Table des matières

INTRODUCTION

Dans le cadre de réalisation de l'exerce de Travaux pratique pour le course de Cryptographie est Sécurité Informatique , se rapport tant à décrire et expliqué le travaille effectué , détailler la logique utilisé avec comme référence des images du code source .

CONTENU

Enfin de facilité l'utilisation du programme , l'auteur de se projet a pris la peine de concevoir un interface graphique facile a comprendre et a utilisé avec une librairie Java appelé **Javafx** .

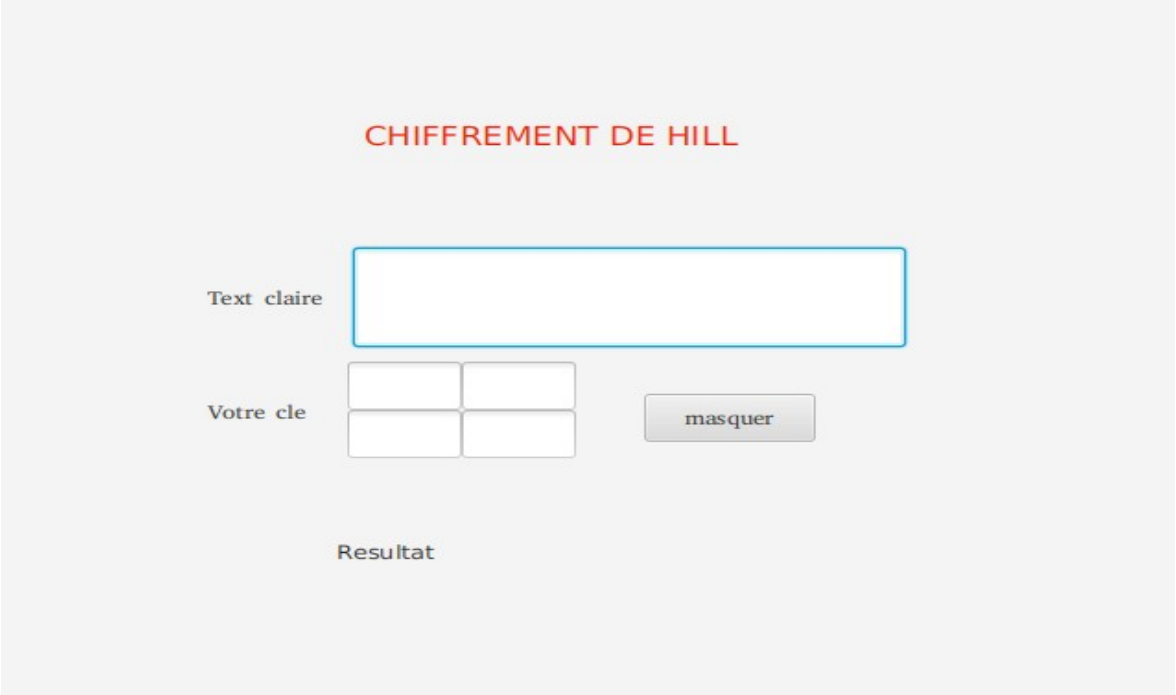

figure 1.1

Derrière la splendeur de cette Interface se cache le fameux algorithme cryptographique étudié par Lester S. Hill et nommé par son noms . Dans les lignes qui suivent j'expliquerai comment j'ai implémenté cette algorithme en forme d'étapes .

Étape 1 : rendre le message paire

cette étape consiste a vérifier si le nombre de caractère du texte claire est paire en réalisant l'opération du modulo avec le nombre 2 comme diviseur , espace exclue , si non on ajoute une lettre optionnelle au message puis on continue l'opération .

```
// for making msg length even
if (msg.length() % 2 != 0) {
    msg += "z";
}
```

Figure 1.2

étape 2 : Convertir les caractère en leur équivalent ASCII

cette étape vas consisté a convertir les caractère selon leur équivalent sur la table ASCII . En java cette opération se fais en castant avec le type entier , puis en faisant la soustraction avec un nombre optionnelle dans le but de ramèné à un intervalle précis .

```
int msgNum[] = new int[msg.length()];
for (int i = 0; i < msg.length(); i++) {
    msgNum[i] = ((int) msg.charAt(i)) - 65;
    // System.out.println(msgNum[i]);
}      You, 2 hours ago • first commit
```

Figure 1.3

étape 3 : Initialisation de la clé

La clé içi est une matrice carré qui ne doit être nulle et le déterminant différent de zéro . Ansi nous pourrons effectuer l'opération de décryptage qui consiste a utilisé l'inverse de la clé cryptographique .

```
int key[][] = new int[2][2];
key[0][0] = Integer.parseInt(a.getText());
key[1][0] = Integer.parseInt(b.getText());
key[0][1] = Integer.parseInt(c.getText());
key[1][1] = Integer.parseInt(d.getText());
```

Figure 1.4

Étape 4 : obfuscation du message

cette étape consiste a masquer la valeur du texte convertis en entier en effectuant une multiplication matricielle . Nous prenons le texte (valeur numérique) , on la découpe en pas de deux si nous utilisons une matrice carré d'ordre 2 . Pour un caractère chiffré, nous effectuons le produit matricielle en additionnant la ligne de clé par la colonne du pas de texte dans se cas 2 puis faisons le modulo par 26 . c'est ansi que nous obtenons le première caractère

```
String eText = "";
for (int i = 0; i < msg.length(); i += 2) {

    int temp1 = msgNum[i] * key[0][0] + msgNum[i + 1] * key[1][0];
    eText += (char) ((temp1 % 26) + 65);

    int temp2 = msgNum[i] * key[0][1] + msgNum[i + 1] * key[1][1];
    eText += (char) ((temp2 % 26) + 65);
}
```

Figure 1.6

CONCLUSION

Le chiffrement de Hill offre plusieurs avantages clés, notamment sa capacité à fournir une sécurité robuste en mélangeant les lettres du message de manière complexe. Il est résistant aux attaques par analyse fréquentielle en raison de sa nature polyalphabétique. De plus, il permet de chiffrer des blocs de lettres simultanément, ce qui le rend efficace pour le traitement de données en masse. En outre, sa structure mathématique offre une flexibilité pour des applications variées.

Selon lien