INFO-F101 – Programmation
Projet 2
"Save private data" or "jHeaven strikes back"

Année académique 2014–2015

La société *Pear* et l'agence ANS (Agency of National Security) ont créé un nouvel algorithme de chiffrement de données. Le nouveau algorithme devrait mieux protéger la vie privée et les photos des célébrités dans le service de stockage sur les serveur de jHeaven.

L'idée principale d'un algorithme de chiffrement est simple : l'algorithme reçoit en entrée un message et une clé, ensuite l'algorithme transforme le message à l'aide de la clé, finalement l'algorithme renvoie le résultat qu'on appelle *le texte chiffré* (ou *cipher* en anglais).

L'idée innovante de Pear consiste en l'utilisation de 2 clés de chiffrement : une *clé de substitution* et une *clé de permutation*. L'agence ANS à son tour a proposé d'utiliser un alignement ou *padding* pour les messages.

Malheureusement tous les développeurs de Pear sont occupés à corriger les erreurs dans le nouveau jTelephone 6.0 et tous les agents de l'ANS sont en train de chercher Eduardo Firemano (un des anciens agents qui s'est enfui avec tous les documents confidentiels et les plus grands secrets d'ANS). Ils n'ont donc pas assez de temps pour développer le nouveau algorithme de chiffrement. C'est donc pour ces raisons qu'ils ont décidé de vous confier le travail de développement.

1 Clé de substitution

Cette clé représente la règle qui décrit comment il faut substituer les lettres du message original. Nous allons appeler cette clé *substitution key*. Cette clé est un nombre entier qui donne le décallage dans l'alphabet.

Par exemple : si cette clé vaut 3 et la lettre du message est `A` le résultat après l'application de la substitution est `D`, la lettre `A` devient donc la lettre `D` (`B` devient `E`, etc.). Pour les lettres qui se trouvent à la fin de l'alphabet on utilise l'alphabet de manière circulaire, dans notre exemple `Y` devient `B`, `Z` devient `C`.

On utilise une technique similaire pour les caractères qui représentent les nombres (sur l'alphabet `'0 1 2 3 4 5 6 7 8 9'`). Donc, avec la clé 3 un `'2'` devient un `'5'`, `'3'` devient un `'6'`, etc. Les caractères de ponctuation (e.g. espaces, virgules) restent inchangés.

Exemple : si on chiffre le message `'Hello1'` avec la clé de substitution 3 on obtient `'Khoor4'`.

2 Clé de permutation

La clé de permutation qu'on va appeler *permutation key* fixe la règle de permutation des caractères dans le message. Cette clé est une valeur entière qui donne la taille d'un block du message. Par exemple, si cette clé vaut 3 cela veut dire qu'il faut regrouper les caractères du message par groupes de 3 caractères pour faire la permutation.

Une fois que les caractères sont groupés on peut les réordonner de la façon suivante : le texte chiffré contient d'abord tous les caractères qui se trouvent en premier position dans chaque block ensuite tous les caractères qui se trouvent en seconde position dans chaque block, etc.

Voici un exemple : si on chiffre le message `'Hello1'` en utilisant la valeur 3 pour la clé de permutation, on va regrouper `'Hel'` et `'lo1'` (groupes de 3 caractères) pour ensuite obtenir `'Hleol1'` comme résultat de la permutation.

3 Padding

Pour que le nombre de blocks dans le message soit un nombre entier il faut que la taille du message soit un multiple de la valeur de la clé de permutation. Par exemple, si le message original est `'Hello1'` et la permutation key vaut 3 on a deux blocks de 3 caractères. Malheureusement, si on prend une valeur de permutation de 4, le premier block de taille 4 est complet (`'Hell'`) mais il n'y a pas assez de caractères pour le block 2 (il reste que `'o1'`).

Le but du padding est d'aligner le message pour que la taille du message soit un multiple de nombres de blocks. La règle que l'agence ANS a inventé pour le padding est la suivante :

– on ajoute les symboles `'@#'` à la fin du message original
– on remplit le reste du dernier block avec des symboles `'#'`

Par exemple : si la clé de permutation vaut 3 et qu'il manque un caractère dans le dernier block on ajoute `'@###'` pour remplir les dernier block avec le caractère `'@'` et ajouter un block de `'###'`. Si on veut ajouter le padding au message `'Hello'` (clé de permutation vaut 3), alors le message avec le padding devient `'Hello@###'`.

4 Le chiffrement

On vous demande d'écrire un programme qui permet de chiffrer et de déchiffrer les messages à l'aide de l'algorithme décrit plus haut. Pour chiffrer un message votre programme doit appliquer le padding suivi de deux modifications (substitution et permutation). Pour déchiffrer un message il suffit d'inverser l'ordre des opérations. Remarquez que le chiffrement et le déchiffrement se ressemblent très fort, pensez donc à regrouper les parties de votre code en fonctions.

Votre projet sera testé à l'aide de la commande suivante :

```
python3 projet2.py <action> <substitution key> <permutation key> <message>
```

La valeur `<action>` est l'action à effectuer, soit `'e'` pour chiffrement (ou *encryption* en anglais) ou bien `'d'` pour déchiffrement (ou *decryption* en anglais). Ensuite `<substitution key>` et `<permutation key>` sont les valeurs des clés de substitution et de permutation respectivement. Finalement `<message>` est le message qu'il faut chiffrer ou déchiffrer.

Pour récupérer les paramètres en ligne de commande vous pouvez utiliser la librairie `sys`. Les données se trouvent dans la liste `argv`. Par exemple, pour afficher le premier paramètre passé dans la ligne de commande en Terminal vous pouvez utiliser le code suivant :

```
import sys
print(sys.argv[0])
```

Voici un exemple d'appel à votre programme :

```
python3 projet2.py e 4 3 Hello, world
Message: Hello, world
Substitution Key:  4
Permutation Key:  3
Encrypted: Lp v@isap#p,sh#
```

Voici un autre exemple d'appel à votre programme [1] :

```
python3 projet2.py d 4 3 Ie simlm \ wehvWarwmrmVw@ahrh \ hkrym#
```

Vous avez le droit de supposer que les messages ne contiennent pas de caractères accentués.

On vous demande de découper votre programme en fonctions. Vous devez créer *au moins* les fonctions `encrypt` et `decrypt` qui prennent en paramètre (dans l'ordre) : le message, la clé de substitution et la clé de permutation. Les fonctions `encrypt` et `decrypt` doivent chiffrer et déchiffrer le message passé en paramètre et renvoyer le résultat. On vous encourage à créer d'autres fonctions pour mieux structurer votre code. Pour la réalisation de ce projet vous avez le droit d'utiliser toute la matière vue aux cours et aux TPs jusqu'au chapitre "Matrices".

Les personnages et les situations de ce récit étant purement fictifs, toute ressemblance avec des personnes ou des situations existantes ou ayant existé ne saurait être que fortuite.

Voici encore un message chiffré :

```
C mkxfkDst escm vyzysobcw#svoqoyxZdoobc yFvsyx dw
s# s ojsd o d dqyyobsefos5o#fcw, b ocfbvoovej xcycc
b#yoo w vd:ook xcc ed,e \ zc#ejcfoko \ e \ nc*
kx \ czvb.#c co f n*hcvo .kf lc koo@#
```

Consignes pour la remise du projet

Les consignes pour la remise du projet sont disponibles en ligne sur la page du cours sur l'Université Virtuelle.

Ces consignes sont à respecter *scrupuleusement* ; relisez-les attentivement avant la remise !

Pour toute question concernant l'énoncé du projet adressez-vous à Nikita Veshchikov, e-mail : nikita.veshchikov@ulb.ac.be.

Date limite de remise. Le lundi 10 novembre 2014 à 13h.

1. Que vaut le résultat de déchiffrement ? Mettez le en commentaire au début de votre code.