TP-1: OPENSSL

15 septembre 2015

`Secure Socket Layer` est un ensemble de protocoles permettant la communication sécurisée en mode client/serveur pour des applications basées sur TCP/IP. Il existe des implémentations gratuites et privées de SSL. `OpenSSL` est la principale implémentation libre mais depuis *Heartbleed*, il existe `LibreSSL` comme alternative libre. A partir de SSL, l'IETF a réalisé le protocole TLS (`Transport Layer Security`) qui sert de base à HTTPS. Nous allons travailler avec OpenSSL. Cet implémentation de SSL a été réalisée en C.

1 Philosophie

Utiliser SSL est relativement simple.

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

Voici quelques commandes principales :
- `enc/dec` pour chiffrer/déchiffrer des fichiers.
- `rand/genrsa` pour générer des valeurs aléatoires ou des clefs RSA.
- `dgst` pour manipuler des fonctions de hachage

Il en existe beaucoup d'autres. En cas de doutes sur OpenSSL, vous pouvez/devez consulter le `man`.

2 Travail à effectuer

Trouver les commandes permettant de réaliser les actions suivantes.

2.1 Chiffrement symétrique

1. Forger une clef AES 128 et 256 bits.
2. Chiffrer un fichier avec AES-128-CTR et AES-128-CBC.
   Que constatez vous sur la taille des fichiers chiffrés (`wc -c`) ?
3. Comparer les performances (oui il y a aussi une commande pour ça aussi) de RC4, AES et DES.
4. Envoyer un mail a un autre groupe afin de vérifier que ce que vous avez chiffré peut être déchiffré.

2.2 Hachage

1. Récupérer les fichiers M1.ps.gz et M2.ps.gz sur l'`ensiwiki`. Après les avoir décompresser ouvrer ces deux fichiers et constater leur contenu.
2. Utiliser la fonction de hachage MD5 pour vérifier l'intégrité de ces deux documents. Que constatez vous ?
3. Même question mais avec SHA1 et SHA256.
4. Que faut-il en conclure ?

2.3 Chiffrement asymétrique

1. Les opérations de chiffrement/déchiffrement en cryptographie asymétrique nécessitent des clef assez complexes. On ne peut pas se contenter de tirer des valeurs aléatoires (même si on en a besoin).

```
openssl genrsa -out mykey.pem 1024
```

2. Comment faut il interpreter le resultat de la commande :

```
openssl rsa -in mykey.pem -text -noout
```

3. Extraire la clef publique de ce fichier afin de pouvoir la distribuer à un autre groupe.
4. Extraire la clef privée de ce fichier afin de pouvoir la stocker dans un fichier qui sera chiffré avec AES-128.
5. Chiffrer un message avec la clef publique d'un autre groupe. Envoyer le chiffre obtenu par mail afin de vérifier que tout fonctionne correctement.
6. Mesurer les performances de RSA 1024 et 2048 bits. Comparer vos résultats à ceux de l'AES.