Université Paris Diderot
L3 Informatique
Sécurité 2017-2018

Projet : Sécurité Informatique

Projet de sécurité informatique de 2017-2018 : Chiffrement par substitution

Membres du groupe :

ZHOU Eric

Fenek Ouarda

Table des matières

I. Chiffrement et Déchiffrement

1) Chiffrement

a) César

L'algorithme de chiffrement de César consiste en un décalage des lettres dans l'alphabet, la clé représentant le décalage. On prendra l'entier modulo 26 (le nombre de lettres dans l'alphabet) pour simplifier le décalage. Les caractères (char) étant des entiers et les lettres représentées les uns à la suite des autres, il suffit d'ajouter la clé à la lettre. Si le résultat obtenu dépasse 'z', on retranchera 26 pour rester dans le domaine des lettres.

« `Hello World !` » avec une clé de 3 donnera « `khoor zruog !` »

b) Vigenère

Le principe de Vigenère est semblable à celui de César. La clé est un mot et chaque lettre de la clé est utilisée une à une pour coder le message lettre à lettre. Ainsi, pour le codage, il suffit d'avoir un compteur de lettres pour savoir quelle est la lettre à utiliser pour l'encodage du message. La lettre une fois codée est la somme de la lettre d'origine et la lettre de la clé. Où la valeur de la lettre de la clé est le numéro de la lettre dans l'alphabet, et la somme d'une lettre et d'un nombre est identique à cette de la méthode de César.

« `Hello World !` » avec la clé « cle » donnera « `jppnz aqcpf !` »

c) Permutation

La méthode de chiffrement par permutation consiste à remplacer une lettre du message initiale par la lettre qui la suit dans la clé ou si c'est la dernière lettre de la clé, la première lettre de la clé. Ainsi, il faut vérifier que la clé est correcte ; la clé ne doit pas contenir deux fois la même lettre. Lors du chiffrement, chaque lettre du texte en clair est à retrouver dans la clé, si elle est retrouvée, alors ce caractère sera remplacé par la lettre suivante de la clé (ou la première si c'était la dernière). Si le caractère à chiffrer n'est pas dans la clé, alors elle n'est pas chiffrée.

« `Hello World !` » avec la clé « cle » donnera « `hceeo wored !` »

d) Utilisation du programme

`java -jar encode.jar <type> <clé> <fichier>`

<type> le type de chiffrement : « c » pour César, « v » pour Vigenère et « p » pour la permutation
<clé> est la clé à utiliser, c'est un entier positif
<fichier> est le fichier contenant le message à traiter

2) Déchiffrement

a) César

Suivant le même principe que le chiffrement, le déchiffrement se fait en retirant la clé (avec un modulo de 26 au préalable) à chaque caractère du message chiffré. Si la lettre obtenue est avant la lettre ‘a’ alors, on ajoute 26 à la valeur de la lettre pour le faire revenir à ‘z’.

b) Vigenère

Tout comme pour le déchiffrement par César, on peut décrypter le message en retranchant la valeur du bon caractère de la clé à la lettre à déchiffrer.

c) Permutation

Toujours sur le même principe que le chiffrement, si une lettre du message codé est dans les permutations (la clé), on la replace par la lettre précédente dans la clé, ou la dernière si c’est la première. On ne change pas une lettre qui n’est pas dans la clé.

d) Utilisation du programme

```
java -jar encode.jar <type> <clé> <fichier>
```

<type> le type de chiffrement : « c » pour César, « v » pour Vigenère et « p » pour la permutation
<clé> est la clé à utiliser, c’est un entier positif
<fichier> est le fichier contenant le message à traiter

II. Décryptage

1) César

a) Par force brute

Sachant que le chiffrement par César ne peut qu'avoir 26 clés uniques, on peut les tester un par un. A chaque fois qu'on teste une clé, on cherche si le message peut avoir un certain sens, cela se fait par la comparaison d'une partie des mots avec un dictionnaire. Si le nombre de mots dépasse un certain seuil (dans notre cas 50%), on peut considérer que la clé est la bonne. On peut retourner le message décrypté. Les autres clés ne sont pas testées, il est peu probable qu'une clé erronée obtienne une bonne proportion de mots correcte.

b) Par analyse statistique

La méthode de César ne change pas les fréquences des lettres, mais uniquement les lettres elles-même. Ainsi, on peut utiliser les statistiques à notre avantage pour décrypter le message. Comme dans le corpus anglais, la lettre la plus utilisée est 'e', il y a de grande chance que la lettre la plus fréquente dans le message chiffré soit un 'e'. Ainsi, on peut calculer la clé en calculant le décalage entre cette lettre et la lettre 'e' et utiliser cette clé pour décrypter le texte.

c) En connaissant un mot

En connaissant un mot du texte, on peut essayer de retrouver le mot codé dans le texte. Il faut commencer par trouver un mot qui contient le même nombre de lettres que notre indice et essayer d'en déduire un décalage possible. Si on trouve un décalage cohérent (toutes les lettres décalées identiquement donnent le mot connu), on considère qu'on a trouvé la clé, on peut décrypter le message avec cette clé.

2) Vigenère

Même si la méthode de Vigenère ne chiffre pas toujours une même lettre par une même autre lettre, il est toujours possible de trouver une clé potentielle grâce aux statistiques à condition de connaitre par avance la longueur de la clé. On remarque que chaque n^{eme} lettre est chiffré par la même lettre de la clé. Ainsi, en séparant chaque lettre dans un tableau de caractères où chaque lettre est chiffrée par un même caractère, on peut retrouver la lettre la plus fréquente et considérer que c'est un 'e' la lettre la plus fréquence en anglais. On peut donc retrouver une clé potentielle pour essayer de décrypter le message.

3) Permutation

Pour cette méthode, il n'y a pas de décalage régulier pour le chiffrement. Alors, il est impossible de déterminer le décalage d'une lettre avec une lettre dont on connaît le décalage. Donc, seules les lettres les plus fréquentes seront décryptées. Nous avons choisi de décrypter les lettres dont la fréquence d'apparition dépasse les 5% avec la même méthode des statistiques des méthodes de chiffrements précédents. Le résultat est un texte où les lettres les plus fréquentes sont potentiellement les bonnes.

`java -jar decrypte.jar <type> <indice> <fichier> <méthode>`
<type> le type de chiffrement : « c » pour César, « v » pour Vigenère et « p » pour la permutation
<indice> l'infirmation qu'on a ; le nombre de lettre dans la clé de Vigenère, où le mot connu pour César, si la méthode de décryptage ne nécessite pas d'indice, il faut mettre un caractère qui ne sera pas pris en compte
<fichier> est le fichier contenant le message à traiter
<méthode> : la méthode a utiliser pour décrypter le message : « b » pour la force brute (César), « s » pour les statistiques (César, Vigenère, Permutation), « m » si on connait un mot

III. Présentation des classes

Package appli

Interface ICode :

- String chiffrer()
- String dechiffrer()
- String decrypter()

Les classes **Encode, Decode** et **Decrypte** qui contiennent sont les classes principales

Package codage

Classe Code :

- Contient deux constructeurs, un pour le chiffrement et de déchiffrement et l'autre pour le décryptage
- Trois méthodes : chiffrer, déchiffrer et décrypter

Les classes **Cesar, Vigenere** et **Permutation** qui contiennent les méthodes pour chiffrer, déchiffrer et décrypter.

Deux enum **TypeCodage** et **TypeChiffrement** qui sont utilisé pour indiquer les fonctions à utiliser

Package utils

Classe FileGetText :

- String getText() pour récupérer le contenu d'un fichier sous forme de String
- List<String> getMots() pour récupérer la liste des mots d'un dictionnaire
- Map<Character, Double> createMap() pour récupérer la fréquences les lettres dans la langue (il peut être amélioré en lisant les informations depuis un fichier)

Classe **Utils** : il contient des fonctions utilitaires :

- int getIntMax(int[]) retourne l'indice de l'élément le plus grand
- int getPopularElement(int[]) : retourne la valeur de l'élément de plus présent
- Map<Character, Double> frequences_text(String) qui retourne les fréquences des lettres

Classe **ValueComparator :** La classe est utilisée pour trier une Map de fréquences

Pour de plus amples informations, consulter la JavaDoc fournie.