OPENSSL

16 septembre 2016

1 Génération de nombres aléatoires

Pour générer des valeurs aléatoires on dispose de la commande `rand` dans `openssl` :

```
$ openssl rand -out dest num
```

On a les options suivantes :
— `num` est le nombre d'octets aléatoires produits.
— `-out` dest est le nom du fichier de destination.
On dispose aussi d'options pour formater la sortie en hexadécimal ou en base64.

Questions

— Utiliser `openssl` pour produire une clef et un vecteur d'initialisation de longueur respective 128 bits sauvegarder dans `key.rnd` et `iv.rnd`.
— En utilisant, la documentation de `openssl` expliquer le processus qui a été mis en œuvre pour générer ces nombres. Est-ce satisfaisant ?
— Sous Linux, on dispose aussi `/dev/random` et `/dev/random`

```
$ dd if=/dev/urandom of=data.rnd bs=16 count=1
```

Quelle est la différence entre `/dev/random` et `/dev/urandom` ?

2 Chiffrement symétrique

2.1 Avec clef et IV

On peut chiffrer ou déchiffrer des fichiers avec `openssl` en utilisant les commandes `-enc -c` et `enc -d` :

```
$ openssl enc -e -ciphername -in clear -out cipher -K mykey -iv my iv
```

avec les options suivantes :
— `-ciphername`,
— `-in` indique le nom du fichier à chiffrer,
— `-out` indique le nom du fichier chiffré,
— `-K` la clef en héxadecimal,
— `-iv` le vecteur d'initialisation en héxadecimal.

Questions

— Chiffrer le fichier `test1` avec l'AES en mode CTR (128 bits) avec la clef et le vecteur d'initialisation des questions précédentes.
— Même chose avec l'AES en mode CBC (128 bits). Comparer la taille du fichier `test1` avec les deux fichiers chiffrés obtenus (`ls` ou `wc`). Est ce que cela correspond à vos attentes ?

— Comparer la performance (vitesse) des différents modes de chiffrement que nous avons vu en cours pour l'AES (128 bits) en utilisant l'option :

```
$ openssl speed ciphername
```

Est ce que cela correspond à vos attentes en terme de performance ?
— Le fichier `test2.enc` a été chiffré. En analysant le fichier avec `hexdump` quel mode opératoire a-t-il été utilisé ?
— Les fichiers `test3.enc` et `test4.enc` ont été chiffrée avec l'AES et la même clef. Quelle erreur a été commise lors du chiffrement ?

2.2 Avec mot de passe

Il est relativement difficile pour un être humain de retenir la clef et le vecteur d'initialisation utilisée pour chiffrer un fichier. `openssl` utilise un système de mot de passe ce qui est compatible avec la mémoire de l'utilisateur :

```
$ openssl enc -e -ciphername -in clear -out cipher
```

`openssl` va demander choisir un mot de passe et de le confirmer.

Questions

— Chiffrer le fichier `test1` avec l'AES en mode CTR (128 bits) en utilisant un mot de passe. Vous obtiendrez un fichier `res1.enc`
— Chiffrer une seconde choix le fichier `test1` avec l'AES en mode CTR (128 bits) en utilisant le même mot de passe que dans la question précédente. Vous obtiendrez un fichier `res2.enc`
— Comparez `res1.enc` et `res2.enc`. Le résultat vous surprend-t-il ?
— Chiffrer une troisième fois `test1` toujours avec le même mode opératoire de l'AES et le même mot de passe mais vous allez utiliser l'option `-p`. Expliquer comment la clef et le vecteur d'initialisation sont générées.
— Utiliser `hexdump` pour comprendre la structure d'un fichier chiffré avec un mot de passe. Quelle option doit-on donnée à `openssl` pour obtenir après chiffrement un fichier qui contient uniquement le texte chiffré ?

3 Hachage

On peut calculer l'empreinte d'un fichier avec openssl en utilisant les commandes `dgst` :

```
$ openssl dgst -hashname filename
```

avec les options suivantes :
— `-hashname` le nom de la fonction de hachage
— `filename` le nom du fichier

Questions

— Calculer l'empreinte du fichier `test1` avec SHA-256 et SHA-512.
— Calculer les empreintes de `l1.ps` et `l2.ps` avec MD5. Visualiser ces deux documents. Qu'en pensez vous ?
— Pour calculer l'empreinte d'une chaîne de caractères, on peut utiliser la commande :

```
$ echo "mystring" | openssl dgst -hashname
```

L'empreinte de "toto" est : `0x0b9c2625dc21ef05f6ad4ddf47c5f203837aa32c` Êtes vous d'accord ? (A quoi faut-il faire attention ?)

— `hashcash` est un système de preuve de travail utilise dans Bitcoint. Pour générer de nouveau bitcoin, il faut réussir à trouver des pré-images pour des empreintes SHA-256 de la forme :

`0x000000XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX`

4 Chiffrement asymétrique

Nous allons regarder comment utiliser RSA avec `openssl` La génération des clefs est faites de la façon suivante :

```
$ openssl genrsa -out filename size
```

avec comme paramètres :

— `filename` le fichier dans lequel les clefs vont être écrites.
— `size` la taille en bits de la clef.

Pour inspecter et visualiser les clefs, on dispose de la commande suivante :

```
$ openssl rsa -in filename -text -noout
```

Le fichier `filename` contient la clef publique et de la clef privée : il faut y faire très attention et ne jamais divulguer la clef privée ! Par contre on peut partager avec tout le monde la clef publique. Pour extraire la clef publique, on dispose de la commande suivante :

```
$ openssl rsa -in filename -pubout -out filename.pub
```

Le fichier `filename.pub` peut être partagé avec n'importe qui ! Pour chiffrer un fichier, on dispose de la commande suivante

```
$ openssl rsautl -encrypt -pubin -in clear -inkey filename.pub -out cipher
```

avec comme paramètres :

— `clear` le fichier contenant le texte clair,
— `cipher` le fichier contenant le texte chiffré.

Questions

— Tester le chiffrement avec RSA 2048 sur un fichier de votre choix en échangeant par mail avec un autre groupe vos clefs publiques.
— Même question pour le fichier contenant 3KB de données aléatoires. Que pensez vous du chiffrement avec RSA ? Comment faire pour être plus efficace ?