

**PROJET CRYPTOGRAPHIE**

**CONCEPTION ET IMPLEMENTATION D’UNE VERSION SIMPLIFIE DE L’ALGORITHME DE CRYPTAGE EGNIMA**

**Réalisé par :** DIMO NGUEMBOU IDRISS GUERIN

NGONO MANGA GHISLAIN GISCARD

MACHIELSEN SIMON

SOUFFO ROSTAND

**Sous la supervision de** : Mr DERASSE ERIC

**2020 - 2021**

## **OBJECTIF**

L’objectif du projet est de concevoir un programme qui simule la version simplifiée de la machine Enigma.

Le programme respecte l’API de base en couche.

La modélisation des rotors est réalisée par des tableaux/listes qui correspondent aux décalages provoqués par les liaisons électriques avec un réglage initial par défaut immuable : par exemple, chaque « a » devient « e », chaque « b » devient « k », chaque « c » devient « m » …

Les rotors sont toujours les mêmes, mais peuvent être placés différemment selon l’initiation. La version des rotors et du réflecteur serons partagés dans un fichier « rotors.init » au format JSON du style suivant :

Graphical user interface

Description automatically generated with low confidence

Le message à coder est lu à partir d’un simple fichier texte. L’on doit formater le message en majuscule, sans espace et sans accent. A chaque lettre, le premier rotor tourne ce qui signifie que deux lettres ne serons jamais codées de la même manière. Dans cette version simplifiée, le codage démarre toujours avec les rotors sur leur position initiale.

Table

Description automatically generated

## **STRATEGIE DE RESOLUTION**

Pour l’implémentation de cette solution, nous allons dans un premier temps analysé le besoin à travers en élaborant un diagramme de flux fonctionnel qui nous permettra d’expliquer le fonctionnement de notre implémentation. Ensuite nous élaborons une interface de programmation décrivant chaque fonction implémentée et enfin nous présenterons quelques exemples de tests

### **DIAGRAMME DE FLUX FONCTIONNEL**

#### **Diagramme de Contexte (diagramme de flux niveau 0)**

Diagram

Description automatically generated

#### **Diagramme de Flux Cryptage et Décryptage**

Diagram, schematic

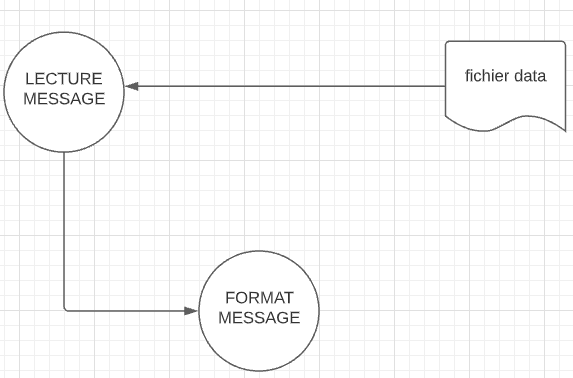
Description automatically generated

#### **Diagramme de Flux Ecriture message**

Diagram

Description automatically generated

#### **Diagramme de flux lecture message**



### **INTERFACE DE PROGRAMMATION**

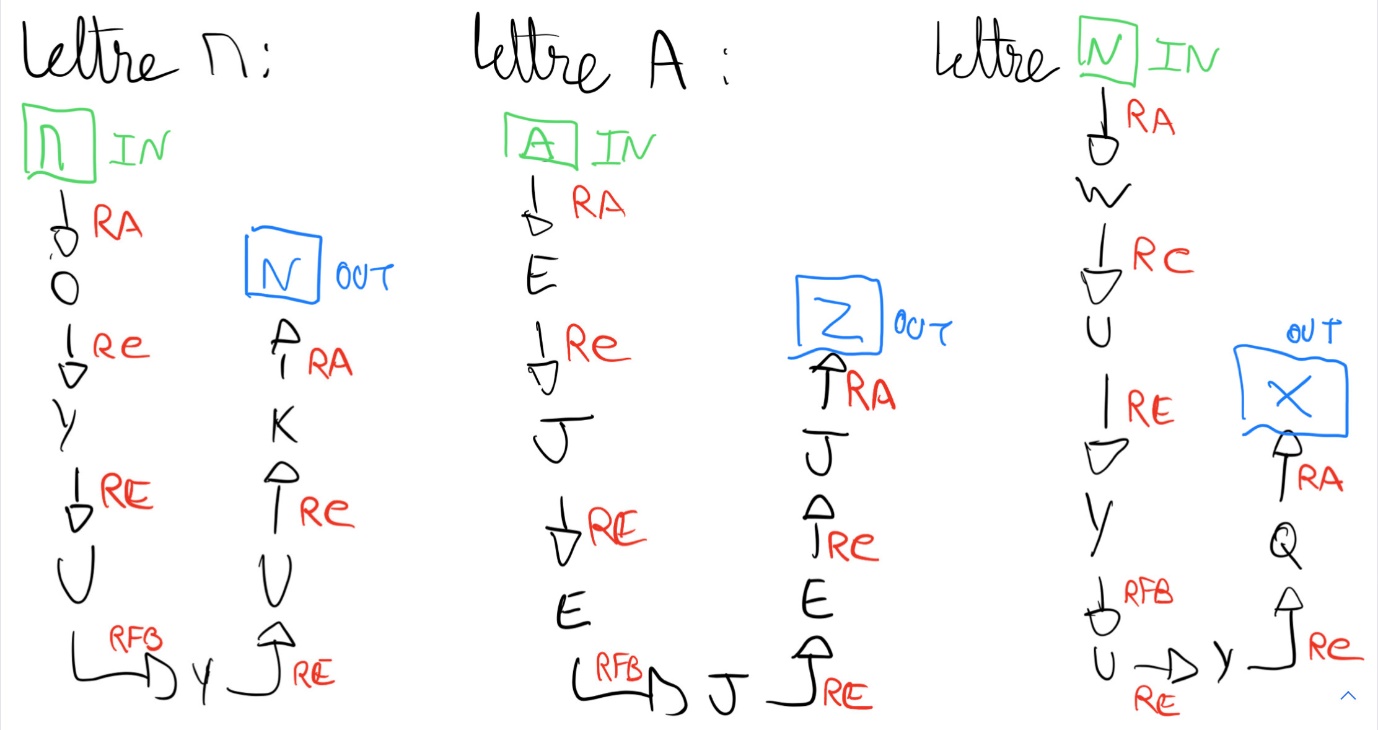
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| COUCHES | FONCTIONS | PARAMETRES | SORTIES | DESCRIPTION |
| INTERFACE | Checkinput() | None | List(string(msg), tuple(liste\_rotors) | Contrôle les entrées de l’utilisateur (depuis une console et retourne le message entré et la liste des rotors à utiliser pour crypter ou décrypter |
| Faire\_choix | String(message) | Entier(choix) | Prend en paramètre le message à afficher à l’utilisateur, récupère le choix de l’utilisateur, valide et renvoi son choix |
| Menu | None | None | Affiche les différents menus de la fonction et appel les différentes fonctions gérant ce menu |
| LOGIQUE | Cryptage() / decryptage() | String(message), tuple(liste\_rotors), Booleen(crypt\_rot) | String(msg\_cp) | Cryptage / decryptage d’un message, le dernier parametre nous dis si nous somme dans un cryptage ou décryptage |
| Rotor() | Char(item), string(string\_equiv), Booleen(crypt\_rot) | Char(item\_corresp) | Renvoi le caractère crypter/décrypter correspondant au caractère entré |
| Reflecteur() | Char(item), string(string\_equiv), Booleen(crypt\_rot) | Char(item\_corresp) | Renvoi le caractère correspondant au caractère entré |
|  | formatString | String(msg) | String(msg) | Récupère un message passé en paramètre la formate et retourne le message formaté |
| DONNEES | SauvegardeMessage | Message  Nom\_du\_fichier | Booléen | Écrire le message dans un ficher |
| LectureMessage | String(source\_file) | String(msg) | Récupère le chemin vers le fichier, lis le fichier, et retourne le contenu du fichier formaté |

### **TESTS**

Afin de s’assurer que notre code repond à nos besoins, nous avons effectué un ensemble tests(listé dans le tableau ci-dessous) . parvenu à ces resultats, nous pouvons dire que le code fonctionne correctement.

Le test de cryptage et décryptage effectué ci-dessous à été fait au préalable manuellement (image ci-dessous) . reprenant ces memes paramettres et en l’éxécutant dans code, nous obtenons les memes resultats

|  |  |  |  |
| --- | --- | --- | --- |
| **Actions ou opérations à vérifier** | **Paramètres d’entrées** | **Tests effectués** | **Résultats attendus** |
| **Checkinput()** | | | |
| Contrôle de la syntaxe et de la cohérence des paramètres d’entrés | ‘mon message ici’ RC RD RE RFB | Result = checkInput()  Si result[0] == ‘mon message ici’ et result[1] == [‘RC’, ‘RD’, ‘RE’] | TRUE |
| Contrôle de la syntaxe des paramètres en mélangeant l’ordre de paramètre | ‘mon message ici’ RC RD RE RFB |  | Demande à l’utilisateur de ressaisi les paramètres |
| Contrôle en diminuant le nombre de paramettre | ‘mon message ici’ RC RE RFB |  | Demande à l’utilisateur de ressaisi les paramètres |
| Contrôle en augmentant le nombre de paramèttre | ‘mon message ici’ RC RD RA RE RFB |  | Demande à l’utilisateur de ressaisi les paramètres |
| Contrôle de la syntaxe retirant une cote sur le message | ‘mon message ici RC RD RE RFB |  | Demande à l’utilisateur de ressaisi les paramètres |
| **Cryptage() / decryptage()** | | | |
| Contrôle de la fonctionnalité pour un cryptage | ‘MAMAN’, [‘RA’, ‘RC’, ‘RE’, ‘RFB’], True | Result = cryptage()  Si cryptage == ‘NZNZX’ | TRUE |
| Contrôle de la fonctionnalité pour un décryptage | ‘NZNZX’, [‘RA’, ‘RC’, ‘RE’, ‘RFB’], False | Result = cryptage()  Si cryptage == ‘MAMAN’ | TRUE |
| Contrôle de la fonctionnalité pour un décryptage | ‘ M A ’, [‘RA’, ‘RC’, ‘RE’, ‘RFB’], True | Result = cryptage()  Si cryptage == ‘ N Z’ | TRUE |



### **PRESENTATION DE LA SOLUTION DEVELOPPEE**

### **Présentation du processus de cryptage et décryptage**

#### **Cryptage**

* Ici l’utilisateur fait son choix

Graphical user interface, text, application

Description automatically generated

* L’utilisateur fait le choix 3 pour le cryptage ou décryptage du message

Graphical user interface, text, application, email

Description automatically generated

* 3- l’utilisateur fais le choix 1 pour crypter un message et rentre les paramètres (messages rotors et réflecteurs)

Graphical user interface, text, application, email

Description automatically generated

#### **Décryptage**

* L’utilisateur fait le choix 2 pour le décryptage

Graphical user interface, text, application

Description automatically generated

* l’utilisateur entre le message à décrypter ainsi que les configurations de décryptage

Graphical user interface, text, application, email

Description automatically generated

# **CONCLUSION**

Après la guerre, les cryptographes allemands furent interrogés par des spécialistes de la mission TICOM. Les interrogateurs constatèrent que leurs prisonniers savaient qu'Enigma n'était pas incassable et qu'il était possible de déchiffrer certains messages. Alan Turing, qui aura largement contribué à « casser » Enigma, est aussi considéré comme un des pères de l’ordinateur moderne. Aujourd’hui, la machine Enigma ne résisterait pas à une « attaque » par la « force brute » de nos ordinateurs, c’est-à-dire en essayant toutes les combinaisons possibles. Côté dénombrement, nous allons chercher à estimer le temps qu’il faudrait dans les années 1930 à un ordinateur pour « casser » Enigma par la « force brute », c’est-à-dire en essayant toutes ces combinaisons.

# **BIOGRAPHIE**

<http://wiki.franklinheath.co.uk>

<http://users.telenet.be/d.rijmenants/en/enigmamenu.htm>

<https://interstices.info/jcms/int_70884/turing-a-l-assaut-d-enigma>

<https://simonsingh.net/cryptography/enigma-photos/>

**IMITATION GAME" :** Fiche du film : <http://www.allocine.fr/film/fichefilm_gen_cfilm=198371.html>

Vous trouverez un simulateur sur :Enigma machine / Tom MacWright / Observable (observablehq.com)

Sur Wikipédia : <https://fr.wikipedia.org/wiki/Enigma_(machine)>

Un travail intéressant : <https://www.math.ch/about-sms/matura-awards/johansson.pdf>