Compte Rendu
Outil automatique de décryptage (sujet 4)

Chahi Rabie Ala Eddine, Arar Akram Mohamed, Attouche Maher, Benammar Ismail,
Chergou Amine Ismaïl, Ben Mallem Amir Fayçal, Mohammed Seghir Said, Yasmine Keskes.

Mardi 21 Mai 2019

Table des matières

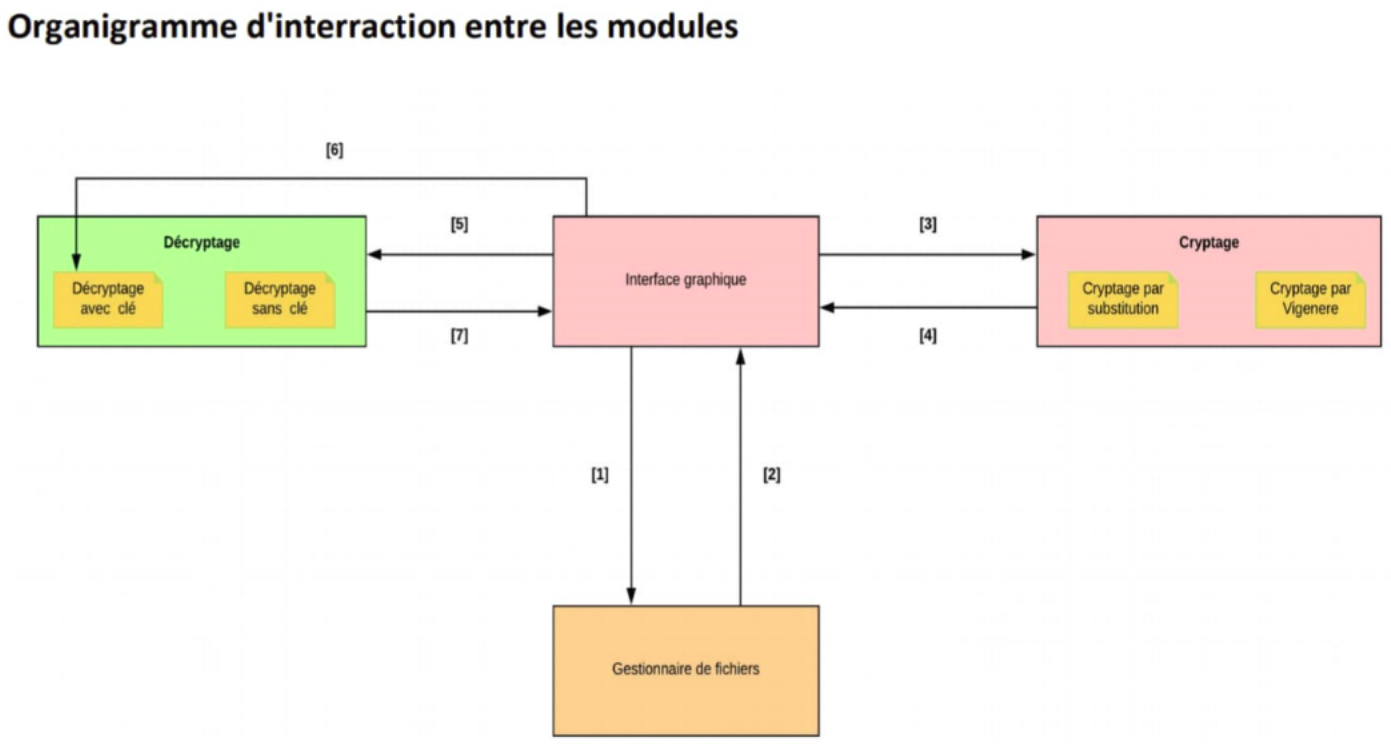
1 Introduction

La cryptologie est la science du secret des mathématiques et plus récemment de l'informatique, elle a permis le développement des échanges électroniques, tant dans le domaine industriel et bancaire que dans celui du commerce en ligne et récemment celui des relations entre les citoyens et les administrations. De ce fait un outil dédié à la cryptographie se veut d'une très grande utilité pour les chercheurs ou des personnes ayant de maigres connaissances dans le domaine et souhaitant apprendre davantage.

Dans le cadre du module *Projet Informatique* du semestre 6 en licence à l'UVSQ, le sujet **Outil automatique de décryptage** nous a été attribué. Ce projet a pour but de fournir une application qui permet d'interagir avec l'utilisateur à travers une interface graphique intuitive, afin de crypter, décrypter ou d'effectuer une cryptanalyse sur un texte soit par la méthode de Vigenère ou par substitution.

Après l'étude en détail des algorithmes de cryptographie et de la cryptanalyse dans le cahier des charges, et l'analyse de la mise en relation des différents modules correspondant aux différentes fonctionnalités, nous avons pu implémenter notre outil automatique de décryptage. Ce document va donc présenter le fonctionnement et l'architecture de l'application avec les explications techniques sur les points délicats, ainsi qu'un bilan technique sur l'application et un bilan humain sur l'équipe de conception.

2 Architecture de l'application



L'étude des algorithmes et des fonctionnalités et contraintes imposées par le client nous ont permis d'elaborer l'organigramme ci-dessus qui présente l'architecture de l'application. Chaque module correspond à un ensemble de fonctionnalités assurant le bon fonctionnement de l'application.

L'outil comporte 4 modules :

- Le module d'interface graphique gère l'interaction entre l'outil et l'utilisateur.
- Le module gestionnaire de fichiers assure les opérations de chargement et de sauvegarde de texte dans des fichier.
- Le module de cryptage assure le cryptage d'un texte donné en utilisant une clé de cryptage et ce soit par substitution ou selon la méthode de Vigenère.
- Le module de décryptage assure le décryptage d'un texte crypté auparavant soit par substitution soit par la méthode de Vigenère.

L'utilisateur commence par introduire un texte ou bien sélectionner un fichier qui contient du texte pour charger ce dernier.
Si l'utilisateur désire charger un texte depuis un fichier, le chemin de celui-ci est envoyé vers le *module gestionnaire de fichiers* (flèche 1) qui s'occupera de charger le texte et de le renvoyer vers le *module Interface graphique* (flèche 2), après cela l'utilisateur peut soit crypter le texte soit le décrypter si ce dernier est déjà crypté. Dans le cas où il désire crypter le texte, il devra saisir une clé et choisir le type du cryptage, ensuite le texte ainsi que la clé et la méthode choisie seront envoyés vers *le module de cryptage* (flèche 3) qui se chargera de crypter le texte et de renvoyer le résultat (flèche 4) vers *le module interface graphique*. Dans le cas contraire où l'utilisateur désire décrypter le texte, il devra renseigner la méthode qui a servi pour crypter le texte et éventuellement la clé qui a servi, si celle-ci est connue. Dans le cas où la clé est connue elle est envoyée depuis le module *Interface graphique* au *sous module décryptage avec clé* (flèche 6). Le texte ainsi que le type de cryptage seront aussi envoyés depuis l'interface graphique vers le *module décryptage* (flèche 5), qui se chargera de décrypter le texte et de renvoyer le résultat au *module interface graphique* (flèche 7). Au final si l'utilisateur désire exporter le texte vers un fichier, le texte ainsi que le chemin du fichier seront envoyés vers le *module gestionnaire de fichiers* (flèche 1) qui se chargera de sauvegarder le texte dans le fichier.

3 Prérequis

Afin de garantir le bon fonctionnement de l'outil certaines règles doivent être réspectées :

- Les textes entrés afin de les crypter devront être écrit en langue française.
- Les textes entrés afin de les cryptanalyser doivent être cryptés à partir d'un texte français correctement orthographié.
- Les textes très courts peuvent donner des résultats erronés lors de la cryptanalyse, pour cela il est préférable d'avoir un texte avec une longueur raisonnable.

Remarque : Ceci est dû au fait que l'attaque statistique opère sur des textes suffisamment longs.

4 Description du fonctionnement

Dans cette partie nous allons mettre en évidence le fonctionnement interne de notre application. Pour cela, nous allons aborder avec précision chaque module de l'organigramme ainsi que les algorithmes implémentés. Enfin nous allons illustrer le fonctionnement de notre application par un exemple de déchiffrement d'un message sans clé.

4.1 Fonctionnement des modules

4.1.1 Interface graphique

C'est le module qui s'occupe des interactions entre l'utilisateur et notre outil, en effet ce module offre à l'utilisateur la possibilité d'insérer ou de charger un texte et d'effectuer un des types de traitement que l'outil propose.

Liste des fonctionalités

- Choisir l'action à effectuer (Cryptage/Décryptage).
- Choisir le type de cryptage/décryptage (Substitution/Vigenère).
- Introduire un texte manuellement.
- Choisir un fichier source pour importer le texte Contenu dans le fichier.
- Choisir la clé avant de lancer le cryptage.
- Choisir de visualiser le déroulement du cryptage.
- Choisir de visualiser le déroulement du décryptage.
- Visualiser le texte en sortie.
- Choisir d'exporter le texte résultant vers un fichier.

4.1.2 Cryptage

Ce module assure le chiffrement d'un texte passé en paramètre en utilisant la méthode de Vigenère ou de substitution.

a) Cryptage par substitution : Ce sous module assure le cryptage d'un texte passé en paramètre avec une clé entrée par l'utilisateur ou générée aléatoirement.

Liste des fonctionalités

- Crypter un texte par substitution.
- Visualiser le déroulement du cryptage étape par étape.

b) Cryptage par Vigenère : Ce sous module assure le cryptage d'un texte passé en paramètre avec une clé entrée par l'utilisateur ou générée aléatoirement .

Liste des fonctionalités

- Crypter un texte par la méthode de blaise Vigenère.
- Visualiser le déroulement du cryptage étape par étape.

4.1.3 Décryptage

Ce module assure le décryptage d'un texte donné crypté auparavant soit par la méthode de substitution soit par la méthode de Vigenère.

a) Décryptage avec clé : Ce sous module assure le décryptage d'un texte donné en entrée et ce, en utilisant deux paramètres qui sont le type du cryptage ainsi que la clé utilisée lors du cryptage.
Remarque : ce sous module trouve le texte résultant identique à 100% au texte original.

Liste des fonctionalités

- Retrouver le texte original à partir d'un texte crypté par substitution et ce en utilisant une clé de décryptage.
- Retrouver le texte original à partir d'un texte crypté par la méthode de Vigenère et ce en utilisant une clé de décryptage.
- Visualiser le déroulement du décryptage étape par étape.

a) Décryptage sans clé : Ce sous module assure le décryptage d'un texte donné en entrée à l'aide d'une attaque statistique, en utilisant un seul paramètre qui est le type du cryptage, ce sous module aide à retrouver une grande partie du texte original à la différence du précédent qui a pour rôle de retrouver 100% du texte original.

Liste des fonctionalités

- Retrouver une grande partie du texte original à partir d'un texte crypté par substitution.
- Retrouver une grande partie du texte original à partir d'un texte crypté par la méthode de Vigenère.
- Visualiser le déroulement du décryptage étape par étape.

4.1.4 Le module gestionnaire de fichiers

Ce module permet de gérer du texte en entrée/sortie sur des fichiers, il offre la possibilité d'importer un texte clair ou chiffré contenu dans un fichier et de sauvegarder un texte résultant d'un des deux processus cryptage/décryptage dans un fichier.

Liste des fonctionalités

- Charger un texte depuis un fichier pdf/word/txt vers l'interface.
- Sauvegarder un texte affiché sur l'interface dans un fichier pdf/word/txt.

4.2 Description des algorithmes proposés

4.2.1 Chiffrement par substitution mono-alphabétique

Cet algorithme chiffre le texte clair modifié au préalable (sans espaces et le tout en majuscules), il prend en entrée la clé de taille 26, il effectue des permutations de sorte que chaque lettre de l'alphabet est remplacée par la lettre qui correspond à son rang dans la clé.

Exemple : On souhaite chiffrer le mot "Exemple" en utilisant la clé AZERTYUIOPQSDFGHJKLMWXCVBN
.

- On fait correspondre la clé de taille 26 au 26 lettes de l'alphabet :
  AZERTYUIOPQSDFGHJKLMWXCVBN
  ABCDEFGHIJKLMNOPQRSTUVWXYZ

- Cela veut dire que tous les A dans le clair seront transformés en A dans le chiffré, tous les B en Z, les C en E .....etc. Donc le mot Exemple est chiffré en "TVTDHST" .

4.2.2 Chiffrement par Vigenère

Cet algorithme chiffre le texte clair modifié au préalable (sans espaces et le tout en majuscules), il prend en paramètre une clé qu'il répète tout au long du texte et effectue une addition modulo 26 entre l'indice dans l'alphabet de chaque lettre de la clé avec la lettre du texte qui lui correspond.

Exemple : On souhaite chiffrer le mot "Exemple" en utilisant la clé "KEY" :

- On commence par répéter la clé tout le long du texte.
- Afin de trouver le chiffré on fait l'addition modulo 26 du rang de la lettre du clair avec le rang de la lettre de la clé.
  Clair : EXEMPLE [4 23 4 12 15 11 4]
  Clé : KEYKEYK [10 4 23 10 4 23 10]
  Resultat : OBCWTJO. [14 1 2 22 19 9 14]

4.2.3 Déchiffrement par substitutions mono-alphabétique

Cet algorithme prend en entrée une clé de taille 26 qui contient les permutations des lettres de l'alphabet, exemple : on permute A par D donc à l'indice zéro qui correspond au rang de la lettre A dans l'alphabet on trouve la lettre D. Pour chaque lettre du texte chiffré on parcourt la clé dès qu'on trouve la lettre en question on prend son indice dans la clé pour retrouver le clair de cette lettre il suffit de voir la lettre qui correspond à cet indice dans l'alphabet.

Exemple : On souhaite déchiffrer le mot "TVTDHST" en utilisant la clé AZERTYUIOPQSDFGHJKLMWXCVBN
.

- On fait correspondre la clé de taille 26 au 26 lettes de l'alphabet :
  AZERTYUIOPQSDFGHJKLMWXCVBN
  ABCDEFGHIJKLMNOPQRSTUVWXYZ

- Cela veut dire que tous les A dans le chiffré seront transformés en A dans le clair, tous les Z en B, les E en C .....etc. Donc le mot TVTDHST est déchiffré en "EXEMPLE" .

4.2.4 Déchiffrement de Vigenère

Cet algorithme permet de déchiffrer un texte chiffré à l'aide de la clé passée en paramètre, il découpe le texte en blocs qui font la même taille que la clé, puis il correspond chaque bloc crypté à la clé et soustrait le rang (dans l'alphabet) de la lettre de la clé du rang de la lettre du texte chiffré le tout modulo 26.

Exemple : On souhaite déchiffrer le mot "OBCWTJO" en utilisant la clé "KEY" :

- On commence par répéter la clé tout le long du texte.
- Afin de trouver le clair on fait la soustraction modulo 26 du rang de la lettre du chiffré avec le rang de la lettre de la clé.

Chiffré :	OBCWTJO	[14 1 2 22 19 9 14]
Clé :	KEYKEYK	[10 4 23 10 4 23 10]
Resultat :	EXEMPLE.	[4 23 4 12 15 11 4]

4.2.5 Algorithme d'analyse des fréquences

Cet algorithme prend en entrée un texte chiffré, et remplit la structure de données qui est une table de hachage, contenant les lettres de l'alphabet, les digrammes et les trigrammes avec leurs fréquences et positions dans le texte.

4.2.6 Algorithme de cryptanalyse de la substitution monoalphabétique

Cet algorithme a pour rôle de trouver une grande partie du texte chiffré sans la clé, dans un premier temps on génére une clé aléatoire puis on déchiffre le texte à l'aide de cette clé, une fois le texte déchiffré on le divise en quadrigrammes et on calcule le taux de compatibilité entre les fréquences des quadrigrammes dans le déchiffré et ceux de la langue française, on garde la permutation de la clé et son taux de compatibilité, ensuite on refait le même procédé en permutant les lettres de la clé si on trouve un taux de compatibilité meilleur par rapport à l'ancien on remplace l'ancienne permutation par la nouvelle permutation. Dans le cas où le taux de compatibilité ne change pas au bout de plusieurs itérations alors la permutation de la clé est probablement la bonne.
Remarque : dans cette algorithme on a utilisé les 30 000 quadrigrammes les plus fréquents dans la langue française.

Exemple : On possède un texte chiffré par substitution que l'on souhaite cryptanalyser :
"XOUENFTZOUXFZFBLAHAIEJACJF"

- On commence par choisir une clé aléatoire de taille 26 :
  BDCSFGMPILKJHNORQZETUXWVAY
- On déchiffre le texte en utilisant cette clé ce qui nous donne
  VOUSNETROUVEREAJYMYISLYCLE.
- On calcule le taux de compatibilité du texte déchiffré

$$Taux = \frac{Prob(VOUS) + Prob(OUSN) + Prob(USNE) + Prob(SNET).....}{n}$$

  avec n le nombre de quadrigrammes dans le texte.
- On Fait des permutations dans la clé afin d'améliorer le taux
  Si on permute le A avec le Y , le résultat du déchiffrement sera :
  VOUSNETROUVEREZJAMAISLACLE
  On remarque que le texte est plus lisible et son taux de compatibilité est meilleur donc on garde cette nouvelle clé.

- On essaye de trouver une permutation qui améliore le taux mais au bout de plusieurs tests aucune permutation n'améliore le taux donc on garde cette clé.

4.3 Algorithme de cryptanalyse de Vigenère

Cet algorithme a pour rôle de retrouver une grande partie du texte original à partir d'un texte crypté, en premier lieu cet algorithme commence par trouver la longueur de la clé qui a servi lors du cryptage et ce en utilisant l'algorithme de Kasiski expliqué ci-desous.

4.4 Algorithme de Kasiski

Afin de trouver la longueur de la clé, l'algorithme commence par trouver les polygrammes fréquents dans le texte, il calcule ensuite les distances qui séparent chaque occurrence du même polygramme ainsi que les diviseurs de ces distances. De cette manière le diviseur le plus commun entre les différents polygrammes et probablement la longueur de la clé.

4.4.1 Algorithme de l'indice de coïncidence

Pour confirmer que la longueur (n) de la clé trouvée est bien celle utilisée pour crypter le texte, le texte est divisé en n parties de sorte que chaque partie est supposée être cryptée en utilisant le même caractère de la clé, ensuite l'indice de coïncidence de chaque partie est calculé.
Si l'indice est proche de 0.0735 qui est l'indice de coïncidence moyen d'un texte écrit en français alors la longueur trouvée est bien celle qui a servi pour crypter le texte.
Sinon le même processus est refait pour les autres longueurs probables.
Après avoir trouvé la longueur n de la clé, l'algorithme découpe le texte en n parties. Dans ce cas chaque partie (Li) a été cryptée en utilisant la lettre i de la clé. Afin de retrouver la i ème lettre de la clé, une attaque statistique est utilisée sur la partie (Li).

4.4.2 Algorithme de l'attaque statistique

Afin de retrouver la lettre (décalage) i qui a servi à crypter un texte, une analyse est faite sur le texte pour retrouver le caractère le plus fréquent, ce caractère représente forcément le caractère E de l'alphabet crypté en utilisant le décalage i. donc afin de retrouver le décalage i, il suffit de soustraire le rang de E (4) du rang du caractère le plus fréquent.
Une fois la clé retrouvée il suffit de décrypter le texte en utilisant cette dernière.

Exemple d'algorithme de cryptanalyse de Vigenère en utilisant l'algorithme de Kasiski, Indice de coïncidence ainsi que l'algorithme de l'attaque statistique :

On possède un texte chiffré par Vigenère que l'on souhaite cryptanalyser :
"WIQCEEOVRBIQWIQAYGXQCCSNYVRKML"

1. On commence par rechercher les polygrammes les plus fréquents.
   On trouve que le Trigramme(WIQ), Bigramme(IQ), Bigramme(VR) sont fréquents et se répètent plusieurs fois.

2. On calcule les distances qui séparent chaque répétition du même polygramme.
   WIQ : 12
   IQ : 9 , 12
   VR : 18

3. On calcule ensuite les diviseurs de chaque groupe de distances.
   Diviseurs (WIQ) : 1,2,3,4,6
   Diviseurs (IQ) : 1,3,6,9,12
   Diviseurs (VR) : 1,2,3,6,9

4. On trouve que les diviseurs les plus fréquents sont 6 et 3. Donc la clé est probablement de longueur 3 ou 6

5. On essaye Longeur=3.

6. On divise le text chiffré en 3 groupes :
   g1 g2 g3
   W I Q - Tous les éléments du groupe g1 ont été crypté en utilisant la même partie de la clé.
   C E E
   O V R - On Trouve que l'élément le plus répété dans le g1 est O qui correspond forcément au E.
   B I Q
   W I Q - On calcule alors la 1ére lettre de la clé O-E=K.
   A Y G
   X Q C - On fait de même pour les deux groupes g2 et g3.
   O S N
   Y V R - On trouve la clé = KEY.
   K M L

7. On décrypte le texte en utilisant la clé KEY
   on trouve "MESSAGERTRESMEQUINMESOPORTAIN".

4.5 Fonctionnement de l'outil par un exemple de cryptanalyse par Vigenère

Un utilisateur choisit de déchiffrer un message crypté par la méthode de Vigenère sans avoir la clé, il a le choix entre, entrer le texte via l'interface graphique ou bien de le charger depuis un fichier, on suppose que le texte est entré via l'interface, ceci aura pour effet d'initialiser un objet de type polygramme qui se chargera de calculer les fréquences des lettres, digrammes, trigrammes ainsi que des quadrigrammes et récupérer leurs positions dans le texte puis stocke les informations de ces derniers successivement dans les structures de données suivantes *map<string,Polygramme *> bigram, map<string,Polygramme *> trigram, map<string,Polygramme *> quadrigram*. La méthode *TextV : :findKeySize()* qui est une implémentation de l'algorithme de Kasiski sera appelée pour trouver la taille de la clé utilisée ceci en calculant les distances entre les différents polygrammes, puis trouve les diviseurs de ces distances et cherche le diviseur commun le plus répété qui sera probablement la taille de la clé la taille de la clé sera confirmé grâce à l'indice de coïncidence. Une fois la taille de la clé retrouvée on fait appel à la méthode *TextV : :findKeyUsed(KeySize)* qui prend en paramètre la taille de la clé, elle aura pour rôle de retrouver la clé utilisée et cela en faisant une attaque statistique c'est-à-dire, on effectue une analyse fréquentielle sur l'alphabet L1 qui correspond à l'ensemble des lettres du texte chiffré à partir de la première lettre de la clé afin d'obtenir cette dernière et on réitère l'opération sur L2 jusqu'à Ln ou n correspond la taille de la clé . Une fois la clé retrouvée il nous reste plus qu'à déchiffrer le texte, et ce en utilisant la méthode *TextV : :decrypt(Key)* qui prend en paramètre la clé, cette méthode effectue une soustraction modulo 26 entre l'indice dans l'alphabet de chaque lettre du texte avec l'indice de l'alphabet de la lettre de la clé qui lui correspond.

5 Choix du language de programmation :

Nous avons opté pour le *C++* car c'est un langage hybride offrant la possibilité d'utiliser l'aspect procédural afin de gérer tout ce qui touche aux routines mathématiques calculatoires qui sont le pilier de notre projet, ainsi que la gestion de fichiers.
Le **C++** offre aussi la possibilité d'utiliser l'aspect orienté objet, en effet, tout dans notre projet peut être vu comme étant un objet, que ce soit un texte, un polygramme ou bien même une clé de cryptage ou de décryptage, nous avons également eu besoin de la notion d'héritage afin de pouvoir implémenter notre code d'une façon optimal, ainsi nous avons par exemple pu faire hériter deux différents types de texte d'une classe mère (Text) et qui sont, TextV (Vigenère) et TextS (Substitution) qui ont chacune des méthodes propres à

elles-mêmes. De plus, nous avons aussi eu besoin de les notions de polymorphisme et de surcharge, en effet la méthode de décryptage diffère selon la présence ou non d'une clé de décryptage et selon le type du texte à décrypter (Substitution ou bien Vigenère). Nous avons aussi eu besoin de l'aspect orienté objet dans le but de modulariser le code, de l'aérer et d'en faciliter la modification.

Choix du Framework **Qt** :
Qt possède une documentation assez complète qui explique le fonctionnement de certaines classes en détail ce qui nous a été grandement utile lors de l'implémentation de notre interface graphique. Le framework **Qt** possède un ensemble de bibliothèques permettant de développer des application complexe.

Enfin nous avons à disposition un ensemble d'outils facilitant le développement d'applications. Il faut savoir que comme le framework fonctionne sur les trois systèmes d'exploitation (Mac OS, Linux et Windows), ces outils sont eux aussi compatibles avec l'ensemble des systèmes.

6 Description des points délicats de la programmation

- Lors de l'implémentation de la cryptanalyse par substitution nous avons rencontré un problème qui n'était pas évident à détecter, en effet lorsqu'on rencontrait des quadrigrammes avec une fréquence nul dans la langue française cela avait pour effet de mettre le taux de compatibilité à moins l'infinie cela est dû à l'utilisation de la fonction logarithme, afin de remédier à ce problème nous avons dû donner une fréquence très faible pour l'ensemble des quadrigrammes ayant la fréquence nul.
- Lors de l'implementation de la cryptanalyse par substitution nous avons remarqué que lorsqu'on générait une première clé totalement aléatoire, l'exécution de l'algorithme prenait du temps, afin de remédier à ce problème et d'optimiser le temps d'exécution nous avons opté pour la génération d'une première clé par le biais d'une attaque statistique sur le texte, celle-ci étant très proche de la clé effective elle diminue considérablement le temps d'exécution.
- Lors de la réalisation du déroulement de la cryptanalyse de Vigenère on a eu des difficultés à récupérer quelques informations nécessaires au bon fonctionnement du déroulement des différentes étapes du décryptage sans clé, afin de remédier à ce problème nous avons décidé d'envoyer ces informations depuis la méthode decrypt() vers la méthode responsable du déroulement.

7 Comparaison entre l'estimation et l'implémentation

Module de l'application	**Coût en temps**	**Coût en nombre de lignes**	**Personnes en charge**
Interface graphique	Estimation : 18 heures Implémentation : 16 heures	Estimation :800 lignes Implémentation :732 lignes	Atouche Maher & Chahi Rabie Ala Eddine
Cryptage et décryptage par substitution	Estimation : 6 heures Implémentation : 2 heure	Estimation :100 lignes Implémentation : 65 lignes	Mohammed Seghir Said & Yasmine Keskes
Cryptage et décryptage par Vigenère	Estimation : 6 heures Implémentation : 2 heures	Estimation : 100 lignes Implémentation : 15 lignes	Amir Fayçal Ben Mallem & Benammar Ismail
Géstionnaire de de fichiers	Estimation : 12 heures Implémentation : 12 heures	Estimation : 200 lignes Implémentation :160 lignes	Arar Akram Mohamed & Chergou Amine Ismaïl
Cryptanalyse par substitution	Estimation : 75 heures Implémentation : 50 heures	Estimation : 350 lignes Implémentation : 140 lignes	Benammar Ismail & Yasmine Keskes & Amir Fayçal Ben Mallem
Cryptanalyse de Vigenère	Estimation : 65 heures Implémentation : 35 heures	Estimation : 250 lignes Implémentation : 280 lignes	Chergou Amine & Mohammed Seghir & Arar Akram Mohamed
Déroulement des étapes crptage/décryptage	Estimation : 36 heures Implémentation : 28 heures	Estimation : 800 lignes Implémentation : 800 lignes	Attouche Maher & Chahi Rabie Ala Eddine
Le coût total	Estimation :182 heures Implémentation : 145 heures	Estimation : 2600 lignes Implémentation : 2192 lignes	

On peut remarquer dans certains cas des différences plus ou moins grandes entre les estimations en nombre de lignes et d'heures de travail et le coût réel et ce pour des raisons différentes :

- Dans le cas du cryptage/décryptage avec clé que ce soit par substitution ou selon la méthode de Vigenère, on peut remarquer que le travail a été plus court que prévu, cela est dû au fait que nous avions prévu un temps pour la documentation, mais vu que durant ce semestre nous avons eu une UE de Cryptographie où on a traité ces différentes méthodes de cryptage/décryptage le travail de recherche a été minime.
- Dans le cas de la cryptanalyse de Vigenère et Substitution, on peut remarquer qu'il y a une différence entre le taux horaire estimé et celui réel, ceci peut être expliqué par le fait que nous avions estimé que le taux d'erreur initial serait un peu élevé et que nous aurions à l'améliorer au cours du temps, chose qui a été faite plus rapidement que prévu.

8 Conclusion technique

Notre outil satisfait actuellement la totalité des objectifs fixés par le client, en plus de ces objectifs notre outil répond également à certaines éxigences supplémentaire motivées pour certaines par le client, d'autres par nous-même en guise de geste commercial.
En effet notre outil est capable de crypter/décrypter un texte soit par substitution ou selon la méthode de Vigenère et ce avec ou sans clé le tout via une interface graphique ergonomique et facile d'accès, en plus l'équipe Cryptopher s'est investi afin de réaliser les fonctionnalités supplémentaires demandées par le client lors des réunions, tel que l'affichage dynamique des étapes de déroulement des cryptage/décryptage/cryptanalyse ainsi que le chargement et l'exportation de texte depuis et dans des fichiers de différents formats tels que txt,word,pdf.
Cependant l'équipe n'a pas eu le temps de finir le déroulement de la cryptanalyse par substitution, cette fonctionnalité étant presque prête elle sera finalisée avant le 28/05/2019.

Lors de la conception l'équipe a veillé à ce que l'outil soit parfaitement modulaire afin de faciliter la modification et l'ajout de fonctionnalités. En effet certaines perspectives d'améliorations sont envisagées, tels que l'ajout de nouveaux types de chiffrement tel que Enigma,Hill,ElGamal... etc. L'ajout de la langue anglaise est aussi envisageable.

9 Conclusion sur l'organisation interne au sein du projet

Durant 16 semaines, notre équipe s'est surpassé afin de réaliser le projet du module IN608 intitulé *outil automatique de décryptage*, ce projet fut l'occasion pour les membres de l'équipe de mettre à profit les acquis et les bonnes pratiques accumulées durant nos trois années de licence, en effet dès la première réunion tout le monde était d'accord pour dire que l'organisation serait la clé de voûte du succès de notre projet, pour cela, il nous tenait à cœur de réaliser un outil qui satisfait la totalité des exigences du client, et ce, en élaborant des plannings qui ont toujours été respectés, ainsi qu'une répartition équitable des tâches. Nous avons décidé de désigner un chef de projet qui avait pour rôle de veiller à ce que le travail soit effectué en respectant les délais mais aussi le maintien d'une bonne entente et communication au sein du groupe. Au final le projet s'est avéré être un point indispensable dans notre apprentissage ainsi qu'une excellente première expérience de travail dans une grande équipe, la réalisation de cet outil nous a beaucoup apporté tant sur le plan professionnel que sur le plan humain.