UNIVERSITÉ LAVAL

Travail pratique N^o 2

Cours : GLO-3100 Cryptographie et sécurité informatique

Objectif

Le but de ce travail est de manipuler des librairies cryptographiques fournies avec des langages de programmation ainsi que de mieux maîtriser le principe de fonctionnement de la cryptographie par seuil. Dans ce TP, on ne vous demande pas d'implanter des systèmes cryptographiques comme AES-CBC-128 ou AES-CTR-128, mais plutôt d'utiliser la librairie cryptographique offerte par votre langage de programmation. Un petit exemple en C++ et un autre en Java montrant comment manipuler ces librairies sont disponibles sur le site web du cours.

1 Exercice 1 : Piratage informatique : *Ransomware* (4.5pts)

On vous demande d'écrire un programme nommé *ex1* qui prend les paramètres suivants :
- un répertoire (spécifié via l'option *-d* ; par défaut, c'est le répertoire courant) ;
- une liste de types de fichiers dont chacun est spécifié avec l'option *-f*. Pour cet exercice, nous nous limitons aux types *xls*, *doc*, *pdf*, *png*, *mp3*, *avi* et *txt*) ; et
- une opération (spécifiée via l'option `-op`) indiquant deux valeurs possibles : `enc` (pour un chiffrement) et `dec` (pour un déchiffrement).

Quand la valeur indiquée par `op` est `enc`, le programme chiffre tous les fichiers ayant les types indiqués par l'option `-f` du répertoire spécifié par `-d` avec le système cryptographique AES-CBC-128. La même opération se fait aussi récursivement pour tous les sous-répertoires du répertoire indiqué.

Une fois le travail terminé, le message suivant est affiché à l'utilisateur : "cet ordinateur est piraté, plusieurs fichiers ont été chiffrés, une rançon de 1000$ doit être payée sur le compte PayPal hacker@gmail.com pour pouvoir récupérer vos données".

Par ailleurs, toutes les informations nécessaires au chiffrement (clé, iv, etc.) sont générées aléatoirement par le programme. Pour la plupart des *ransomware*, toutes ces informations sont envoyées au pirate. Mais, pour simplifier l'exercice, on se contente de les enregistrer dans un fichier nommé *pirate.txt* et de le mettre dans le répertoire qu'on vient de chiffrer son contenu (fourni avec l'option -d). Par ailleurs, les types de fichiers chiffrés sont également stockés dans le fichier *pirate.txt*. En quelque sortes tous les paramètres nécessaires à remettre le répertoire dans sont état initial seront enregistré dans le fichier *pirate.txt*

Exemples d'appels de votre programme.

```
ex1 -f doc -f pdf -f png -op enc
ex1 -d c: -f doc -f pdf -f png -op enc
```

Une fois un répertoire est chiffré, il est possible de le déchiffrer en faisant appel à *ex1* en donnant son nom (avec l'option *-d*), avec l'option *-op dec*. Si le répertoire ne contient pas le fichier *pirate.txt*, un message d'erreur sera affiché.

Exemple :

```
ex1 -d c: -op dec
```

2 Exercice 2 : Gestionnaire de mots de passe (4.5pts)

On vous demande d'écrire un programme qui vous permet de gérer vos mots de passe liés à plusieurs comptes (Gmail, Facebook, etc.) d'une manière sécuritaire. Votre trousse de mots de passe est protégée par un secret unique (un autre mot de passe qui ne devrait pas être enregistré sur votre ordinateur.). À chaque fois que vous voulez accéder à un de vos comptes, vous déverrouillez les informations requises dans le gestionnaire en fournissant le secret.

Voici plus de détails techniques sur les fonctionnalités de ce gestionnaire.
- Nous voulons avoir un ficher sur notre disque dur contenant des enregistrements ayant les informations suivantes :

  - *URL* : l'adresse URL du service.
  - *user* : le nom d'utilisateur lié au compte.
  - *pwd* : le mot de passe lié au compte associé à l'URL.
- On vous recommande d'utiliser le format *Json* et d'écrire les enregistrements en question dans un fichier texte, mais vous pouvez faire d'autres choix.
- Une fois sur le disque dur, les informations liées à ces champs devraient être chiffrées séparément avec AES-CTR-128 bits. La clé de protection est générée à partir d'un secret (un mot de passe) donné par l'utilisateur. Il s'agit du seul secret que l'utilisateur a besoin de s'en souvenir. Chaque champ aura son propre iv (nécessaire pour le mode CTR) généré aléatoirement et enregistré à côté du champ en question pour faciliter son déchiffrement plus tard.

Pour simplifier le programme, nous supposons qu'il s'appelle ex2 et que nous pouvons le manipuler en mode ligne de commandes avec les options suivantes :

- l'option -a permet d'ajouter un nouvel enregistrement à la trousse de clé. Elle doit être suivie par le mot de passe de la trousse. Dans ce cas, l'utilisateur doit spécifier les champs de l'enregistrement avec les options suivantes :
  `-url` : donne l'URL.
  `-user` : donne le nom d'utilisateur.
  `-pwd` : donne le mot de passe permettant d'accéder au service.

  Exemple :
  ```
  ex2 -a 123pwd -url www.facebook.com -user alice@gmail.com -pwd 12t3r
  ```
- l'option -l suivie du mot de passe de la trousse permet des lister les enregistrements. L'affichage commence par une ligne indiquant le contenu de chaque colonne. Chaque enregistrement est précédé par un numéro de ligne. Tout doit apparaitre en claire sauf les noms d'utilisateurs et leurs mots de passe qui restent chiffrés et nous affichons "*****" à leurs places.

  Exemple de commande :
  ```
  ex2 -l 123pwd
  ```

  Exemple de résultats.

  ```
  ligne             url                          user             pwd
  1                  www.facebook.co             *****             *****
  2                  www.gmail.com               *****             *****
  ```

- l'option `-d` suivie du mot de passe de la trousse permet d'afficher une ligne particulière tout en déchiffrant certains champs de plus. Le numéro de la ligne est spécifié avec l'option `-i`. Si nous voulons déchiffrer le nom, nous le spécifions avec le champ `-user` et si nous voulons déchiffrer le mot de passe, nous le spécifions avec l'option `-pwd`. Autrement, ces informations restent chiffrées.

  L'exemple suivant permet d'afficher le contenu de toute la ligne numéro 1 en clair :
  ```
  ex2 -d 123pwd -i 1 -user -pwd
  ```

  Exemple de résultats.

  ```
  ligne            url                user             pwd
    1              www.facebook.com   alice@gmail.com  12t3r
  ```

  Un autre exemple `ex2 -d 123pwd -i 2 -pwd`

  ```
  ligne       url                 user             pwd
  1           www.facebook.com     *****           5ewr1
  ```

  À noter que votre programme ne devrait enregistrer nulle part le mot de passe qui protège la trousse de clés. Il n'a pas à savoir aussi si l'utilisateur a fourni le bon mot de passe ou non. Tout ce qu'il fait, c'est de déchiffrer l'information demandée avec le mot de passe fourni. Si l'utilisateur fournit un mauvais mot de passe, le message affiché n'aurait pas de sens.

3 Exercice 3 : Cryptographie par seuil (3pts)

Écrire un programme ex3 permettant de partager un secret et de le retrouver selon l'algorithme de Shamir. Les entrées du programme sont comme suit :

— l'option -e suivie d'un seuil (k, n), deux entiers naturels k et n, avec $k \leq n$, (spécifié avec les options *-k* et *-n* respectivement), un nombre premier q (spécifié avec l'option *-q*) et un secret s (spécifié avec l'option *-s*). Le programme retourne, dans ce cas, n points secrets $s_1, \ldots, s_n$ ou $s_i = (i, f(i)) \; mod \; q$ qui cache s selon l'algorithme par seuil de Shamir.

Exemple :
`ex3 -e -k 3 -n 5 -s 7 -q 31`
Pour cet exemple, l'algorithme génère aléatoirement les coefficients a_1 et a_2 d'un polynôme de degré 2 $(k-1)$. Le coefficient a_0 prend la valeur du secret. Le polynôme serait par exemple $f(x) = 7 + 19x + 21x^2$ lorsque $a_1 = 19$ et $a_2 = 21$). À partir de ce polynôme, le programme extrait 5 points qui cache le secret 7 (exemple (1,f(1))= (1,16), (2,f(2))=(2,5), (3,f(3))=(3,5), (4,f(4))=(4,16) et (5,f(5))=(5,7) ; le calcul se fait modulo $q = 31$). Plus tard, il nous faut au moins trois points pour pouvoir retrouver le secret 7.

— l'option `-d` suivie de k points (générés selon l'algorithme de Shamir et spécifié avec l'option *-p*) retourne le secret. Par ailleurs, le q doit être toujours spécifié avec l'option $-q$. Un point `(x,y)` sera spécifié par `-p x y`.

Exemple :
`ex3 -d -p 3 5 -p 1 16 -p 2 5 -q 31`

Pour simplifier votre code, vous pouvez utiliser la méthode brute pour inverser une valeur a dans $\mathbb{Z}_n$ comme suit :

```
int modInverse(int a, int n)
{
    a = a%n;
    for (int x=1; x<n; x++)
         if ((a*x) % n == 1)
               return x;
}
```

4 Remarques

1. Le travail est individuel.
2. Seuls les langages C, C++ et Java sont permis.
3. Le barème est à titre indicatif et 10% des points seront attribués aux commentaires.
4. Attention au plagiat ! Faites vos TPs par vous-même.

5 À remettre

Utiliser le site web du cours pour remettre un seul fichier ".zip" (de taille maximale 40 Mb) qui porte votre nom au complet et qui contient un répertoire par exercice (ne m'envoyez pas vos TPs par courriels s.v.p.). Le répertoire en question doit contenir l'exécutable (.jar, etc.) aussi bien que le code source **bien commenté**. Assurez-vous que votre exécutable n'aura besoin d'aucun autre fichier externe pour pouvoir fonctionner sur Kali.

6 Échéancier

Le 18 novembre 2020 avant 14h00. Une pénalité de 0,0347% de la note sera appliquée à chaque minute de retard (l'équivalent de 2,08% point par heure), et ce, pour un maximum de 48 heures. Après 48 heures de retard, la note sera zéro. Par exemple, pour un étudiant qui a eu 10 points, mais avec 6 heures de retard, sa note sera 10 - 10*0,0208*6=8,75.