TP4 : Casser le chiffrage de Vigenère (partie 1)

Dans ce TP, nous verrons comment casser le chiffre de Vigenère dans les cas où l'on connaitrait la longueur du mot de passe. **Il est recommandé de répondre aux questions posées en écrivant les fonctions demandées avec le bon prototype**.

1 Le TP3 : dernière chance pour le corriger et le compléter (1h maximum)

Vous trouverez, sur Moodle, le résultat des évaluations automatiques du TP3, sur le codage de Vigenère. Si votre programme a été noté comme ayant des problèmes, vous pouvez le corriger dès à présent, et effectuer un nouveau dépôt de votre travail (ce sera le dernier concernant le TP3). N'oubliez pas de suivre à la lettre les instructions de rendu données dans le TP3. Vous devrez effectuer votre rendu avant 9h30.

2 Attaquer le chiffre de Vigenère si on connait la longueur du mot de passe

Imaginons que l'on connaisse la longueur du mot de passe (ici, pour notre exemple, quatre) utilisé pour crypter le message (juste la longueur, pas le mot de passe lui-même). On pourra alors dire que, toutes les quatre lettres, le décalage est le même...

On peut alors étudier le sous-message obtenu en conservant uniquement une lettre sur quatre du message à partir de la première lettre (on ne conserve donc que la 1°, 5°, 9°, etc... lettre de notre message). Par exemple, de notre message (voir énoncé du TP3) codé " vb lsyhnbwnwuspj, d' otp wbbenoop csfj !", on obtiendrait "vywsobos" (les symboles et espaces ne sont pas comptés).

Dans ce sous-message, on sait que chaque lettre a subi le même décalage (dans notre exemple, ces lettres ont subi un décalage de 10, correspondant à la lettre "j" - la première lettre - de notre mot de passe "java") : elles ont donc subi un chiffrage de César. On peut donc chercher, pour trouver le décalage qui a été appliqué, quelle lettre apparaît le plus souvent. En répétant ce processus quatre fois (quatre étant la longueur du mot de passe), en prenant une lettre sur quatre à partir de la 1° lettre, puis à partir de la 2° lettre, etc... on peut déchiffrer le message entier.

Questions

Nous allons étudier le fichier *text_ crypte_ vigenere_ petit.txt*, dont le mot de passe est de longueur 5.

1. Proposez une fonction **void normaliserFichier(FILE *in, FILE *out)** qui recopie le texte du fichier **in** dans le fichier **out** en ne conservant que les lettres non accentuées, et transformant les majuscules en minuscules (on évite les signes de ponctuation, les chiffres, les lettres accentuées, ...).

2. Proposez une fonction **void decouperFichier(FILE *in, FILE *out, int longueur, int debut)**, qui recopie, dans le fichier **out**, certaines lettres du fichier **in**. La fonction doit se placer d'abord sur la lettre à la position **debut**, et la recopier dans le fichier **out**. Puis, elle se décale de **longueur** lettres, et recopie la lettre qui s'y trouve. Elle se décale de nouveau de **longueur** lettres et recopie le caractère qui s'y trouve. On continue ainsi de recopier des lettres toutes les **longueur** lettres, jusqu'à arriver à la fin du fichier.

3. Proposez une fonction permettant de récupérer le mot de passe du texte crypté. Votre fonction prendra en paramètre le fichier contenant le texte crypté, ainsi que la longueur supposée du

mot de passe (pour notre fichier de test, c'est 5). Votre fonction devra, dans un premier temps, normaliser le texte d'entrée avec la fonction **normaliserFichier**, puis trouver le mot de passe à l'aide de la fonction **decouperFichier** pour découper le fichier en 5 parties.

Votre fonction devra, pour renvoyer une chaîne de caractères, soit allouer dynamiquement un tableau de caractères, soit allouer un tableau statique suffisamment grand pour contenir les caractères du mot de passe. Tous les mots de passe feront moins de 100 caractères : dans ce cas, quelle est la taille du tableau à allouer ?

4. Proposez un programme permettant de décoder complètement le texte crypté avec un mot de passe de longueur 5. Vous devrez restituer le texte en respectant les minuscules, majuscules et signes de ponctuation. Votre programme devra être appelé à l'aide de la ligne de commande suivante :

```
vigenere_decodage entree.txt taille_mot_de_passe sortie.txt
```

Votre programme devra être rendu pendant ce TP, en suivant les instructions de la section suivante. Si vous souhaitez tester votre programme sur d'autres textes que vous aurez cryptés vous-même, utilisez des textes assez longs, afin que les sous textes soient représentatifs du français et que la lettre e soit la plus fréquente.

3 Rendu du décodage de Vigenère

Vous devez rendre le travail réalisé pour le TP4 sur Moodle ou par mail, à l'adresse

chaussard.laga+idb@gmail.com

si vous n'avez pas de compte Moodle (et seulement à cette condition).
Ce rendu doit suivre un format spécifique. Votre programme devra s'exécuter à l'aide de la commande

```
./vigenere_decodage mon_message.txt taille_mot_de_passe sortie.txt
```

où *mon_message.txt* est le message codé, *taille_mot_de_passe* est la longueur du mot de passe utilisé pour le codage, et *sortie.txt* contient le message déchiffré. De plus, votre programme devra compiler à l'aide de la commande

```
gcc vigenere_decodage.c -o vigenere_codage
```

Si votre ligne de compilation est plus complexe, vous devrez alors la spécifier dans un *Makefile*. Tous vos fichiers devront être directement placés dans un fichier zip (et ne pas être dans un sous dossier du fichier zip), dont le nom sera

```
VotreNom_VotrePrenom_VotreNumeroEtudiant.zip
```

sans aucun espace (si votre nom ou votre prénom contiennent un espace, ne les faîtes pas figurer). Les formats de compression acceptés sont .zip, .tgz, .tar.gz, .7z, .rar.

Si vous avez un binôme, il vous faudra écrire son nom, prénom et numéro d'étudiant séparés par des espaces, en plus des vôtres, dans un fichier *binome.txt*, dont le format sera

```
Nom1 Prenom1 Numero_etudiant1
Nom2 Prenom2 Numero_etudiant2
```

Vous ne devez rendre le fichier qu'une seule fois par binôme.