Travaux Pratiques – TP
Chiffrement Asymétrique

Partie 1 : OpenSSL

OpenSSL est une boîte à outils cryptographiques implémentant les protocoles SSL et TLS. Il offre :

- Une bibliothèque de programmation en C permettant de réaliser des applications client/serveur sécurisées s'appuyant sur SSL/TLS
- Une commande en ligne (OpenSSL) permettant
  - la création de clés RSA, DSA (signature)
  - la création de certificats X509
  - le calcul d'empreintes (MD5, SHA, RIPEMD160, ...)
  - le chiffrement et déchiffrement (RSA, DES, IDEA, RC2, RC4, Blowfish, ...)
  - la réalisation de tests de clients et serveurs SSL/TLS
  - la signature et le chiffrement de courriers (S/MIME)

La syntaxe générale de la commande *openssl* est

```
$ Openssl  <commande>  <option>
```

On trouvera toutes les informations la concernant à l'adresse : http://www.openssl.org

a) Lancer une machine virtuelle *Debian* et taper la commande *su –s (mot de passe : root)*
b) Vérifier l'installation de l'outil en tapant la commande *openssl*
c) Les commandes à utiliser dans ce TP :
   1. genrsa : est une fonction pour générer une paire de clés RSA
   2. *rsautl :* est la fonction de chiffrement et déchiffrement asymétrique.
   3. *dgst :* est une fonction pour générer une empreinte de hachage.
   4. *rsa : permet de visualiser le contenu d'un fichier au format PEM contenant une paire de clés RSA.*

Pour connaitre les paramètres des commandes :

openssl enc -h

openssl rand -h

openssl rsautl –h

openssl rsa –h

Partie 2 : Chiffrement asymétrique

Dans cet exercice, l'intérêt est de chiffrer un document avec une clé publique que seul son créateur pourra déchiffrer.

La commande *openssl rsautl* permet de chiffrer et déchiffrer des messages. Cette commande permet aussi de signer un fichier et vérifier la signature. Plus d'informations sur cette commande peuvent être trouvées en tapant openssl rsautl –h.

La commande *openssl rsa* permet de générer une paire de clés. Plus d'informations sur cette commande peuvent être trouvées en tapant openssl rsa –h.

a) Générer une paire de clés RSA 1024 bits que vous nommerez PrivateKeyMyName.priv. Ex : PrivateKeyDupont.priv. Utilisez l'option –p.

b) Extraire ensuite la clé publique que vous nommerez PublicKeyDupont.pub
c) Envoyer à votre voisin votre clé publique (pas ssh ou clé usb)
d) Créer un fichier texte Plain.txt et y écrire : « Travaux Pratiques ». Chiffrer le fichier Plain.txt avec la clé publique de votre collègue
e) Demander à votre collègue de chiffrer son fichier Plain.txt avec votre clé publique. Déchiffrer le fichier que vous avez reçu avec votre clé privée.

f) Grâce à la commande rand, créer un fichier GrandFichier.txt avec des données aléatoires, d'une taille d'environ 1000 Mo (1 Go). Chiffrer le fichier GrandFichier.txt avec la clé publique de votre collègue. Vous devez remarquer un problème. Comment l'expliquez-vous ?

Partie 3 : Chiffrement symétrique/asymétrique et signature numérique

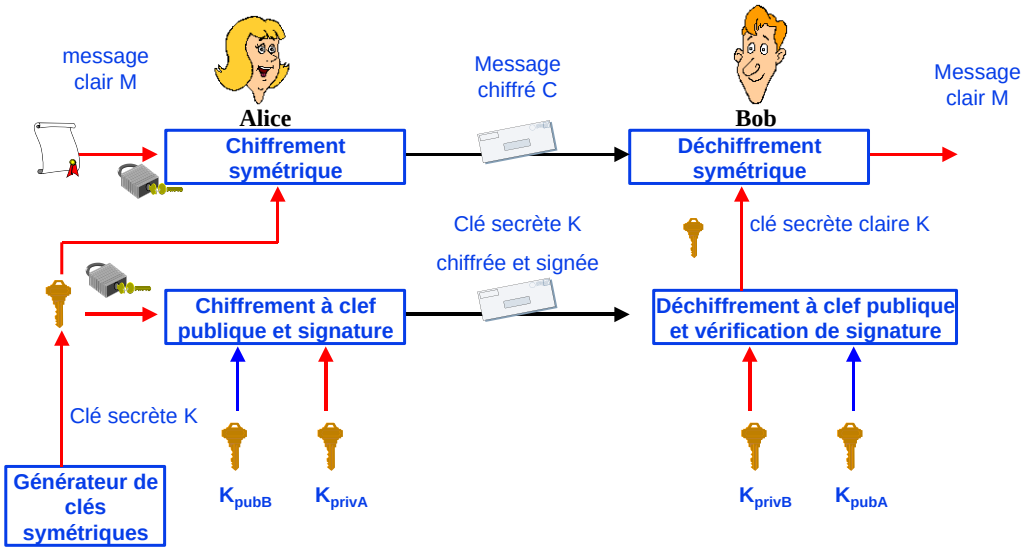

Figure 1. Cryptographie hybride (symétrique et asymétrique)

a) Expliquer le scénario ci-dessus.
b) Quels sont les différents services de sécurité réalisés dans ce scénario.
c) Générer avec la commande rand une clé symétrique SymKey.key de taille 128 bits. Encoder la clé en hex.
d) Visualiser la clé symétrique en binaire en utilisant la commande linux : xxd
xxd –b –c 8 SymKey.key
e) Réaliser le scénario ci-dessus en échangeant d'une manière sûre la clé symétrique avec votre collègue. Pour information, on signe le hash d'un fichier et non pas le fichier lui-même.

Quelques commandes utiles :

1. OpenSSL possède une sous-commande qui permet de tester si le nombre passé en paramètre est un nombre premier. Exemple :
   openssl prime 11
   11 is prime

2. OpenSSL possède une sous-commande pour générer un nombre premier :
   openssl prime -generate -bits 64
   openssl prime -generate -bits 2048 -hex
3. L'option speed permet de tester les capacités du système pour encrypter des données avec les différents algorithmes de cryptage pendant une période donnée.
   a. *openssl speed sha256*
   b. *openssl speed des*