

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 comment les exigences fonctionnelles vont être implémentées (cahier des spécifications), 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 decryptage 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 decrypter un texte ou fichier et ce par méthode de Substitution ou Vigenère. Mais aussi, il lui est 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 Français (decryptage)
- bouton Anglais (decryptage)
- affichage pour l'analyse fréquentielle
- charger un fichier texte
- sauvegarder un fichier texte
- créer un nouveau fichier texte (résultats)
- Demander clef de Vigenère

Decryptage Vigenère :

- decrypter le message

Cryptage Substitution :

- créer une clé aléatoirement
- crypter le message

Cryptage Vigenère :

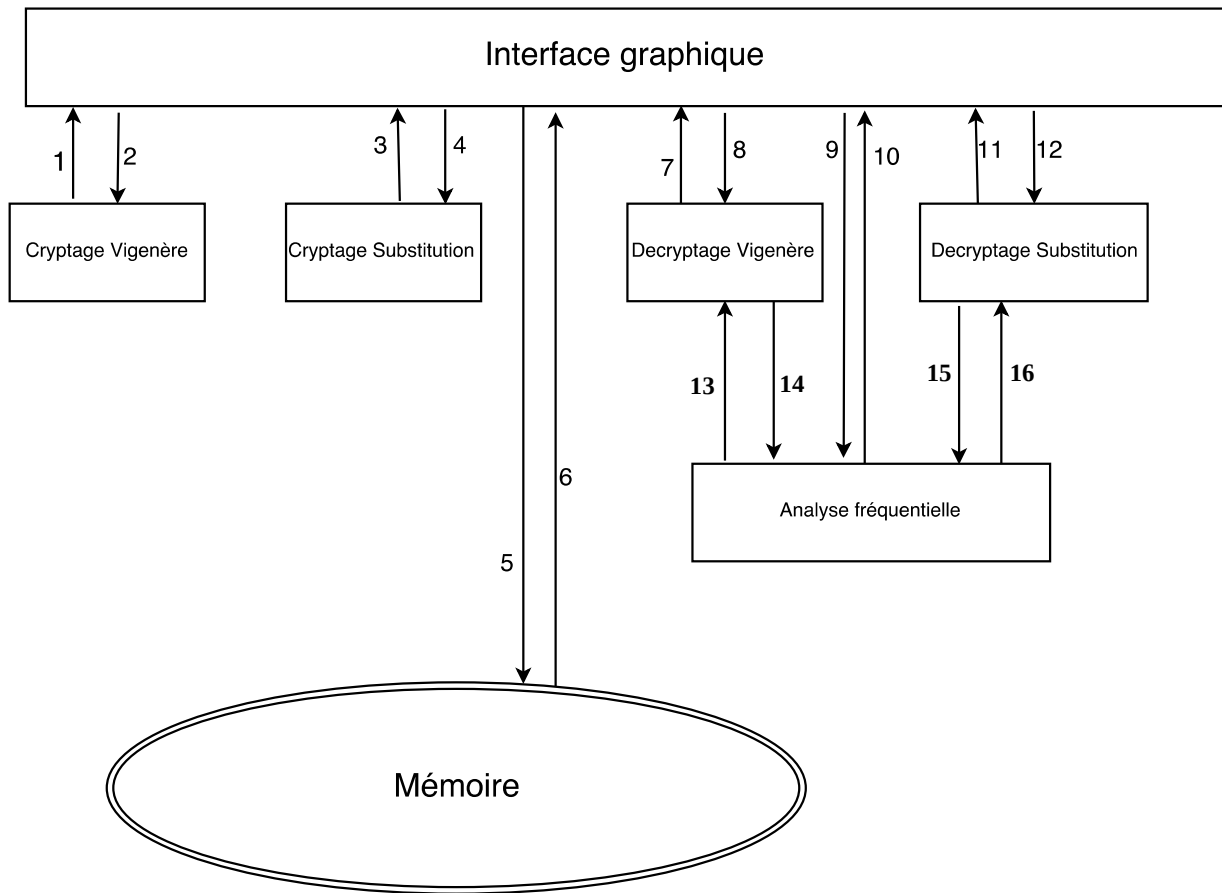
- crypter le message

Decryptage Substitution :

- decrypter le message

Analyse fréquentielle :

- analyse fréquentielle sur texte donné



PRESENTATION DES MODULES DE L'ORGANIGRAMME.

Notre organigramme se decompose en six modules : un module d'interface graphique, un d'analyse frequentielle, un de cryptage par substitution, un de decryptage par substitution, un de cryptage par la methode de vigenere et un de decryptage par methode de vigenere.

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 interagit avec tous les autres modules en appelant leur fonctions principales. Son rôle principale est de metre à disposition de l'utilisateur une interface graphique (composé de menus, de boites dialogue et de fenêtres). Ce module va egalement 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, decrypter 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 methode utilisée ainsi qu'une analyse frequentielle.

Le module d'analyse frequentielle va, comme son nom l'indique, effectuer une analyse frequentielle sur un texte donné. Il va donc remplir une structure nommé 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 decryptage par substitution ou pour permettre un decryptage par la methode de vigenere.

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

Le module de decryptage par substitution va permettre de decrypter un texte crypter par la methode 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 methode de vigenere va permettre de crypter un texte clair grace à la methode de vigenere et d'une clef de cryptage rentrée par l'utilisateur.

Le module de decryptage par la methode de vigenere va permettre de decrypter un texte crypté par la methode de vigenere. Ce module permettra d'obtenir un texte décrypté ainsi que la clef utilisée lors du cryptage.

PRESENTATION DES INTERACTIONS ENTRE LES DIFFERENTS MODULES DE L'ORGANIGRAMME ET DE LA MEMOIRE.

L'interface graphique va interagir 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 memoire le nom de fichier ainsi que le texte crypté ou décrypté afin de l'enregistrer (fleche 5 et 6 sur le schema).

L'interface graphique va interagir avec le module de l'analyse frequentielle, elle va lui envoyer une chaine de caractère correspondant au texte à analyser entré par l'utilisateur. Le module d'analyse va lui renvoyer l'analyse frequentielle sous la forme d'une structure analyse contenant le nombre d'occurrences de chaque lettre, des differents digrammes et trigrammes que l'interface graphique pourra afficher (fleche 9 et 10 sur le schema).

L'interface graphique va interagir avec le module de cryptage par substitution, elle va lui envoyer une chaine de caractères correspondant au texte en clair à crypter, et deux chaines de caractères vides. Le module de cryptage par substitution va remplir les deux chaines 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 interagir avec le module de décryptage par substitution, 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 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 interagir avec le module de cryptage par la méthode de Vigenère, elle va lui envoyer une chaîne de caractères correspondant 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 interagir 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 fréquentielle va interagir avec le module de décryptage par substitution, le module de décryptage par substitution va lui envoyer une chaîne de caractères correspondant 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 fréquentielle va interagir avec le module de décryptage par la méthode 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 et les raisons étant les suivantes :

Nous avons 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 avons 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 développement de notre application.

5 Partie technique

Tout d'abord, l'application fonctionne (compilation et execution) et l'ensemble des fonctionnalités demandées par le client ont été réalisées.

Les commandes pour compiler et executer sont les suivantes :

-compilation :

-execution :

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

L'equipe de developpement a evidemment rencontré bon nombre de problèmes ou bugs. Voici pour chaque module une liste non exhaustive :

5.1 Interface graphique

-probleme du switch, remplacé par if..

-

5.2 Analyse frequentielle

-probleme de segfault du a la gestion de la memoire : remplace "gchar* exemple" par "gchar exemple[10]"

-

5.3 Decryptage vigenère

Le decryptage 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 eventuellement l'afficher en double (ex : clecle). *S'est effectué lors du premier clic gauche de l'utilisateur.

Avec itération manuel 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 Decryptage substitution

-Nous avons sous-estimé le nombre de lignes attribués au module de cryptage de substitution, ce problème couplé au nombreux changements à apporter aux structures, tableaux et différentes variables à chaque fois que nous decryptons une lettre du texte crypté a vite rendu la fonction prévue dans le cahier de spécification illisible et difficile à corriger et paramétrer. Ainsi pour permettre une meilleure lecture du code nous avons divisé cette fonction en plusieurs sous fonctions.

5.5 Specifications

Des approximations et des erreurs de jugement sur le cahier des specifications ont conduit a certains changements ou problemes :

- changement de la structure (rajout de champs et de fonctions ainsi que la division de certaines fonctions en plusieurs petites fonctions).

-
-

La decoupe technique des modules s'est faite en fonction des fonctionnalités. Chacune de ces dernieres correspondant a un module.

5.6 Tests (unitaires)

L'exactitude des resultats de l'analyse etant tres importante pour les clients, nous avons predefinis des tests unitaires pour chaque module(ou plutot les fonctions le composant). Ainsi, a chaque etape de la conception du logiciel nous avons pu verifier que le calcul n'etait pas altere.

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

Nous avons décidé d'utiliser Cunit qui est un systeme pour l'écriture, l'administration et l'execution de tests unitaires en langage C. Il fournit a son programmeur une fonctionnalité de test de base avec une variété d'interfaces utilisateur flexible.

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 a rajouter a la compilation des tests et d'inclure la bibliotheque avec un `#include <CUnit/CUnit.h>`.

Les resultats des tests unitaires sont les suivants :

METTRE SCREEN TESTS "RUN" ICI

METTRE COMMANDE POUR COMPILER ET CELLE POUR EXECUTER TESTS

6 Organisation interne et affectation des taches

L'organisation de l'équipe est une chose importante pour le bon fonctionnement du projet et le deroulement 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 repartition 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 developpement

La phase de developpement constitue l'étape critique du projet, avec d'une part la decision de coder l'interface graphique et d'autre part l'integration de tous les modules séparément. Lors de cette phase, les modules ont connu leurs dernieres evolutions ou plutot ajustements.

Avant de tester l'ensemble de l'application, nous avons dans un premier temps codé et testé chaque fonction ou plutot 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 très rapidement

En évaluant les connaissances de chacun et en faisant un point régulièrement sur nos taches respectives, cela a permis d'avancer efficacement dans la réalisation le projet.

7 comparaison lignes de code

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

Certains modules ont dépassés le nombre de lignes estimées selon les raisons suivantes :

-decryptage Vigenere : nombre de ligne 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 meme lettre.

-analyse frequentielle : fonction AnalyseFreq en plus qui est une tres legere variante de l'analyse des occurences de chaque lettre faite deja dans AnalyseFrequentielle. Aussi, dans cette derniere, l'analyse des trigrammes fonctionne exactement comme celle des digrammes et donc on a repeté ces "autres" 40 lignes

-interface graphique : pk??

8 Conclusion

-des choix a changer

->au niveau des specifications et des fonctions

-critique du projet

->Manque de reflexion ou d'approfondissement de certaines notions a des moments

->??????

-si a refaire, pareil??

->Le groupe serait le meme 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 aboutir

véritablement à une version "2.0" vraiment intéressante.

Quelques améliorations pourraient être ajoutées : -Vigenere : ajout d'informations supplémentaire qui permettent d'aider/améliorer le decryptage (exemple : on connaît déjà la taille de la clé). -permettre à l'utilisateur de rentrer un type de texte (poème, roman..) pour aider au decryptage

-plus de langues disponibles

-plus de cryptages/decryptages 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 précisément de 16 semaines) est une vraie expérience. Cela a été enrichissant d'un point de vue personnel ou 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 en groupe à 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.