chiffrement symmétrique. Normalement, en cas de tentative de déchiffrement du mauvais chiffré, le padding sera incorrect, il n'est pas nécessaire d'indiquer au destinataire quel chiffré lui est destiné.
- Une fois le déchiffrement réussi, le destinataire possède la clé de chiffrement symmétrique, et peut retrouver le contenu du ficher transmis.