Rapport Final

Alabi Steve - Benyamna Younes - Capdenat Nicolas-
Chouipe Thibaut - El Harti Zakaria - Lienhardt Florian

Chef de projet : Benyamna Younes

Sous la direction de Mme Kloul

Outil automatique de décryptage

29 mai 2017

1 Introduction

Après avoir réalisé le cahier des charges, qui exprime les besoins/attentes du client ainsi que les contraintes à respecter, et après avoir décrit dans le cahier des spécifications comment les exigences fonctionnelles vont être implémentées, nous nous sommes donc lancés dans la réalisation du produit.

Dcrypt est maintenant disponible et utilisable par le client. Un manuel d'utilisation détaillé lui sera également fourni pour expliquer le fonctionnement de l'application.

Dans ce rapport, il s'agira de s'intéresser tout d'abord à la logique/architecture de l'application et au choix du langage. Enfin, nous pouvons nous pencher sur la partie technique du produit. Pour cela, nous montrerons le fonctionnement de l'application à l'aide de tests, puis nous parlerons de l'organisation interne et pour finir, nous verrons la comparaison du nombre de lignes de code réelles avec les estimations faites dans le cahier des charges.

2 Objectif de l'application

Le but du projet est donc de développer un logiciel pour automatiser le décryptage d'un texte. Le produit est réalisé afin de satisfaire le client, et pour se faire il doit répondre à toutes ses attentes.
Ainsi, le client peut crypter ou décrypter un texte ou un fichier et ce par la méthode de Substitution ou de Vigenère. Mais il lui est aussi possible d'effectuer indépendamment une analyse fréquentielle sur un texte donné.

3 Architecture de l'application

Interface graphique :
- bouton cryptage
- bouton decryptage
- bouton substitution
- bouton Vigenère
- affichage de texte(complet et partiel)
- affichage pour la clé de substitution
- bouton Francais(decryptage)
- bouton Anglais(decryptage)
- affichage pour l'analyse fréquentielle
- charger un fichier texte
- sauvegarder un fichier texte
- créer un nouveau fichier texte(resultats)
- Demander clef de Vigenère

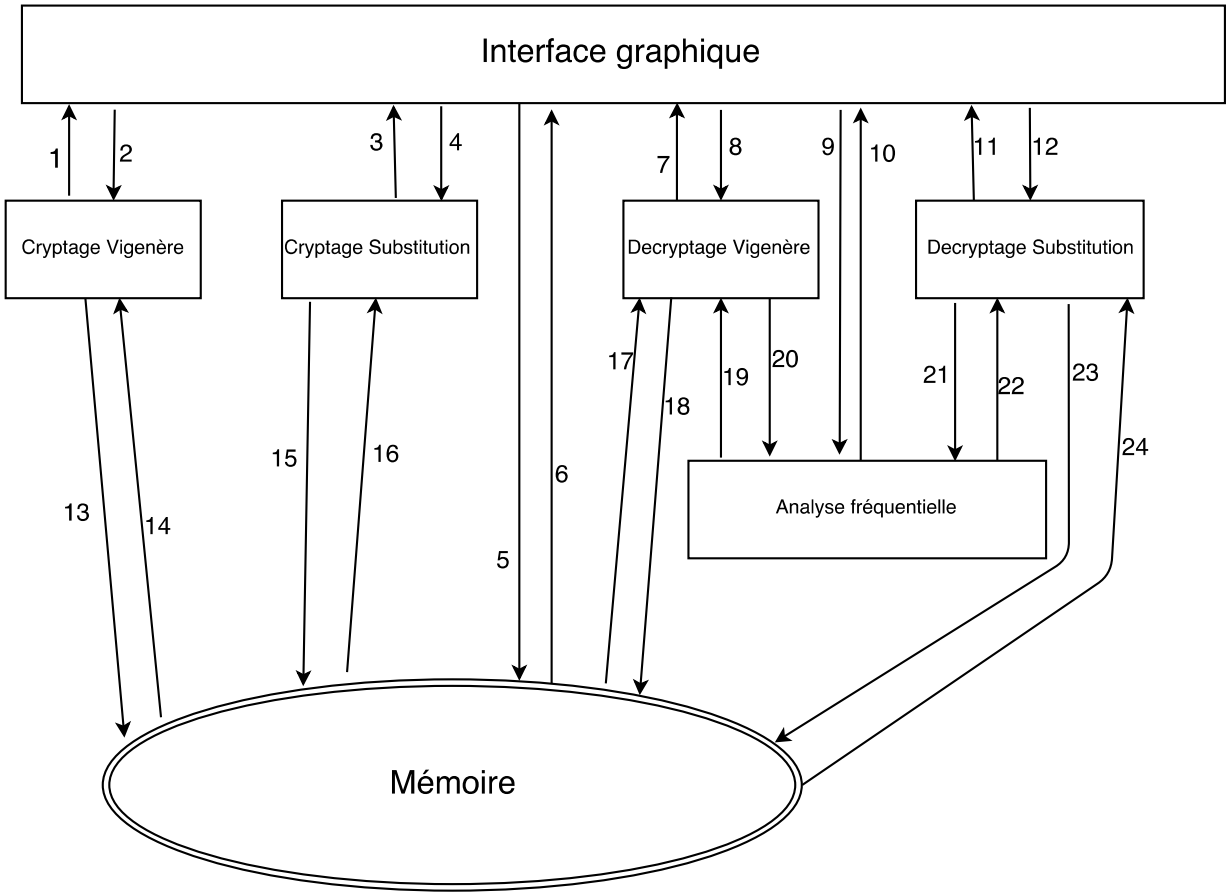
Decryptage Vigenère :
-decrypter le message

Cryptage Substitution :
-créer une clé aleatoirement
-crypter le message

Cryptage Vigenère :
-crypter le message

Decryptage Substitution :
-decrypter le message

Analyse fréquentielle :
-analyse frequentielle sur texte donné



1) Une chaine de caractère contenant le texte crypté
2) Une chaine de caractère contenant le texte claire
3) Une chaine de caractère contenant le texte crypté
4) Une chaine de caractère contenant le texte claire
5) Un nom de fichier
6) Fichier texte : clair, crypter, décrypter ou analyse fréquentielle
7) Une chaine de caractère contenant le texte crypté
8) Une chaine de caractère contenant le texte claire
9) Une chaine de caractère contenant le texte à analyser
10) Une structure de données contenant l'analyse fréquentielle
11) Une chaine de caractère contenant le texte crypté
12) Une chaine de caractère contenant le texte claire
13) Un nom de fichier
14) Fichier texte clair
15) Un nom de fichier
16) Fichier texte clair
17) Un fichier texte crypté
18) Un nom de fichier
19) Une structure de données contenant l'analyse fréquentielle
20) Une chaine de caractère contenant le texte à analyser
21) Une chaine de caractère contenant le texte à analyser
22) Un ensemble de données contenant l'analyse fréquentielle
23) Un nom de fichier
24) Un fichier texte crypté

Présentation des modules de l'organigramme.

Notre organigramme se décompose en six modules : un module d'interface graphique, un d'analyse fréquentielle, un de cryptage par substitution, un de décryptage par substitution,un de cryptage par la methode de vigenère et un de décryptage par méthode de vigenère.

Le module d'interface graphique est le centre de notre programme dans le sens où il est le seul qui voit ses fonctions appelées dans le main et qu'il intéragit avec tous les autres modules en appellant leur fonctions principales. Son rôle principal est de mettre à disposition de l'utilisateur une interface graphique (composée de menus, de boîtes de dialogue et de fenêtres). Ce module va également permettre à l'utilisateur de paramétrer les options du logiciel, il pourra choisir la langue du texte (français ou anglais) qu'il pourra ensuite faire crypter, décrypter ou soumettre à une analyse fréquentielle. Pour se faire, il aura la possibilité de charger un fichier texte ou de taper un texte ainsi que d'enregistrer le texte crypté ou décrypté, la clef de cryptage en fonction de la méthode utilisée ainsi qu'une analyse fréquentielle.

Le module d'analyse fréquentielle va, comme son nom l'indique, effectuer une analyse fréquentielle sur un texte donné. Il va donc remplir une structure nommée structure ANALYSE en observant le nombre d'occurrences de chaque caractère du texte ainsi que les nombres d'occurrences des digrammes et trigrammes du texte. Ce module sera utilisé de trois manières différentes : il pourra être appelé pour l'analyse d'un texte donné, pour permettre un décryptage par substitution ou pour permettre un décryptage par la méthode de vigenère.

Le module de cryptage par substitution va permettre de crypter un texte clair grâce à la méthode de la substitution tout en créant une clef de substitution aléatoire que l'utilisateur pourra sauvegarder via l'interface graphique.

Le module de décryptage par substitution va permettre de décrypter un texte crypté par la méthode de la substitution. Ce module permettra d'obtenir un texte partiellement décrypté ainsi que la clef partielle de substitution.

Le module de cryptage par la méthode de vigenère va permettre de crypter un texte clair grâce à la méthode de vigenère et d'une clef de cryptage rentrée par l'utilisateur.

Le module de décryptage par la méthode de vigenère va permettre de décrypter un texte crypté par la méthode de vigenère. Ce module permettra d'obtenir un texte décrypté ainsi que la clef utilisée lors du cryptage.

Présentation des interactions entre les différents modules de l'organigramme et de la mémoire.

L'interface graphique va intéragir avec la mémoire, elle va lui envoyer le nom de fichier entré par l'utilisateur afin de recevoir le fichier texte en clair ou crypté correspondant. L'interface pourra également lui envoyer à la mémoire le nom de fichier ainsi que le texte crypté ou décrypté afin de l'enregistrer (flèche 5 et 6 sur le schéma).

L'interface graphique va intéragir avec le module de l'analyse fréquentilelle, elle va lui envoyer une chaîne de caractère correspondant au texte à analyser entré par l'utilisateur. Le module d'analyse va lui renvoyer l'analyse fréquentielle sous la forme d'une structure 'analyse' contenant le nombre d'occurrences de chaque lettre, des différents digrammes et trigrammes que l'interface graphique pourra afficher (flèche 9 et 10 sur le schéma).

L'interface graphique va intéragir avec le module de cryptage par substitution, elle va lui envoyer une chaîne de caractères correspondant au texte en clair à crypter, et deux chaînes de caractères vides. Le module de cryptage par substitution va remplir les deux chaînes de caractères vides, une avec le texte crypté

obtenu à partir du texte clair et l'autre avec la clef de substitution correspondante générée de façon aléatoire. L'interface graphique pourra alors afficher ces deux chaînes de caractères (flèche 3 et 4 sur le schéma).

L'interface graphique va intéragir avec le module de décryptage par substitution, elle va lui envoyer une chaîne de caractères corespondant au texte crypté à décrypter, et deux chaînes de caractères vides. Le module de décryptage par substitution va remplir les deux chaînes de caractères vides, une avec le texte décrypté obtenu à partir du texte crypté et l'autre avec la clef de substitution correspondante. L'interface graphique pourra alors afficher ces deux chaînes de caractères (flèche 11 et 12 sur le schéma).

L'interface graphique va intéragir avec le module de cryptage par la méthode de vigenère, elle va lui envoyer une chaîne de caractères corespondant au texte en clair à crypter,une chaîne de caractères contenant la clef choisie par l'utilisateur et une chaîne de caractères vides. Le module de cryptage par la méthode de vigenère va remplir la chaîne de caractères vides avec le texte crypté obtenu à partir du texte clair et de la clef correspondante choisie par l'utilisateur. L'interface graphique pourra alors afficher cette chaîne de caractères (flèche 1 et 2 sur le schéma).

L'interface graphique va intéragir avec le module de décryptage par la méthode de vigenère, elle va lui envoyer une chaîne de caractères correspondant au texte crypté à décrypter, et deux chaînes de caractères vides. Le module de décryptage par la méthode de vigenère va remplir les deux chaînes de caractères vides, une avec le texte décrypté obtenu à partir du texte crypté et l'autre avec la clef de vigenère correspondante. L'interface graphique pourra alors afficher ces deux chaînes de caractères (flèche 7 et 8 sur le schéma).

Le module de l'analyse fréquentilelle va intéragir avec le module de décryptage par substitution, le module de décryptage par substitution va lui envoyer une chaîne de caractères corespondant au texte crypté à analyser. Le module d'analyse va alors lui renvoyer l'analyse fréquentielle de ce texte sous la forme d'une structure 'analyse' contenant le nombre d'occurrences de chaque lettre, des différents digrammes et trigrammes que le module de décryptage utilisera pour décrypter le texte (flèche 15 et 16 sur le schéma).

Le module de l'analyse fréquentilelle va intéragir avec le module de décryptage par la methode de vigenère, le module de décryptage va envoyer à celui d'analyse une chaîne de caractères correspondant au texte crypté à analyser ainsi qu'un entier kasiski qui permettra de conjecturer la taille de la clef. Le module d'analyse va alors lui renvoyer l'analyse fréquentielle de ce texte sous la forme d'une structure 'analyse' contenant le nombre d'occurrences de chaque lettre, des différents digrammes et trigrammes que le module de décryptage utilisera pour décrypter le texte (flèche 13 et 14 sur le schéma).

4 Language choisi et explications

Le développement de l'application s'est fait en langage C pour les raisons suivantes :

Nous avions besoin d'une application fonctionnelle et donc d'un langage procédural. Le meilleur langage sur le marché étant le C, le choix s'est logiquement porté sur ce dernier.

Aussi, un argument de choix est celui de la portabilité. En effet, nous avions besoin d'une portabilité sur plusieurs environnements et donc d'un besoin de standard ou norme. D'où le langage C, car en effet il est possible d'utiliser le même programme sur tout autre système (autre hardware, autre système d'exploitation), simplement en le recompilant.

La bibliothèque graphique que nous avons décidée d'utiliser avec ce langage est GTK+ parce qu'elle permet d'implémenter des boutons, des zones de texte, des menus ou encore du traitement de fichier. Elle nous semblait donc la plus adaptée au dévelopement de notre application.

5 Partie technique

Tout d'abord, l'application fonctionne (compilation et exécution) et l'ensemble des fonctionnnalités demandées par le client ont été réalisées.
Les commandes pour compiler et exécuter sont les suivantes :
-compilation : make all
-execution :make run

Nous souhaitions avoir une application complète qui puisse tourner sur des ordinateurs de tout système d'exploitation et ce, sans pour autant qu'il ne consomme trop de mémoire ou de temps processeur.

L'équipe de développement a évidemment rencontré bon nombre de problèmes ou bugs. Voici pour chaque module une liste non exhaustive :

5.1 Interface graphique

Nous avons eu plusieurs problèmes liées à GTK+ :
-Avec GTK+ on ne peut passer qu'un seul argument dans un bouton, pour remédier à ce problème on a décidé de passer en argument une structure ou alors d'enregistrer les informations dans un fichier pour pouvoir les utiliser plus tard.
-On a rencontré aussi une erreur avec un switch et la solution qu'on a trouvée à été de faire la même chose avec un if et le problème était résolu.

-On a aussi un problème non résolu, lorsque l'on sauvegarde, le résultat ou la clef, une 2ème fois rapidement le programme crash avec une erreur mémoire et ce n'est pas encore résolu.

5.2 Analyse fréquentielle

-probléme de segfault du à la gestion de la mémoire : remplace "gchar* exemple" par "gchar exemple[10]"
-PROBLEME AVEC STRCPY REMPLACE PAR BOUCLE

5.3 Décryptage Vigenère

Le décryptage fonctionne en deux parties :
avec recherche automatique : test de Kasiski (marche 1 fois sur 3, taux de reussite plus important pour des petites clés et peut éventuellement l'afficher en double (ex : clecle). *S'est effectué lors du premier clic gauche de l'utilisateur.

Avec itération manuelle pour trouver la taille de la clé car le test de Kasiski peut se tromper. *Disponible après le premier clic gauche. => Permet d'augmenter considérablement le taux de réussite de Cryptanalyse d'un texte.

Problèmes rencontrés avec Kasiski et l'hypothèse faite sur la taille de la clé.

5.4 Décryptage substitution

-Nous avons sous-estimé le nombre de lignes attribuées au module de cryptage de substitution. Ce problème, couplé aux nombreux changements à apporter aux structures, tableaux et différentes variables à

chaque fois que nous décryptions une lettre du texte crypté, a vite rendu la fonction prévue dans le cahier de spécifications illisible et difficile à corriger et à paramètrer. Ainsi pour permettre une meilleure lecture du code, nous avons divisés cette fonction en plusieurs sous-fonctions.

On notera également que le décryptage ne pourra presque jamais donner un résultat parfait, certaines lettres resteront cryptées et d'autres seront décryptées de la mauvaise manière.

5.5 Spécifications

Des approximations et des erreurs de jugement sur le cahier des spécifications ont conduit à certains changements ou problèmes :
-changement de la structure (rajout de champs et de fonctions ainsi que la division de certaines fonctions en plusieurs petites fonctions).

-

-

Le découpage technique des modules s'est fait en fonction des fonctionnalités, chacune de ces dernières correspondant à un module.

5.6 Tests (unitaires)

L'exactitude des résultats de l'analyse étant tres importante pour les clients, nous avons prédéfinis des tests unitaires pour chaque module (ou plutôt les fonctions le composant). Ainsi, à chaque étape de la conception du logiciel nous avons pu vérifier que le calcul n'était pas altèré.

Les tests permettent de voir comment le logiciel réagit lors de futures améliorations ou d'une éventuelle maintenance.

Nous avons décidé d'utiliser Cunit qui est un système pour l'écriture, l'administration et l'exécution de tests unitaires en langage C. Il fournit à son programmeur une fonctionnalité de test de base avec une variété d'interfaces utilisateur flexibles.
Un "test" CUnit est une fonction C ayant comme signature :
void_test_func()
Le corps de la fonction sera lui composé d'assertions Cunit.
Il y aura création d'un registre et d'une suite de tests unitaires.
Nous noterons aussi la création automatique d'un Summary(récapitulatif) des tests.
remarque : besoin de l'option -lcunit à rajouter à la compilation des tests et d'inclure la bibliotheque avec un #include <CUnit/CUnit.h>.

Les résultats des tests unitaires sont les suivants :

```
Run Summary:    Type  Total    Ran Passed Failed Inactive
              suites      1      1    n/a      0        0
               tests     11     11     11      0        0
             asserts     22     22     22      0      n/a
```

METTRE COMMANDE POUR COMPILER ET CELLE POUR EXECUTER TESTS

6 Organisation interne et affectation des tâches

L'organisation de l'équipe est une chose importante pour le bon fonctionnement du projet et le déroulement du codage, et est faite selon nos capacités et nos disponibilités.

Module	Personne(s)
Décryptage Vigenere	Chouipe et Alabi
Décryptage Substitution	Lienhardt et Alabi
Cryptage Vigenere	El harti et Chouipe
Cryptage Substitution	El harti et Lienhardt
Analyse Fréquentielle	El harti
Interface Graphique	Capdenat et Benyamna

EXPLICATIONS SUR TABLEAU DES TACHES

L'organisation étant bonne et la cohésion du groupe certaine, la répartition et la mise en place d'un planning s'est faite naturellement et a été respectée.
La communication étant la clé d'un projet en groupe, nous avons préféré nous réunir très régulièrement et travailler tous ensemble et ce quel que soit l'étape du projet.

6.1 Planning de développement

La phase de développement constitue l'étape critique du projet, avec d'une part la décision de coder l'interface graphique et d'autre part l'intégration de tous les modules séparement. Lors de cette phase, les modules ont connu leurs dernières évolutions ou plutôt ajustements.
Avant de tester l'ensemble de l'application, nous avons dans un premier temps codé et testé chaque fonction ou plutôt module pour savoir si elles fonctionnaient séparément.
Nous les avons ensuite réunies en les assemblant étapes par étapes pour construire l'application finale.
–> en priorité : l'interface graphique consistait la base de l'application et devait donc être commencée et avancée tres rapidement
En évaluant les connaissances de chacun et en faisant un point régulièrement sur nos tâches respectives, cela a permis d'avancer efficacement dans la réalisation du projet.

7 comparaison lignes de code

Module	Nombre de lignes de code (estimation)	Nombre de lignes de code (avéré)
Décryptage Vigenere	200	210
Décryptage Substitution	150	380
Cryptage Vigenere	50	25
Cryptage Substitution	50	65
Analyse Fréquentielle	50	110
Interface Graphique	1000	1300
Total	1500	

Certains modules ont depassé le nombre de lignes estimées selon les raisons suivantes :
-décryptage Vigenère : nombre de lignes trop juste au vu de certaines fonctions qui permettent de vérifier nos résultats, par exemple la taille du mot clé.
-cryptage substitution : il faut une fonction(tirage) qui va permettre qu'une lettre est toujours chiffrée par une seule et même lettre.
-analyse fréquentielle : fonction AnalyseFreq en plus qui est une tres légère variante de l'analyse des occurences de chaque lettre faite déjà dans AnalyseFrequentielle. Aussi, dans cette dernière, l'analyse des trigrammes fonctionne exactement comme celle des digrammes et donc on a répété ces "autres" 40 lignes
-interface graphique : pk??

8 Conclusion

-des choix à changer
->au niveau des spécifications et des fonctions

-critique du projet
->Manque de réfléxion ou d'approfondissement de certaines notions à des moments
->??????

-si à refaire, pareil??
->Le groupe serait le même et la base du projet aussi.

Finalement, nous avons une version "1.0" de l'application. La majorité des fonctionnalités de base ont été implémentées et fonctionnent correctement mais il existe quelques améliorations qui pourraient véritablement

aboutir à une version "2.0" intéressante.
Quelques exemples d'améliorations pourraient être ajoutées : -Vigenère : ajout d'informations supplémentaires qui permettent d'aider/améliorer le décryptage (exemple : on connait déja la taille de la clé). -permettre à l'utilisateur de rentrer un type de texte (poéme, roman..) pour aider au décryptage
-plus de langues disponibles
-plus de cryptages/décryptages différents
=> Rajout d'un bouton "Recherche Intelligente" qui tente de trouver du premier coup la taille de clé. -

Ce projet d'une durée d'un semestre (et plus précisement de 16 semaines) est une vraie expérience. Cela a été enrichissant d'un point de vue personnel et collectif : il nous a apporté beaucoup, tant au niveau technique qu'en terme de gestion de projet.

Nous connaissons maintenant l'importance du découpage d'un projet en plusieurs étapes : cahier des charges, spécifications, codage.. C'est un projet en "conditions d'entreprise" qui nous servira dans le futur. C'est la première fois que nous travaillons avec un groupe aussi nombreux sur un projet avec des caractéristiques bien définies.

Nous sommes globalement satisfaits de ce que nous avons réalisé. Au niveau de la gestion du projet en équipe, nous avons réussi à bien nous répartir les tâches afin de réaliser nos objectifs dans les temps et l'ambiance générale du groupe était très bonne.