UNIVERSITÉ LAVAL

Travail pratique N^o 1

Cours : GLO-3100 Cryptographie et sécurité informatique

Objectif

Le but de ce travail est de mieux maîtriser les systèmes de chiffrement symétriques et leurs modes de fonctionnement.

1 Exercice 1 : Briser Vigenère (20pts)

Le chiffrement d'un texte m écrit en anglais en utilisant Vigenère a donné le résultat suivant :

YHKMV MSXVV XTMGV WMRCJ KSQRE JYCBE JYIVOD EFDSC TIRTPE SMPVP ITSYEG BYTNZ SXWLC TLSSXB VTBOG YJWYF GIEZOR XCEKY RHCEE BYMHW SBMEN ZSXCEF CINPPU ZRDEB NZRDO YHZRD EPFZKS BPYKIH TGUCP ODGCG LORXY OXNEG LPTDIS HZWD HILVZO RWYZR YTLYIA YRHMD SFIRAW VYMXB VYXIR NVPVIK CSPOC MJYIBT IRKFKC ONFTV AMHKI HTEWZ TREVC JEZAM LFJKLKI IMDHQ MKLKT GLVEDE XBVIXC VSGXS ORUEH DHILVZ ORWCE KNEGLP TDISHK LODINR MVEHIG IBAXCF RYFEW ZTREVC JGYNXL FPVEHV FXRBCN YIKLKIIM DHQUE HSNIUT LSNWN RRMEFS ROOYXB ZWSSE MVGBEX CUIKLPS BRYWRI EPITSNY IMOQGL RSCEHK WESYU CPIAWB FVDSXL ZRQOJ WYEBA GNVVC WLCTLS SRYVHO DXIUIM RCJKX REGCG LORXY OXKCV SGXYS CMKIWI WNYIY RHYIIN LMMKS PEPYDI XTWIW JSNMN VTYSW CSPOP PUZRD EBNJJS NMNVT YSWCS POCCJ YIBTIRK WPIRCK IZOWM ZFVEOY PWKNH NYIONG LPTDIS HRRNDI WICZT MIEEVG SLZXRM WQYMM HGIIVOS TIEHDO IUTLUE CEVCCA VYZQZO VNRRDA WWZTR EVMNM DHSOKZ KRMUSP OKISJGK NFYKVS VMUCPI BVIBIXW MNYSXL CNYIUN SQCINGI IWXREG CGLORY MVHKN HUIIDHIL VJYRIOJI VEWMFV OVIHTSE NXYITBO HOTXSV IZFVWO WNGYBP SMVWRI WNFVSC EFCCMIT BVVCWI LVSPTIH LWODHC IIMTPSW SBERWI CZTMIE SBDIWIC ZTMIEA STLILXK DHCKM YNEFG VYCIXL VOSWO TLKSEO KLONX CTEDISH FVSNXY XVSTCW YIMKW

Pour les questions suivantes, on vous demande de **bien justifier vos réponses en donnant tout le calcul intermédiaire qui vous amène à la conclusion**. À noter que des réponses sans justifications ne vous donnent pas des points.

1. **(8pts)** Utiliser le test de Friedman pour estimer la taille de la clé sachant qu'elle est inférieure ou égale à 8.
2. **(17pts)** Utiliser l'indice de coïncidence mutuel pour trouver la clé et décrypter le message. L'indice de coïncidence mutuel devrait être utilisé pour trouver le décalage des colonnes par rapport à la colonne 0. Par la suite, en se basant sur la lettre les plus fréquente dans la colonne 0 et celle dans la langue d'origine, on calcule la valeur K_0. À partir de K_0 et des décalages, on calcule les autres valeur de la clé et on décrypte le message.

Remarque : Le programme Vigenere (voir le fichier Vigenere.zip sur le site web du cours) vous aide à faire vos calculs.

2 Exercice 2 : Modes de chiffrement (25pts)

Dans cet exercice, on vous demande d'implanter les modes de chiffrement ECB, CBC, CFB, OFB et CTR avec le système cryptographique symétrique suivant :

$$E_{(k_0,k_1,k_2,k_3,k_4)}(b_0.b_1.b_2.b_3.b_4) = (b_0 \oplus k_0).(b_1 \oplus k_1).(b_2 \oplus k_2).(b_3 \oplus k_3).(b_4 \oplus k_4)$$

Votre programme doit s'appeler *ex2* et offrir les options suivantes (dans n'importe quel ordre) :
-msg suivie d'un message m qui est une suite binaire ayant une taille multiple de 5,
-key suivie d'une suite binaire de 5 bits.
-op suivie de *enc* ou *dec* pour dire si l'on veut chiffrer ou déchiffrer le message.
-mode suivie d'un mode de chiffrement qui est un texte dans ECB, CBC, CFB, OFB, CTR.
-iv (optionnel) suivie d'une suite binaire aléatoire de taille 5. Si le *iv* n'a pas été donné par l'utilisateur, alors le programme le choisi aléatoirement.
-r (seulement pour les modes CFB et OFB) suivie d'un nombre r avec $1 \leq r \leq 5$.

Ensuite, il retourne le résultat de l'opération (chiffrement, déchiffrement), du message m par E_k selon le mode choisi et en utilisant iv comme vecteur d'initialisation.

Exemples d'appels de votre système :

```
ex2 -msg 100110111111010 -key 10101 -op enc -mode CBC -iv 0011
ex2 -msg 100110111111010 -key 11001 -op dec -mode CBC
ex2 -msg 100110111111010 -key 11100 -op enc -mode CFB -iv 10111 -r 3
ex2 -key 10001 -mode CBC -msg 100110111111010
```

3 Exercice 3 : OpenSSL (15pts)

Pour cet exercice, nous vous recommandons d'utiliser la machine virtuelle Kali2.0 64 bits et de la configurer comme indiqué dans cette vidéo). Elle contient déjà OpenSSL et d'autres outils intéressants. Le site site officiel de OpenSSL est : www.openssl.org/. Plus de documentation sur OpenSSL est disponible ici.

Voici quelques commandes de OpenSSL :

- Générer une clé privée RSA de "*size*" bits (512, 1024, etc.)
  $openssl genrsa -out <fichierRsa.priv> <size>
- Création d'un clé publique associée à la clé privé "*fichierrRsa.priv*"
  $openssl rsa -in <fichierrRsa.priv> -pubout -out <fichierRsa.pub>
- Chiffrer une clé privée avec l'algorithme DES3 ou autre.
  $openssl rsa -in <fichierRsa.priv> -des3 -out <fichierOut.>
- Chiffrer le "*fichier.txt*" avec l'algorithme "*algo*" en utilisant la clé qui se trouve dans la première ligne du fichier "*key*".
  $openssl enc <-algo> -in <fichier.txt> -out <fichier.enc> -kfile <key>
- Déchiffrer le "*fichier.enc*" avec l'algorithme "*algo*".
  $openssl enc <-algo> -in <fichier.enc> -d -out <fichier.txt> -kfile <key>
- Hacher un fichier avec "*algo*" (sha1, md5, rmd160, etc.)
  $openssl dgst <-algo> <entree> -out <sortie>
- Générer un nombre aléatoire sur "*nbits*" et mettre le résultat dans "*file.key*" (utiliser l'option "*base64*" pour la lisibilité)
  $openssl rand -out <file.key> <nbits>

Questions : Écrire votre nom et votre prénom dans un fichier nommé plaintext.txt et utiliser OpenSSL pour faire les opérations suivantes :

1. **(2 pts)** Chiffrer plaintext.txt avec AES256 et le mode CBC en utilisant un mot de passe de votre choix et mettre le résultat en format base64 dans le fichier ciphertext.enc. Prendre une capture d'écran montrant, le contenu de fichier plaintext.txt, la commande que vous avez tapée pour le chiffrer ainsi que le contenu du fichier ciphertext.enc
2. **(1 pts)** Comparer la taille des deux fichiers (pliantext.txt et ciphertext.enc) et expliquer la différence.
3. **(2 pts)** Utiliser triple DES_3EDE (Encryption-Decryption-Encryption avec trois clés différentes) avec le mode *ofb* pour chiffrer, puis déchiffrer le fichier plaintext.txt en utilisant un mot de passe de votre choix À noter que OpenSSL utilise une fonction KGF (Key Generation Function) pour générer des clés à partir d'un mot de passe et d'un *salt* (une valeur générée aléatoirement). Prendre une capture d'écran montrant les commandes utilisées et les résultats obtenus. Vos copies d'écran doivent montrer aussi le contenu du fichier en clair, son correspondant chiffré et le résultat de son déchiffrement.
4. **(2 pts)** Mettre le fichier plaintext.txt dans un répertoire portant votre nom, le compresser avec la commande "tar" et le chiffrer avec DESX en mode *cbc* et en utilisant un mot de passe de votre choix. Demander aussi à OpenSSL de vous afficher la clé, le *salt* et le *IV* qu'il a utilisés. Prendre une capture d'écran montrant les commandes utilisées et les résultats obtenus.
5. **(2 pts)** Pour chiffrer un fichier avec une clé explicite, il faut utiliser les options *-K* (clé en hexadécimal) et *-iv* (vecteur d'initialisation en hexadécimal). Générer une clé de 128 bits et un *iv* de 128 bits et utiliser les pour chiffrer, puis déchiffrer le fichier plaintext.txt avec camellia128 en utilisant le mode *cbc*. Prendre une capture d'écran montrant les commandes utilisées et les résultats obtenus. Vos copies d'écran doivent montrer aussi le contenu du fichier en clair, son correspondant chiffré et le résultat de son déchiffrement.
6. **(3 pts)** Chiffrer le contenu du fichier plaintext.txt en mode binaire avec blowfich (bf) en mode *cbc* et le déchiffrer avec l'option *-nopad* (permet de préserver le bourrage lors de déchiffrement) puis visualiser le contenu du fichier chiffré avec la commande *xxd* (visualiser le contenu en hexadécimal). Répétez l'expérience 3 fois, mais à chaque fois vous enlevez un caractère du contenu de plaintext.txt et observez l'effet sur le bourrage introduit par OpenSSl. Donner vos constatations accompagnées d'une ou plusieurs captures d'écran montrant les commandes utilisées et les résultats obtenus.
7. **(3 pts)** Avec la commande `openssl speed`, comparer la rapidité de AES par rapport à DES et DES avec RSA. Prendre des captures d'écran montrant les commandes et les résultats obtenus tout en commentant les résultats.

4 Exercice 4 : Piratage informatique : *Ransomware* (20pts)

On vous demande d'écrire un programme nommé *ex4* qui permet de prendre comme paramètres un répertoire (spécifié par l'option *-r* ; par défaut c'est le répertoire courant), une liste de types de fichiers (chacun type est spécifié avec l'option *-t* et, pour cet exercice, nous nous limitons aux types *ipg*, *doc*, *pdf* et *txt*), une adresse courriel (spécifiée par l'option -c) et de chiffrer tous les fichiers ayant ces types dans ce répertoire et ses sous-répertoires avec le système AES-CBC-128. La clé de chiffrement peut être spécifié en hexadécimal (via l'option -k), sinon elle est générée aléatoirement et elle est de 128bits. Une fois le chiffrement terminé, le programme affiche un message disant que "cet ordinateur est piraté, plusieurs fichiers ont été chiffrés, une rançon de 20$ doit être payée sur le compte PayPal hacker@gmail.com".

```
ex4 –t doc –t pdf –t jpg –k E2 32 FC F1 91 12 91 88 B1 59 E4 E6 D6 79 A2 93
ex4 –r c: –t doc –t pdf –t jpg
```

Une fois un répertoire est chiffré, il est possible de le déchiffrer en faisant appel à *ex4* en donnant son nom (avec l'option *-r*), la clé (avec l'option -k) et en spécifiant que nous voulons le déchiffrement (avec l'option *-d*).

```
ex4 –r c: –d –k E2 32 FC F1 91 12 91 88 B1 59 E4 E6 D6 79 A2 93
```

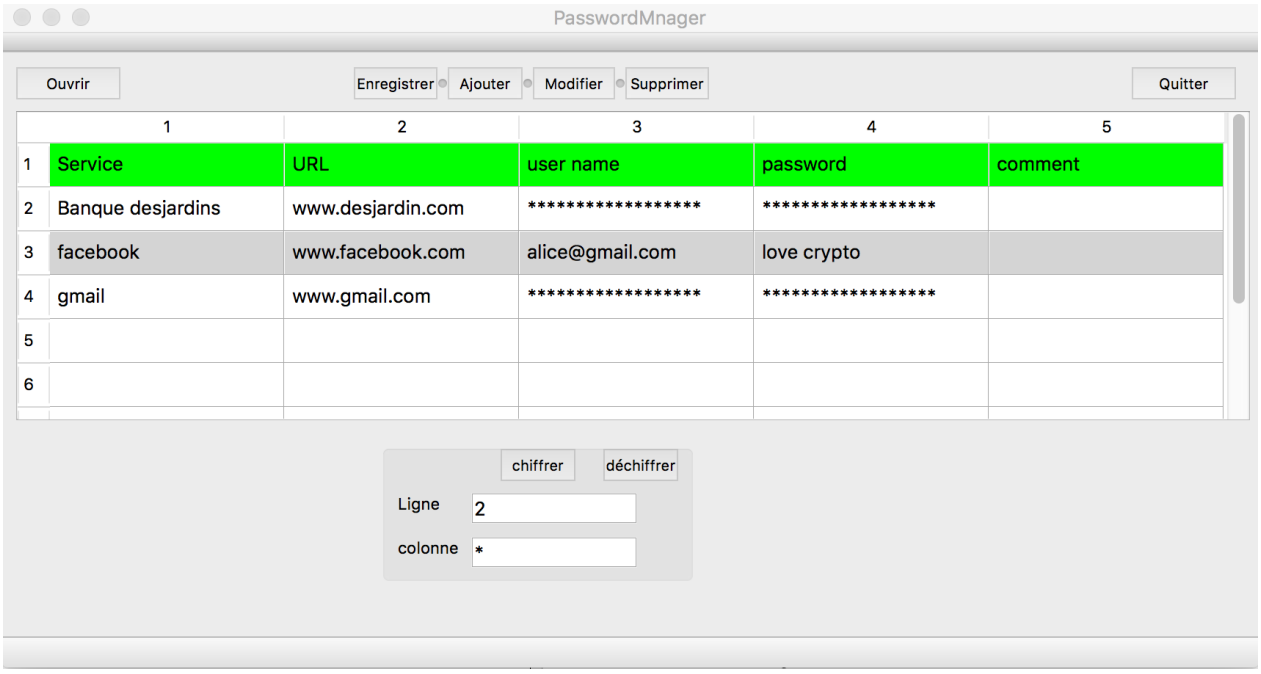
5 Exercice 5 : Gestionnaire de mots de passe(20pts)

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. À chaque fois que vous voulez accéder à un de vos comptes, vous déverrouillez les informations requises dans le gestionnaire.

Voici plus de détails techniques sur les fonctionnalités de ce gestionnaire.

- Nous voulons avoir sur notre disque dur des enregistrements ayant les informations suivantes :
  - *Service* : le nom du service (Gmail, Facebbok, etc.) lié au compte.
  - *URL* : l'adresse URL du service.
  - *User name* : les noms d'utilisateurs liés aux comptes.
  - *Password* : les mots de passe liés à ces comptes.
  - *Comment* : un champ pour ajouter d'autres commentaires.
- 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 avec AES-CTR 128 bits. La clé de protection est générée à partir d'un mot de passe donné par l'utilisateur. Il s'agit du seul mot de passe que l'utilisateur a besoin de s'en souvenir.

L'interface de l'application doit ressembler à celle de la figure suivante.



Elle doit offrir les possibilités suivantes :

– Une fois, l'application est lancée, les informations s'affichent chiffrées sur l'écran. L'utilisateur a la possibilité de déchiffrer la totalité ou partiellement, en spécifiant les lignes et les colonnes et en donnant son mot de passe. Il peut à tout moment aussi chiffrer ces informations de nouveau, et ce, sans donner son mot de passe.
– L'utilisateur doit pouvoir ajouter, modifier et supprimer des enregistrements en fournissant son mot de passe. S'il appuie sur le bouton ajouter, une autre ligne sera ajoutée avec un nombre aléatoire inséré dans la case *Password*. L'utilisateur peut modifier le contenu de cette ligne incluant le mot de passe.
– Vous aurez 5 points de plus (bonis) si vous ajoutez l'option permettant d'ouvrir et de remplir automatiquement la page d'authentification d'un service en cliquant son lien à partir de gestionnaire de mots de passe.

6 Remarques

1. Le travail est individuel.
2. Les langages C, C++, C#, Java, JavaScript sont permis.
3. Le barème est à titre indicatif et que 10% des points associés aux questions d'implantation seront réservés aux commentaires.
4. Attention au plagiat ! Faites vos TPs par vous-même.

7 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.). Quand il s'agit d'un exercice de programmation, le répertoire en question doit contenir l'exécutable aussi bien que le code source **bien commenté**. Assurez-vous aussi que votre exécutable n'aura besoin d'aucun autre fichier externe pour pouvoir fonctionner. Retourner un fichier ".pdf" ou ".doc" pour l'exercice 1 et un fichier ".pdf" ou ".doc" pour l'exercice 5. Les réponses doivent garder les mêmes numéros que les questions et les captures d'écran doivent être bien lisibles.

8 Échéancier

Le 7 novembre 2016 avant 14h00. À noter que les TPs remis en retard ne seront pas acceptés.