Sécurité TP

Bertrand Guillaume
Adequin Renaud
Villette cyril

Utilisation :

Le programme s'utilise comme les exemples montrés dans le sujet (avec une majuscule a Decrypt/Chiffre/Dechiffre soit :

```
java securiteL3/Decrypt c texte 1 motConnu     (Cesar avec un mot connu)
java securiteL3/Decrypt c texte 2              (Cesar par fréquence)
java securiteL3/Decrypt c texte 3              (Cesar par force brute)
java securiteL3/Decrypt v texte                (Vigenère)
java securiteL3/Decrypt v texte taille         (Vigenère avec taille de la clé)
java securiteL3/Decrypt p texte                (Permutation)
```

Tronc Commun :

Nous avons utilisé une interface Code commune étendue par chacune des trois méthodes de codage.
Dans cette interface nous avons définis les fonctions basique chiffre et déchiffre (à l'aide du default introduis avec java 8).
Nous avons également utilisé un StringBuilder au lieu de l'objet String classique pour un gain de temps sur nos concaténation (très nombreuses).
Enfin nous avons, la aussi pour un gain de temps, transformé notre lexique en un nouveau plus rapide à traduire en arbre dans lequel il est stocké, ce afin de pouvoir voir très rapidement si un mot lui appartient.

Cesar :

Notre class Cesar implémente les trois fonctions de décryptage possible. Ces fonctions ont un cœur commun (le split du texte pour travailler mot à mot et la traduction/test du texte a l'aide d'une clé). La force brute essai tous l'alphabet, la méthode par fréquence le fait aussi mais dans un ordre précis (la lettre la plus présente est remplacée par la plus fréquente dans la langue fran^aise et si cela échoue par la suivante...). Enfin la méthode avec mot connu cherche un mot de même taille dans le texte, voit si l'écart entre ses lettres est le même que celui du mot connu et utilise, le cas échéant, l'écart entre ces deux mots comme clé.

Vigenère :

La classe Vigenere implémente le decryptage avec et sans la taille de la clé.
Le decrypt avec la taille de la clé découpe en texte en fonction de la taille de la clé et utilise les fréquences en partant du principe que la lettre e les la

plus présente et calcul le décalage de la lettre e et la lettre avec la plus grade fréquence.
Le decrypt sans la taille de la clé fait appelle au decrypt en connaissant la taille au début une taille de 1 mais teste la clé avec un déchiffre mot par mot et si un mot n'est pas dans le lexique il recommence avec avec une taille plus grande.

Permutation complète :

Lors du début de la permutation nous avons une pile contenant toutes les lettres de l’alphabet de la moins fréquente en bas de la pile à la plus fréquente en haut de la pile.
De plus, il y a une liste comprenant toutes les lettres de l’alphabet comprenant dans un ordre décroissant de la plus présente à la moins présente dans le texte crypté.

Le programme récupère la première lettre de cette liste et la remplace dans tous les mots par la lettre crypté la plus présente et vérifie si ce remplacement est possible :

- Si oui, il associe les deux lettres dans un tuple et les rajoute a la liste et recommence avec les suivantes

- Si non, il met la lettre de la pile dans une pile temporaire et recommence jusqu’a ce que la lettre fonctionne et remet tous les éléments de la liste temporaire dans la liste principale

On continue jusqu’à ce que le nombre de tuple soit égale à 26 (réussi) où jusqu’a ce que la pile principale soit vide (texte encrypté non valide)