Table des matières

[I. Introduction 2](#_Toc408328931)

[II. Méthode utilisée 3](#_Toc408328932)

[III. Algorithme 4](#_Toc408328933)

[A. Matrice 4](#_Toc408328934)

[B. Fichier 4](#_Toc408328935)

[IV. Fonctionnement 5](#_Toc408328936)

[V. Problèmes rencontrés 7](#_Toc408328937)

[VI. Conclusion 8](#_Toc408328938)

# Introduction

Pour le projet sécurité informatique, il y a un sujet parmi quatre à choisir:

* Projet codage/décodage de fichiers
* Wifi: ecrire un script sur 1 Fake AP
* Projet écoute: ettercap & Drifnet
* Attaque Matasploit sur Kali

Le sujet choisit est le suivant, outil de codage linéaire. Il s'agit de sélectionner un fichier et de choisir s'il doit être codé ou décodé.

# Méthode utilisée

Pour faire le chiffrement, j'ai utilisé la méthode vu en cours. J'ai d'abord fait des calculs à la main afin de trouver l'algorithme correct à écrire en langage C.

Le calcul du codage se fait à l'aide du produit matriciel d'une matrice génératrice, qui est constituée d'une matrice d'identité de dimension 4\*4 et d'une matrice entrée 4\*4; et du fichier transformé en base binaire et mis dans un tableau.

Le calcul se fait étape par étape, c’est-à-dire chaque bit entré du fichier dans le tableau est multiplié par chaque colonne de la matrice génératrice.

# Algorithme

## Matrice

La console demande à l'utilisateur d'entrée une matrice qui va être assimilée à la matrice génératrice. Afin d'avoir une matrice génératrice correcte, il y a plusieurs conditions à respecter pour que le calcul matriciel soit bon.

Aucune colonne entrée ne doit être nulle. Pour cela, il fallait une boucle qui vérifie qu'aucune colonne n'a quatre zéros. C’est-à-dire, la boucle se positionne sur la première colonne entrée, et dès qu'elle détecte que la valeur entrée est nulle, elle incrémente le compteur. Si le compteur est strictement à quatre à la fin de la lecture de la colonne, le compteur est remis à zéro, et on passe à la lecture de la colonne suivante. Si le compteur est égal à quatre, la matrice entrée est considérée comme fausse et l'utilisateur doit entrer une autre matrice.

Aucune colonne entrée ne doit être identique à au moins l'un de la matrice d'identité. Pour vérifier, il fallait créer un compteur de un. Si le compteur est égale à un, après la lecture de la colonne, la matrice est considérée comme fausse.

Deux colonnes entrées ou plus doivent être non identiques. Il fallait faire enregistrer chaque colonne dans son propre tableau, et comparer un à un afin d'être sûr qu'il n'y a pas de colonne identique.

## Fichier

Dans cette partie, j'ai mis les fichiers en durs pour avoir des tests qui fonctionnent en attendant de laisser la possibilité à l'utilisateur de choisir. Donc pour nous avons le choix entre quatre types de fichiers: fichier txt (texte), fichier jpeg (image), fichier mp3 (musique) et fichier mp4 (vidéo).

Ensuite, le programme lit le fichier octet par octet e, hexadécimal et met en décimal les valeurs dans un tableau. Ensuite les valeurs sont converties en binaires pour le produit matriciel.

# Fonctionnement

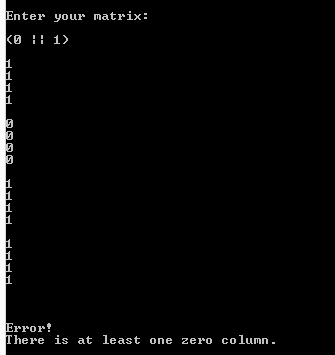
La console demande à l'utilisateur d'entrer une matrice.

Si elle répond aux conditions, les matrices d'identité, entrée et la génératrice sont affichées.

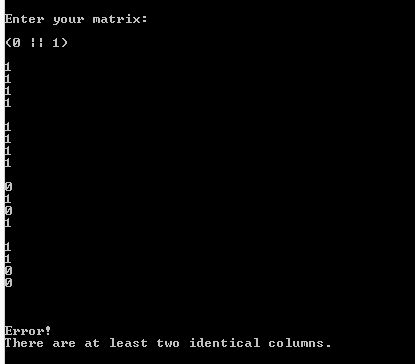
Sinon, l'utilisateur doit entrer à nouveau une matrice.

Les conditions sont:

* Pas de colonnes nulles

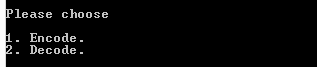


* Pas de colonnes identiques à celle de la matrice d'identité
* Pas de colonnes identiques entre elles

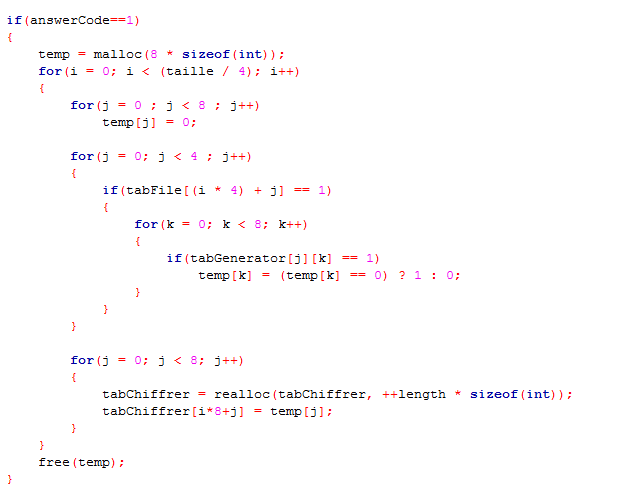


Ensuite, la console demande ce que veut faire l'utilisateur:

* Coder
* Decoder



Si l'utilisateur a choisi de coder le fichier, le calcul se fait en arrière-plan et un message de confirmation s'affiche.



C:\Users\Sophie\Documents\GitHub\MatrixCryptingProject\Images\succes.PNG

Sinon, le fichier est décodé à condition que la matrice entrée corresponde à la matrice du codage utilisé.

# Problèmes rencontrés

J'ai rencontré des problèmes pour les vérifications nécessaires de la matrice entrée. Par exemple, problème de boucle ou de déclarations.

Pour la gestion de fichier, trouver comment mettre un fichier octet par octet dans un tableau, m'a pris énormément de temps car il fallait trouver quelle fonction correspondait le mieux. C'était la fonction fread() qui était la plus utile.

Ensuite, pour mettre dans un tableau, il fallait régler le problème de la taille dynamique, donc faire un malloc() était nécessaire.

Pour convertir un décimal en binaire, j'ai utilisé une fonction qui faisait le calcul.

Pour le calcul du chiffrement, j'ai rencontré des problèmes avec la gestion des tableaux et des boucles.

Lors de la création du nouveau fichier codé, j'ai rencontré un problème pour l'écriture du fichier codé.

# Conclusion

Malheureusement, j'aurais aimé avoir eu plus de temps pour corriger le problème de l'écriture dans le nouveau fichier, développer la partie déchiffrement avec la matrice d'origine enregistrée quelque part et la partie graphique.

J'ai rencontré beaucoup de difficultés pour développer le programme par manque de connaissances nécessaires en C ce qui m'a fait perdre beaucoup de temps.

Néanmoins je ne regrette pas d'avoir choisi ce projet, car c'était très intéressant d'avoir été confrontée à ces difficultés et de trouver des solutions pour les résoudre afin d'avancer sur le projet.