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 a respecter, et après avoir decrit comment les exigences fonctionnelles vont être implémentées(cahier des specifications) , 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 egalement fournit pour expliquer le fonctionnement de l'application.

Dans ce rapport, il s'agira de s'interesser 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 developper 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 methode de Substitution ou Vigenère. Mais aussi, il lui est possible d'effectuer independamment une analyse frequentielle 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

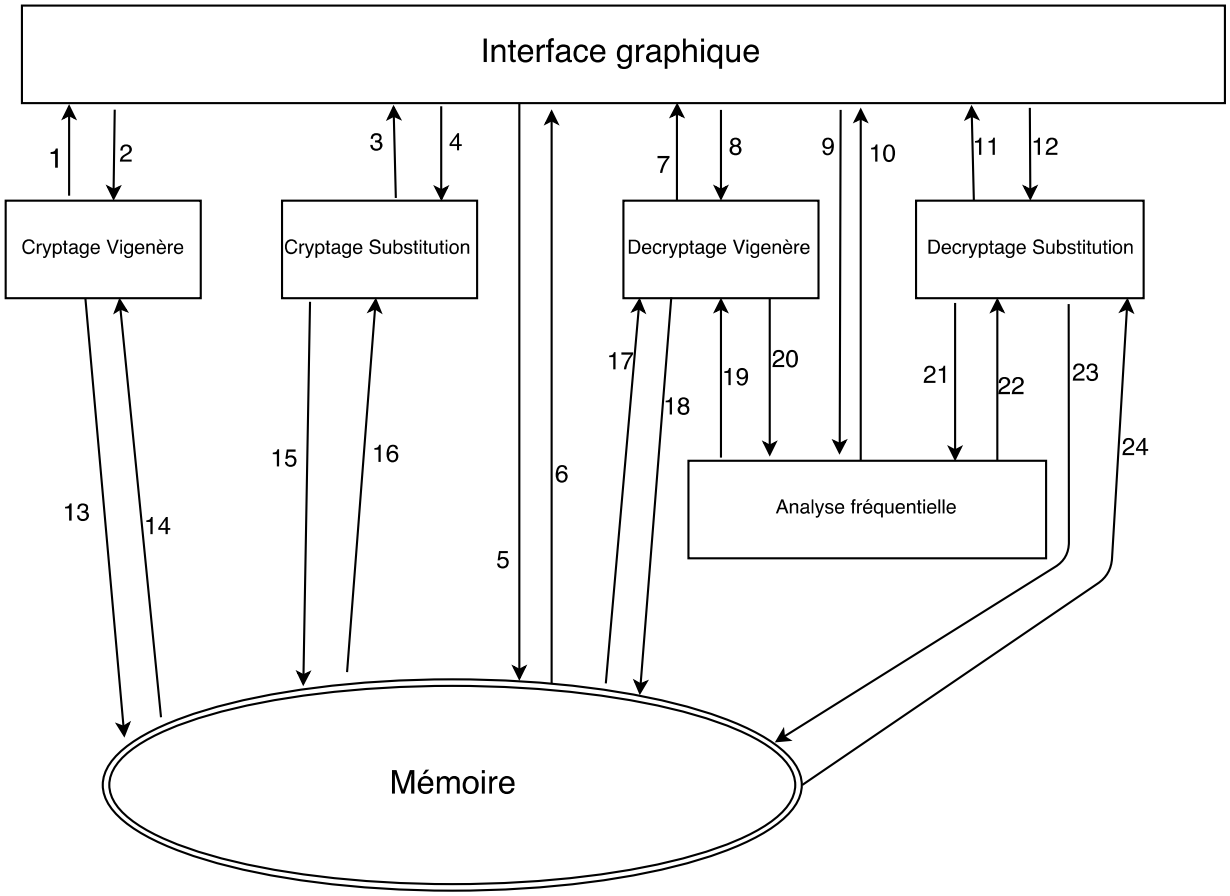
Decryptage Vigenère :
-decrypter le message

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

EXPLICATION DE L'ORGANIGRAMME ICI

4 Language choisi et explications

Le developpement de l'application s'est fait en langage C et les raisons étant les suivantes :

Nous avions besoin d'une application fonctionnelle et donc d'un langage procédurale. Le meilleur langage sur le marché etant 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'ou 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 developement de notre application.

5 Partie technique

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

Nous souhaitions 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 (younes nicolas)
-

5.2 Analyse frequentielle

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

5.3 Decryptage vigenère

Itération manuel pour trouver la taille de la clé car le test de Kasiski se trompe souvent. Problème rencontrées avec Kasiski et l'hypothèse fait sur la taille de la clé.
Avec itération automatique : marche 1 fois sur 2 pour petite clé et peut lafficher en double (ex : clecle).
solution => itération manuel pour la taille.

5.4 Decryptage substitution

-

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)
-
-

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'ecriture, 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 etant bonne et la cohesion 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 reunir très regulierement et travailler tous ensemble et ce quel que soit l'etape du projet.

6.1 Planning de developpement

La phase de developpement constitue l'etape critique du projet, avec d'une part la decision de coder l'interface graphique et d'autre part l'integration de tous les modules séparement. 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 tres rapidement
En evaluant les connaissances de chacun et en faisant un point regulierement 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	?
Cryptage Vigenere	50	25
Cryptage Substitution	50	65
Analyse Fréquentielle	50	110
Interface Graphique	1000	?
Total	1500	

Certains modules ont depassés 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/ameliorer le decryptage (exemple : on connait déja la taille de la clé). -permettre a l'utilisateur de rentrer un type de texte (poeme, roman..) pour aider au decryptage
-plus de langues disponibles
-plus de cryptages/decryptages differents
=> 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 precisement de 16 semaines) est une vrai 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 decoupage d'un projet en plusieurs etapes : cahier des charges, specifications,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 a aussi nombreux sur un projet avec des caracteristiques 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.