



Université Paris-Est Créteil Val de Marne
Master 1 Informatiques

La boîte à outil

La cryptanalyse symétrique

Réalisé par :

SAIDANI Meriem

HARROUCHE Ghania

MAMOUNI Walid

SOUALAH Youba

Encadré par :

Mr Alexis Bes

07/07/2014

Inv

Organisation du document

I. Conception	4
1. Présentation générale des chiffrements	4
1.1. Aperçu historique	5
1.1.1. Le chiffrement de César	5
1.1.2. Transposition	5
1.1.3. Le chiffrement de Vigenère	6
1.1.4. Chiffrement par substitution	6
1.1.5. Le chiffrement par transposition	7
1.2. Les règles qui assurent la relation complexe entre (M, K, C)	7
2. La Cryptanalyse	8
2.1. Les attaques	8
2.1.1. Analyse des fréquences	8
2.1.2. L'attaque à l'aide de l'analyse statistique	9
2.1.3. L'attaque en force (Brute force attack, Exhaustive key search attack)...	9
2.1.4. L'attaque à l'aide de texte chiffré seulement	9
2.1.5. L'attaque à l'aide de texte clair (Known--plaintextattack) ou de texte clair choisi (Chosen--plaintextattack)	9
II. Modélisation du projet	9
1. langage utilisé « java »	10
2. L'attaque sur la boîte noire	10
3. La boîte à outil	10
3.1. L'interface graphique	10
3.2. Les fonctions principales	23
4. Gestionnaire de version	29
4.1. Présentation du gestionnaire de version	29
4.2. Types principaux de logiciels de gestion	29
5. Description du plan de travail	30
5.1. Problèmes rencontrés	31
5.2. La répartition des tâches	31
Conclusion	35

Introduction

Bien que la cryptologie moderne s'est scindée en deux catégories : la cryptographie et la cryptanalyse. La cryptographie est une discipline visant à garantir la confidentialité, l'intégrité et l'authenticité lors d'échanges d'informations entre deux parties. Elle a des applications dans plusieurs secteurs civils et militaires. D'ailleurs, la sécurité du commerce électronique ainsi que la protection de la vie privée reposent de plus en plus sur cette science.

La cryptographie à clé secrète (ou symétrique) est la forme la plus ancienne de chiffrement. Elle suppose qu'une clef est a priori partagée par les deux parties, sa souplesse et sa facilité d'adaptation aux besoins concrets de l'industrie ont permis de rester très active.

Les chiffrements à clef secrète ne sont pas pour autant infaillibles. En effet, un crypto-système sans faille, du point de vue mathématique, est concevable. Cependant, sa grande complexité du point de vue implémentation et son coût d'utilisation le rendent prohibitif. Il ne serait donc pas d'une grande utilité pour l'industrie.

Chercher les failles de ces systèmes "imparfaits" est donc le but de la cryptanalyse. Cette discipline regroupe tous les moyens de casser les crypto systèmes sans avoir aucune idée de leurs fonctionnement interne (boite noire), et de retrouver des messages sans posséder leur clef en utilisant tous ces failles on va étudier une boite noire (crypto-système symétrique) simple, faible, et fortement inspiré d'un système réel et de mettre en évidence ses faiblesses. On disposera d'un certain nombre de triplets (clair, clé, chiffré).

Organisation du document

Nous commencerons ce rapport par une introduction aux crypto-systèmes symétrique. On décrira ensuite les règles qui rendent la relation entre message clair et son chiffré et la clé complexe.

La deuxième partie sera conçue sur la partie pratique ou on commence par le langage proposé pour l'implémentation, ensuite l'interface graphique et les fonctionnalités implémentées dans la boite à outil ensuite la présentation du gestionnaire de version choisi et enfin le plan de travail.

I. Conception

1. Présentation générale des chiffrements

La cryptographie utilise un chiffre pour coder un message. Le déchiffrement est l'opération inverse, par une personne autorisée à retrouver le message clair.

[1]

Uncryptosystème est la donnée:

- d'un ensemble fini M de messages (textes) clairs
- d'un ensemble fini C de cryptogrammes (messages cryptés)
- d'un ensemble fini K de clefs
- d'une application d'encryptage

$e : M \rightarrow C$

Et d'une application de décryptage

$d : C \rightarrow M$

Telles que, pour tout $M \in M$ et tout $K \in K$

$d(e(M, K), K) = M$

-Cryptosystème symétrique [2]

Si $e = d$, la clef $K = e = d$ est dite symétrique de même que le cryptosystème

Et on dit aussi, dans ce cas, que la clef est secrète (ou bien privée).

-Cryptosystème asymétrique

Si e est publique et d est privée, on dit que le système (ou la clef) est asymétrique ou encore que c'est un système à clef publique. Dans un tel système asymétrique, le calcul de e (resp. d) en fonction de d (resp. e) doit être infaisable pratiquement. [2]

1.1. Aperçu historique

1.1.1. Le chiffrement de César

Il est considéré comme l'un des premiers crypto-systèmes à clef secrète. La clef K correspond à un chiffre entre 0 et 25. Pour chiffrer un message, chaque lettre de ce dernier est décalée de K positions (décalage dans l'ordre alphabétique). Déchiffrer le message revient à décaler dans le sens inverse. [3]

En pondérant les lettres de l'alphabet par des coefficients (de 0 à 25), ces deux opérations peuvent être vue comme une addition dans le groupe $(\mathbb{Z}/26\mathbb{Z}, +)$. Le chiffrement revient à additionner le coefficient de la lettre avec K . On déchiffre en ajoutant $26 - K$, toujours dans $\mathbb{Z}/26\mathbb{Z}$.

1.1.2. Transposition

Construire des anagrammes en changeant l'ordre des lettres. Connue depuis l'Antiquité (scytale des Spartes) [4] L'ordonnement des lettres doit suivre un système rigoureux sur lequel d'expéditeur et l'envoyeur se sont préalablement entendus.

1.1.3. Le chiffrement de Vigenère

Le chiffre de Vigenère est un système de chiffrement poly-alphabétique, c'est un chiffrement par substitution, mais une même lettre du message clair peut, suivant sa position dans celui-ci, être remplacée par des lettres différentes, contrairement à un système de chiffrement mono-alphabétique comme le chiffre de César [4] (qu'il utilise cependant comme composant). Cette méthode résiste ainsi à l'analyse de fréquences, ce qui est un avantage décisif sur les chiffrements mono-alphabétiques.

1.1.4. Chiffrement par substitution

Il existe 4 méthodes de chiffrement par substitution:

✓ La substitution mono-alphabétique:

Le codage par substitution mono-alphabétique est le plus simple à imaginer. Il s'agit de remplacer chaque lettre ou symbole par un caractère différent. Son principal avantage est la rapidité de la mise en œuvre. Un des désavantages de cette méthode est la longueur de la clé (qui est égale au nombre de caractères dans l'alphabet).

De plus, la plus grande faiblesse de cette méthode est que : dans les langues, toutes les lettres n'ont pas la même fréquence d'apparition. Cette propriété rend illusoire le nombre de clés possibles (théoriquement égal à $26!$, ce qui en soit est un chiffre énorme).

✓ La substitution poly alphabétique:

Pour coder un message, on choisit une clé de longueur arbitraire. On écrit ensuite cette clé sous le message à coder. Si la clé n'est pas assez longue pour couvrir tout le mot on la répète.

Cet algorithme de cryptographie possède deux principaux avantages :

- Une lettre peut être codée différentes façons, ce qui le rend invulnérable à une attaque

Par analyse statistique.

- L'on a une infinité de clés et il n'y a pas de règle quant au choix de la clé.

✓ La substitution homophonique:

Pour échapper à l'analyse de fréquences, la substitution homophonique(ou substitution à représentations multiples) a été proposée. Son principe consiste à remplacer une lettre non

pas par un symbole unique, mais par un symbole choisi au hasard parmi plusieurs, cela empêche la mise en correspondance des lettres les plus fréquentes avec les symboles les plus représentés dans le texte chiffré.

✓ **La substitution par polygrammes:**

La substitution par polygrammes est moins connue que les trois méthodes que nous venons de présenter. Dans cette méthode, au lieu de substituer des caractères, on substitue par des polygrammes (souvent par des groupes de lettres). [5]

1.1.5. Le chiffrement par transposition

Les méthodes de cryptographie par transposition sont celles pour lesquelles on chiffre le message en permutant l'ordre des lettres du message suivant des règles bien définies. Autrement dit, on produit une anagramme du message initial.

Du fait qu'on ne change pas les lettres du message initial, on pourrait imaginer que ces procédés de chiffrement ne sont pas sûrs du tout. C'est effectivement le cas si on chiffre de petits messages, comme des mots, où le nombre d'anagrammes est très réduit. Mais dès que l'on s'intéresse à des messages assez grands, le nombre de transpositions possibles est extrêmement grand, et il est impossible de tester toutes les permutations possibles. Cela dit, il faut que l'expéditeur et le destinataire se mettent d'accord sur une façon de permuter les caractères de façon assez régulière pour qu'elle puisse s'appliquer à n'importe quel message. C'est ce choix qui va rendre le chiffrement par transposition plus ou moins résistant aux attaques. [6]

1.1.6. Le chiffrement par blocs:

On désigne par chiffrement par blocs (block-cipher en anglais), tout système de chiffrement (symétrique) dans lequel le message clair est découpé en blocs d'une taille fixée, et chacun de ces blocs est chiffré.

La longueur n des blocs et la taille l des clés sont deux caractéristiques des systèmes de chiffrement par blocs.

Le message m à chiffrer est découpé en blocs de n bits.

$m = m_1m_2...m_k$

Si la longueur du message n 'est pas un multiple de la longueur d'un

Bloc, on le complète : c'est le bourrage ou padding en anglais. [7]

1.2. Les règles qui assurent la relation complexe entre (M, K, C).

Confusion et diffusion c'est rendre la relation aussi complexe que possible entre M, C, K

M : message

C : cryptogramme

K : clé

1.2.1. Confusion

La **confusion** cache les relations entre le texte clair et le texte chiffré pour éviter les attaques par analyse statistique un changement d'un seul bit dans le texte clair doit affecter un grand nombre de bits (tous) du texte chiffré. C'est l'effet avalanche: petite modification, grand impact. [8]

1.2.2. Diffusion

La **diffusion** disperse la redondance du texte clair dans le texte chiffré, **par exemple** deux lettres doublées ne doivent pas rester côte à côte après cryptage [8].

2. La Cryptanalyse

La cryptanalyse est l'ensemble des techniques permettant à une personne non autorisée de trouver le contenu d'un message. [9]

Principes de la cryptanalyse:

Les méthodes et outils permettant de découvrir le maximum d'informations secrètes appartenant au domaine de la cryptographie, Clés, textes en clair, langues utilisées, algorithmes de chiffrement. La cryptanalyse cherche à « casser » du code par une combinaison :

- de raisonnement analytique,
- d'application d'outils mathématiques,
- de recherche de modèle,
- de patience, de détermination et aussi de chance

2.1. Les attaques :

2.1.1. Analyse des fréquences

Chaque langue possède un profil particulier en ce qui concerne la fréquence des lettres, des bigrammes, trigrammes, etc.

- Ceci peut être utilisé pour monter une attaque à texte crypté seulement (attaque distinctive, décryptage, récupération de la clé).

- Contre les substitutions monoalphabétiques ou de poly grammes.
- On peut contrer partiellement cette attaque en réduisant la redondance du message (p.ex. compression). [10]

2.1.2. L'attaque à l'aide de l'analyse statistique (Statistical analysis attack).

Le cryptographe possède des informations sur les statistiques du message clair (fréquences des lettres ou des séquences de lettres). Exemple l'attaque sur le chiffrement par bloc Vigenère (substitution poly-alphabétique), [11] selon la taille de la clé, les statistiques peuvent être utilisées sur les lettres qui sont chiffrées de la même façon plus clairement si on a une clé de taille 3 :

- ✓ Les lettres du texte en clair qui sont aux positions: 0, 3, 6, Sont chiffrées de la même façon: même décalage.
- ✓ Les lettres du texte en clair qui sont aux positions: 1, 4, 7, Sont chiffrées de la même façon: même décalage.
- ✓ Les lettres du texte en clair qui sont aux positions: 2, 5, 8, Sont chiffrées de la même façon: même décalage.

2.1.3. L'attaque en force (Brute force attack, Exhaustive key search attack).

Le cryptographe essaie toutes les combinaisons de clés possibles jusqu'à l'obtention du texte clair.

- Attaque à texte crypté seulement.
- Permet la récupération de la clé.
- Applicable en principe à tous les cryptages moins forts que le masque jetable.
- Il "suffit" d'avoir un espace des clés assez grand pour éviter cette attaque. [12]

2.1.4. L'attaque à l'aide de texte chiffré seulement (Ciphertext-only attack).

Le cryptographe dispose du message chiffré par l'algorithme et fait des hypothèses sur le texte clair (expressions, mots, sens du message...)

2.1.5. L'attaque à l'aide de texte clair (Known--plaintext attack) ou de texte clair choisi (Chosen--plaintext attack) :

Le cryptographe dispose des messages ou parties de message clairs et de leur version chiffrée et tente de résoudre les mécanismes de chiffrement de l'algorithme. [13]

II. Modélisation du projet

1. langage utilisé « java » :

Java est un langage de programmation orienté objet et un environnement d'exécution, développé par Sun Microsystems. Il fut présenté officiellement en 1995. Le Java était à la base un langage pour Internet, pour pouvoir rendre plus dynamiques les pages (tout comme le JavaScript aujourd'hui). Mais le Java a beaucoup évolué et est devenu un langage de programmation très puissant permettant de presque tout faire. Java et sa JVM permettent au programmeur d'avoir un très haut niveau d'abstraction par rapport à la machine. Il n'aura donc pas besoin de s'occuper de la compatibilité matérielle. Java met à la disposition du développeur une API très riche lui permettant de faire de très nombreuses choses, Ceci est un énorme avantage, cela augmente grandement la productivité de développement. Java est complètement orienté objet. Java permet de développer des applications d'une façon orientée objet et permet d'avoir une application bien structurée, modulable, maintenable beaucoup plus facilement et efficace. Cela augmente une fois de plus la productivité. Donc en résumé les avantages de Java:

- Portabilité excellente
- Langage puissant
- Langage orienté objet
- Langage de haut niveau
- JDK très riche
- Nombreuses librairies tierces
- Très grande productivité
- Applications plus sûres et stables
- Nombreuses implémentations, JVM et compilateurs, libres ou non
- IDE de très bonne qualité et libres : Eclipse et Netbeans par exemple
- Supporté par de nombreuses entreprises telles que Sun ou encore IBM et des projets comme Apache

[14]

2. L'attaque sur la boîte noire

Cette partie a été réalisée par notre collègue qui consiste à étudier un cryptosystème en boîte noire, c'est à dire, découvrir le fonctionnement d'un système de chiffrement, où on

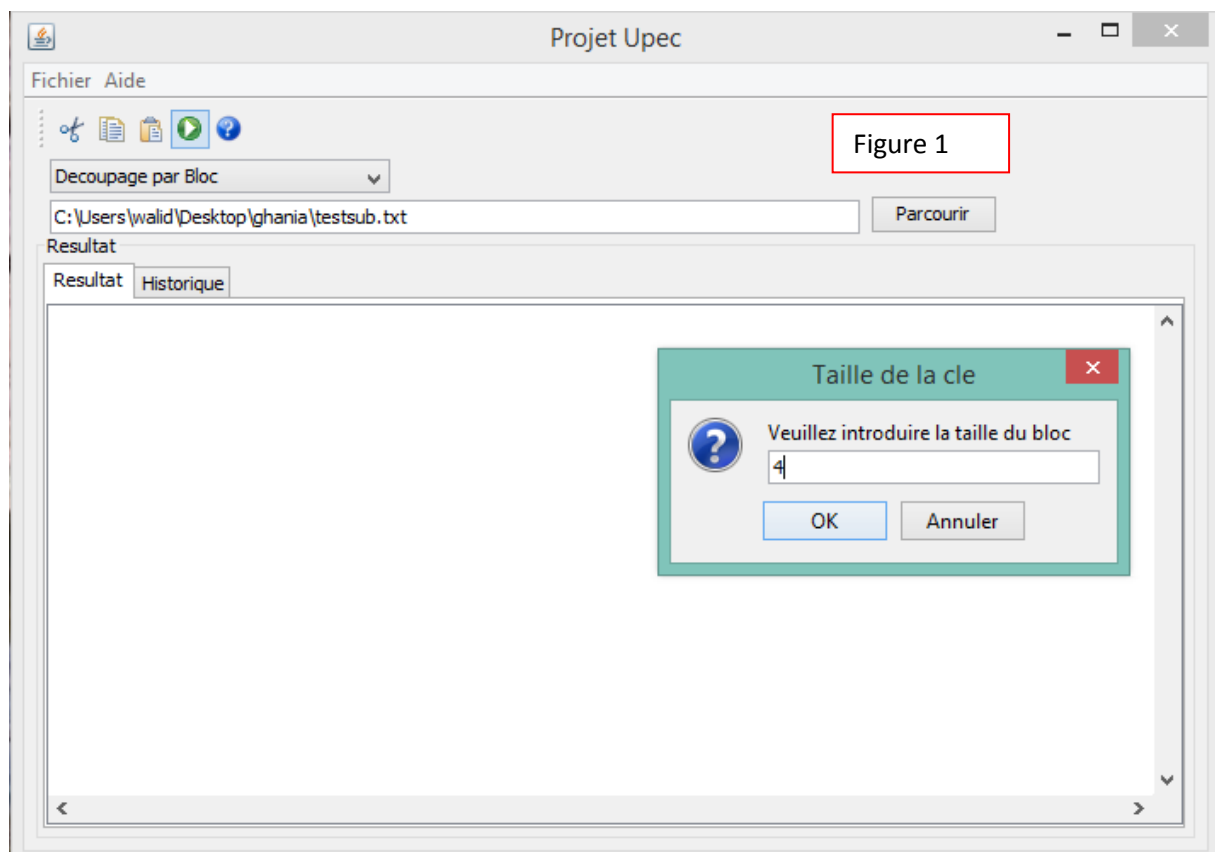
dispose de certaines données, messages clairs, leurs chiffrés correspondants et les clés de chiffrement. L'objectif de cette étude est de pouvoir mettre en évidence ses faiblesses, c'est à dire développer des méthodes qui permettent d'attaquer le cryptosystème. Il a fini par comprendre que le crypto-système utilise un chiffrement par substitution et par transposition (permutation). Il a développé une méthode automatique pour déchiffrer la substitution et une autre pour la transposition.

3. La boîte à outil

3.1 L'interface graphique :

Dans cette partie nous allons décrire les différentes fonctionnalités de notre application :

a) Découpage par bloc : nous allons d'abord choisir dans le menu déroulant la fonctionne souhaité dans ce cas Découpage par Bloc, ensuite nous allons choisir un fichier grâce au bouton « Parcourir » et exécute puis on termine par introduire la taille du bloc et on confirme (Figure 1). Le résultat s'affiche dans la Figure 2



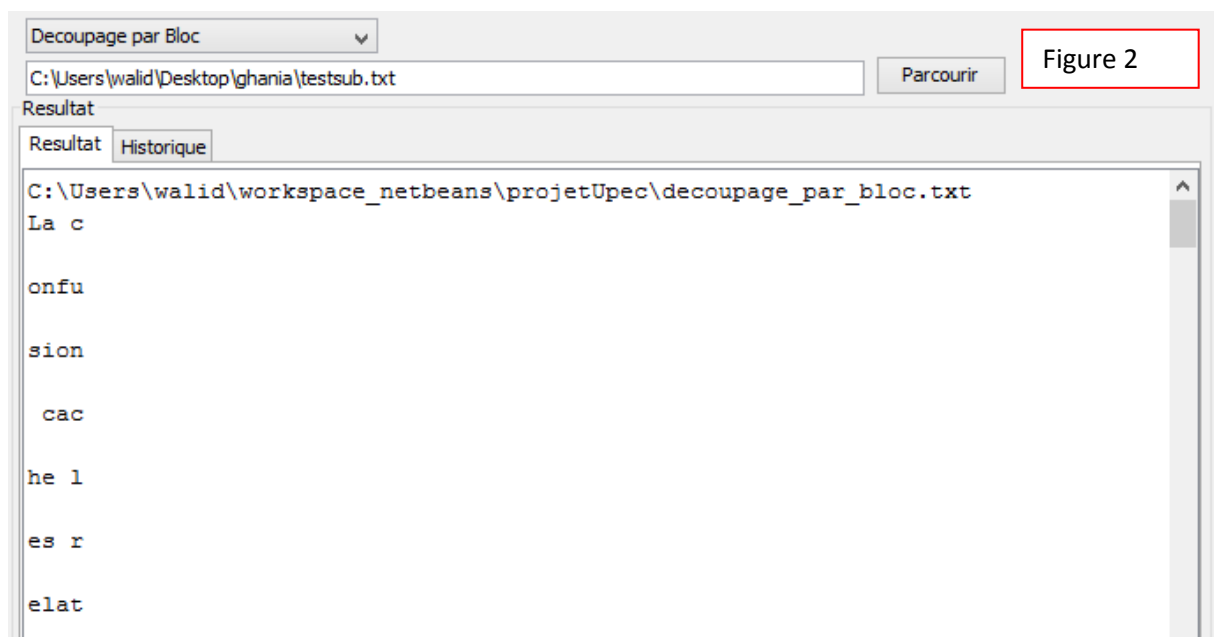


Figure 2

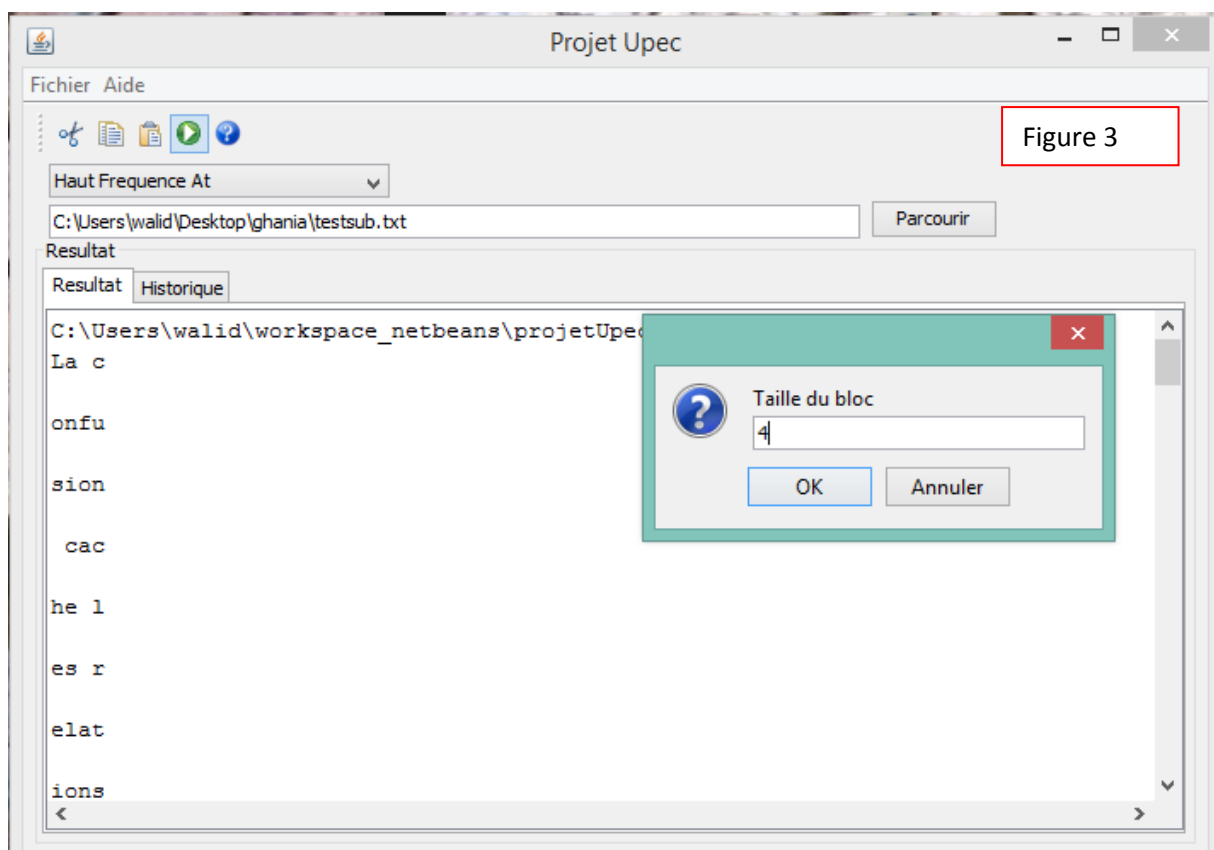
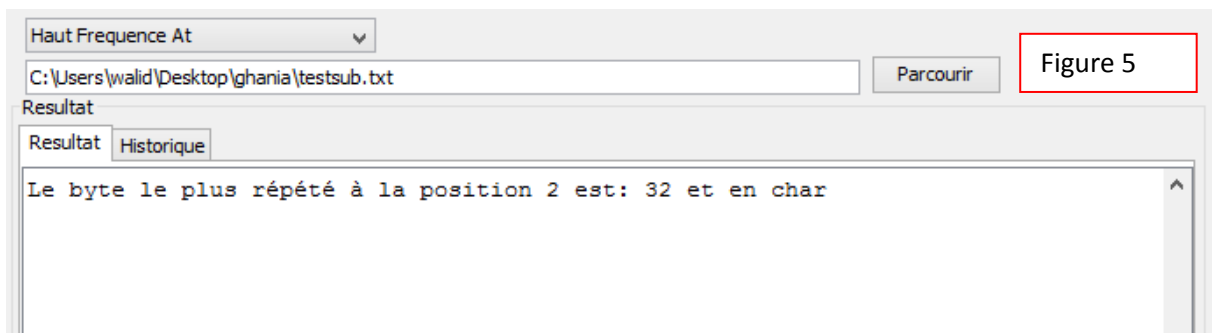
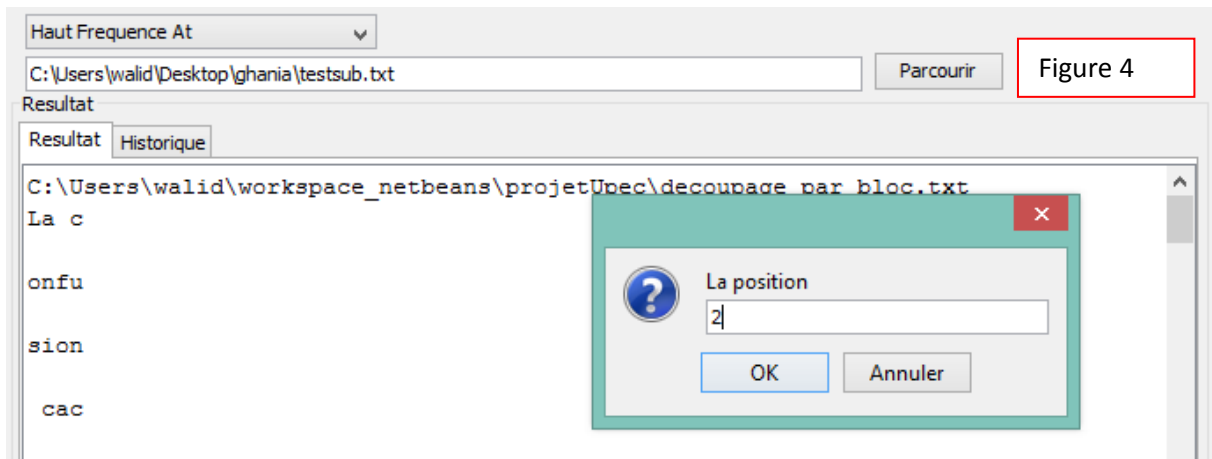
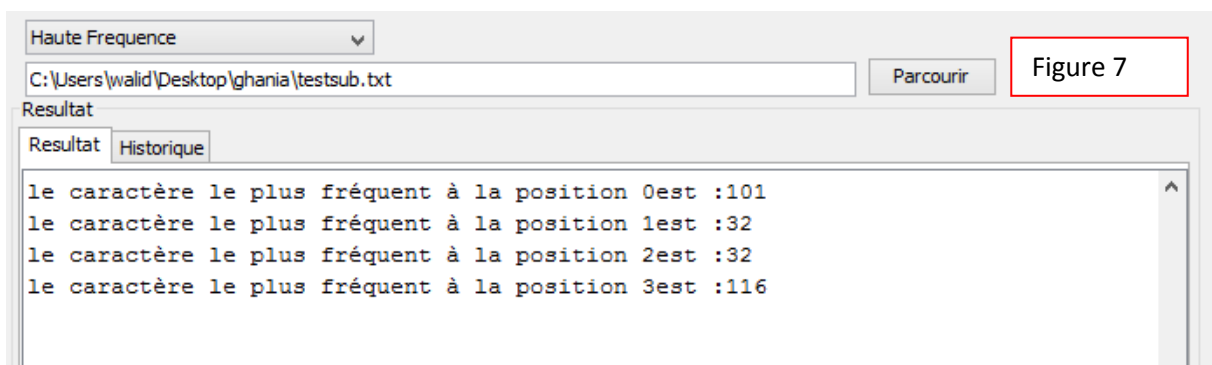
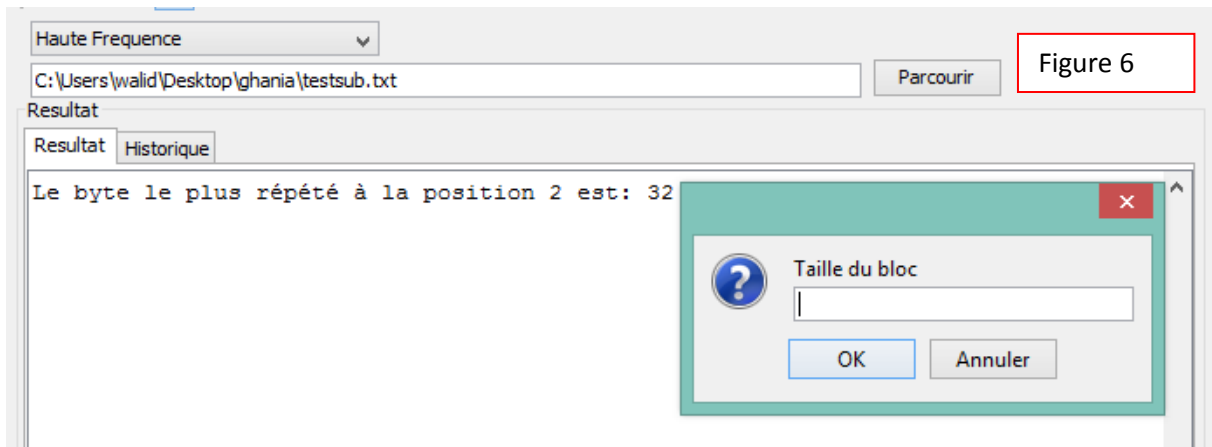


Figure 3

b) HauteFrequenceAt : Après avoir sélectionné la fonctionne et le fichier l'application nous demande d'introduire la taille du bloc et la position du caractère (Figure 3 et 4) une fois les informations saisit et confirmer le résultat s'affiche sur la console (figure 5)



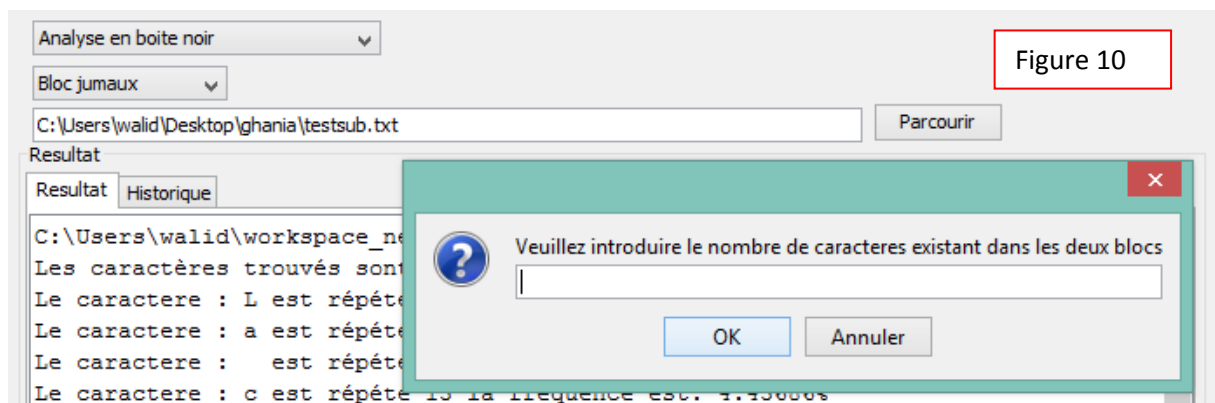
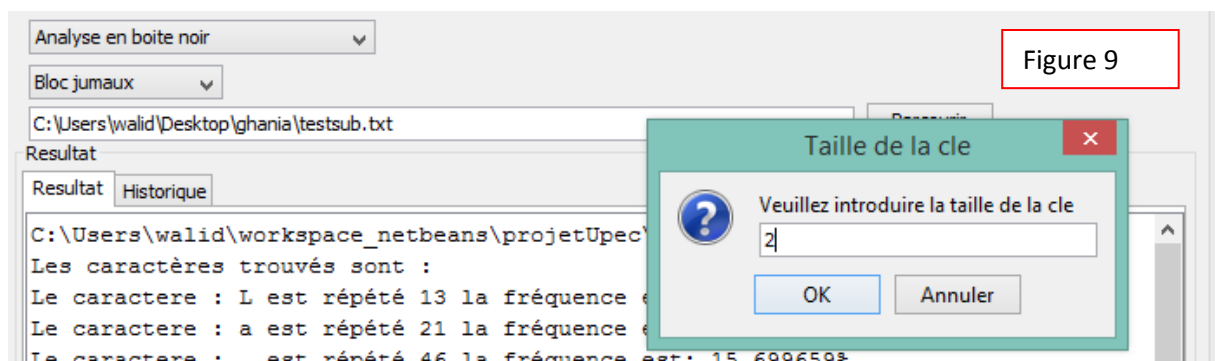
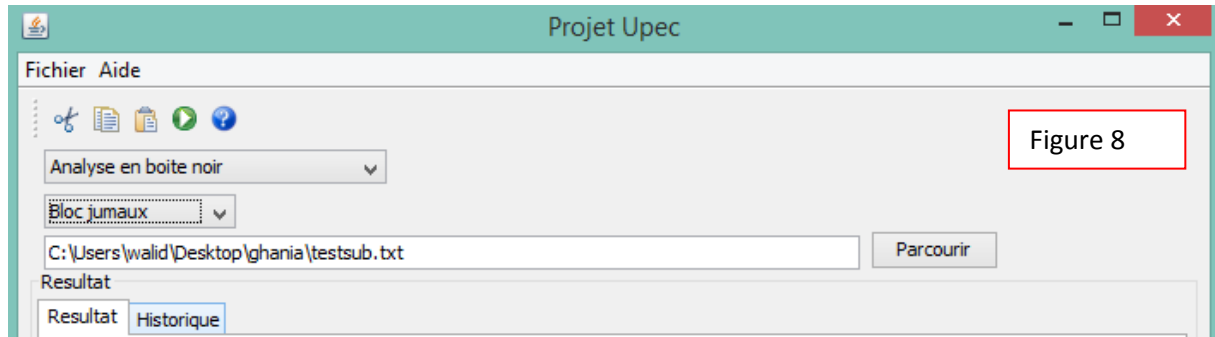
c) HauteFrequence : Même principe que la fonction précédente sauf que cette fois-ci la fonction nous renvoie le caractère à haute fréquence pour chaque position. (Figure 6 et 7)

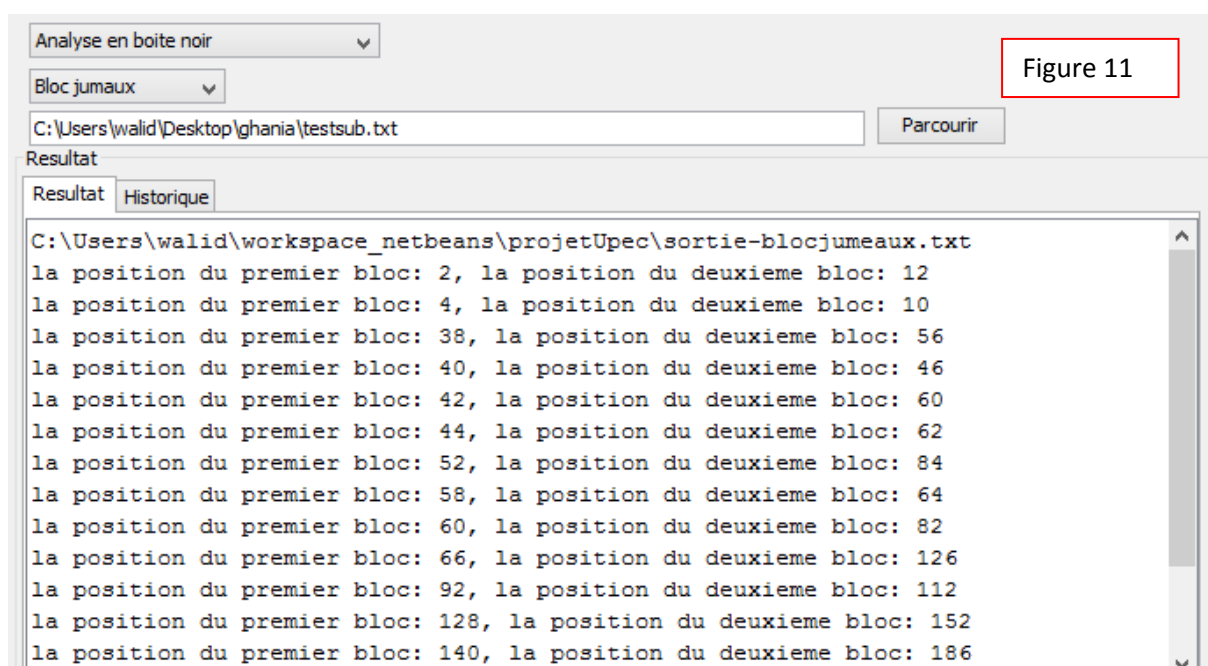


d) Analyse en boîte noire :

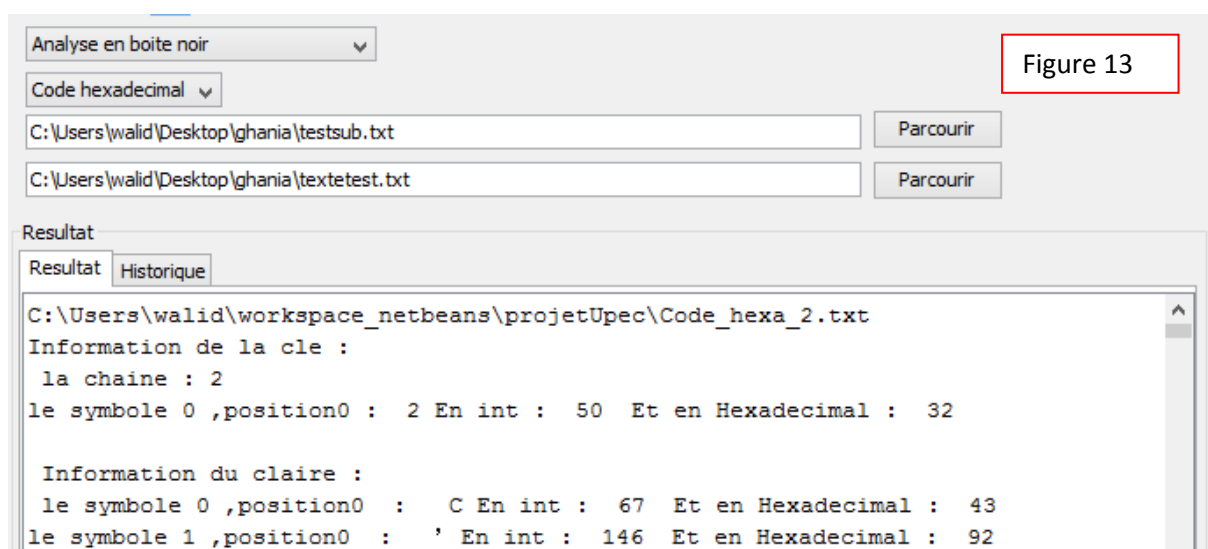
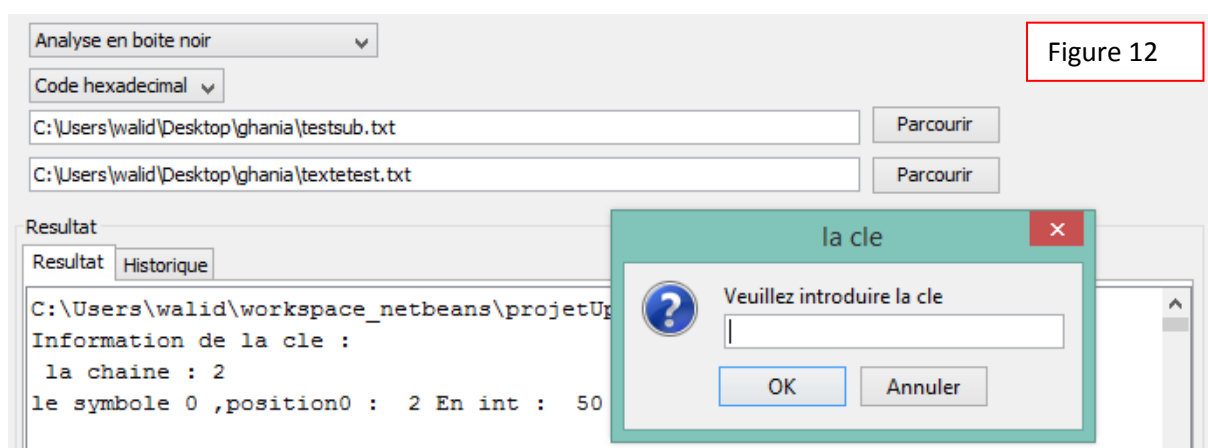
Elle se décompose en trois sous fonction (Bloc jumeaux, Code hexadécimal, Code binaire)

d.1) Bloc jumeaux : Cette fonction nous demande en entrée la taille de la clé et le nombre de caractères existant dans les deux blocs (Figure 9 et 10) le résultat s'affiche sur la Figure 11

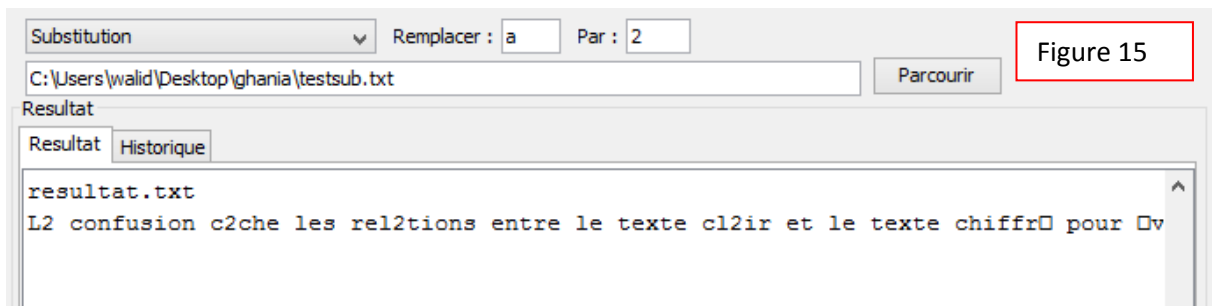
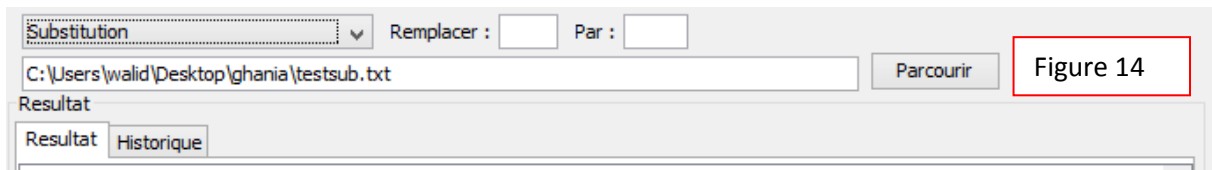




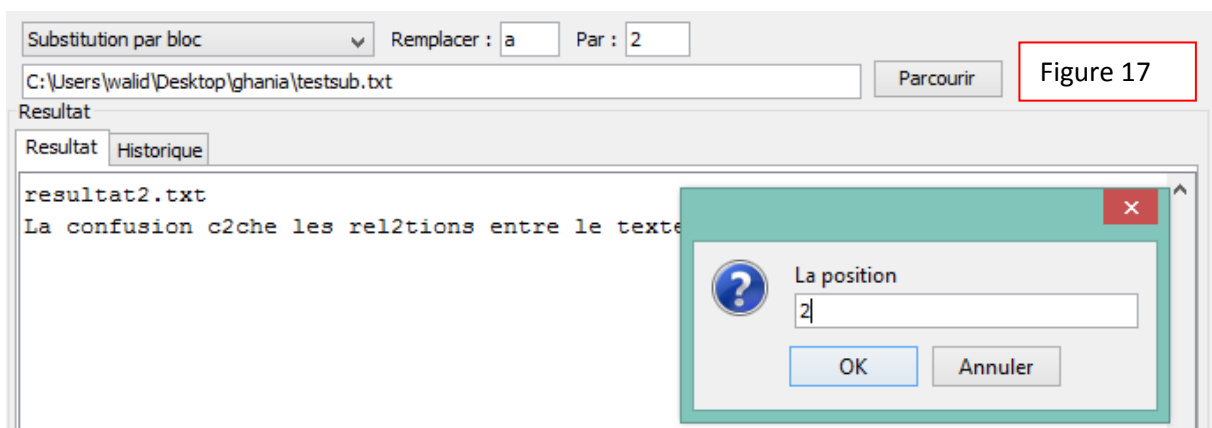
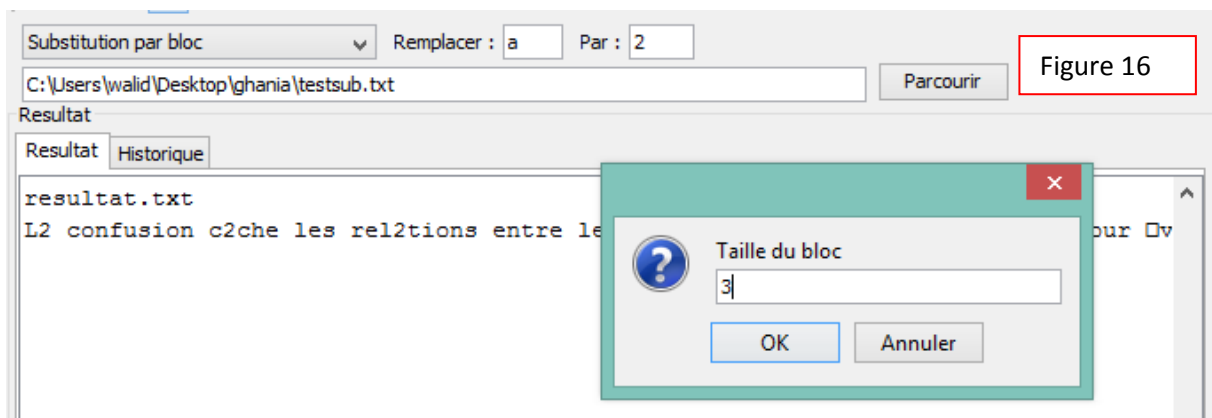
e) Code hexadécimal : Dans cette fonction nous allons choisir deux fichiers après exécution on introduit une clé et on confirme (Figure 12) Le résultat s'affiche sur la (Figure 13), même principe pour Code binaire.



f) Substitution : Après avoir choisi la substitution deux champs apparaissent, « Remplacer » et « Par » on choisit le caractère à remplacer et Par quoi le remplacer (Figure 14) Le résultat s'affiche sur la (Figure 15).



g) Substitution Par Bloc : Cette fois-ci en plus des deux champs «remplacer » et « Par » , on doit indiquer la taille du bloc et la position à laquelle on veut remplacer la caractère (Figure 15 et 16) Le résultat s'afficher sur (La Figure 17).



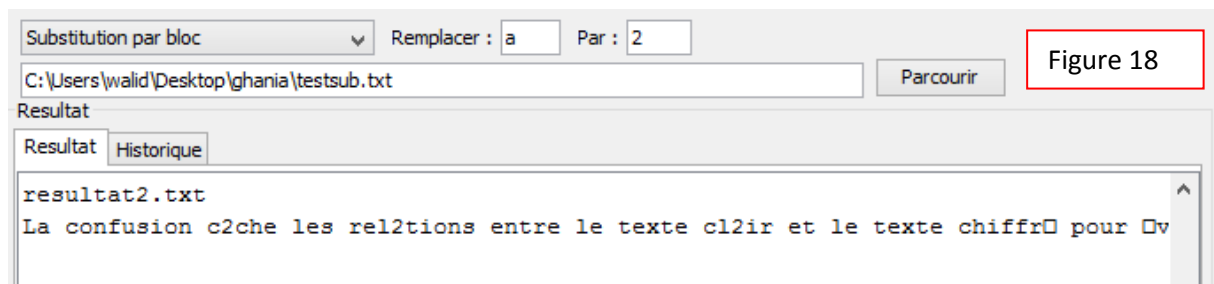


Figure 18

h) XOR : Pour le Xor il existe deux type : (Xor par bloc et Xor par caractère)

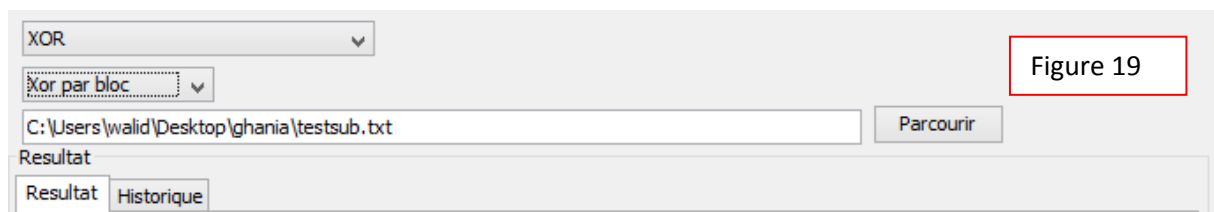


Figure 19

h.1) Xor Par Bloc : Après avoir choisit le fichier, on introduit la clé et on confirme (Figure 20) le résultat s'affiche sur (La Figure 21).

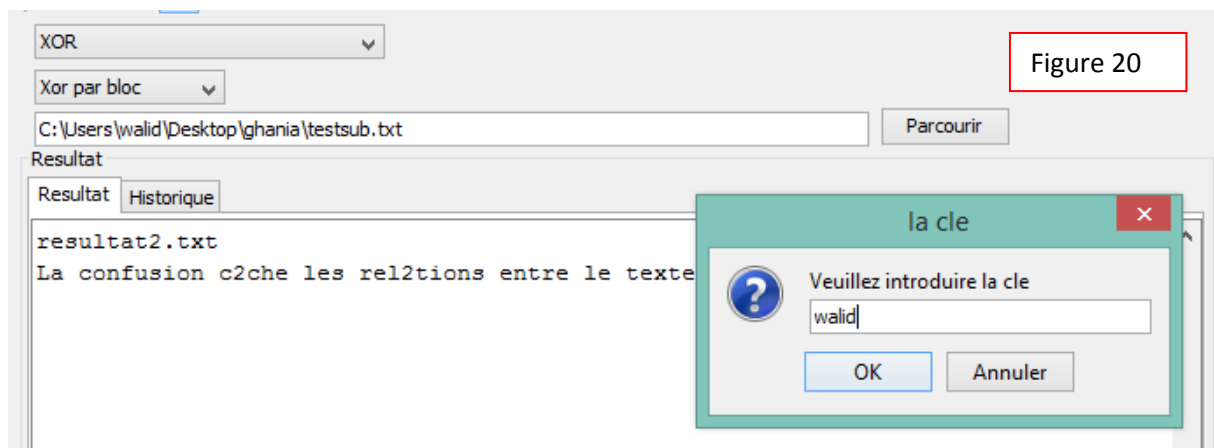


Figure 20

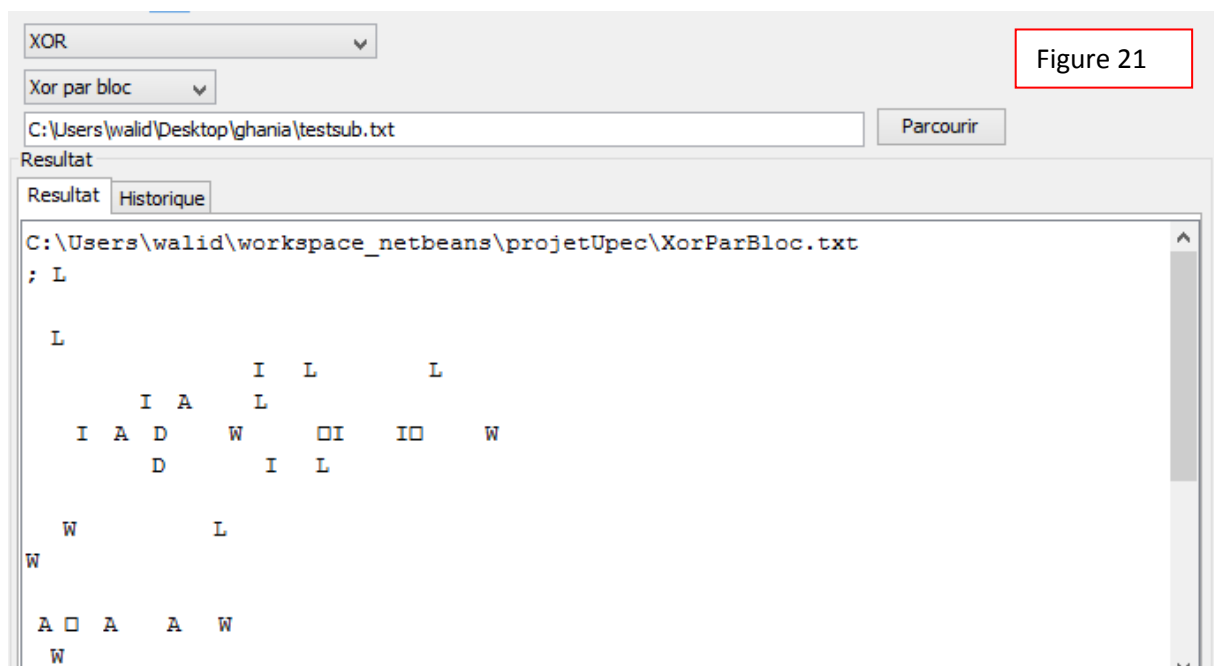
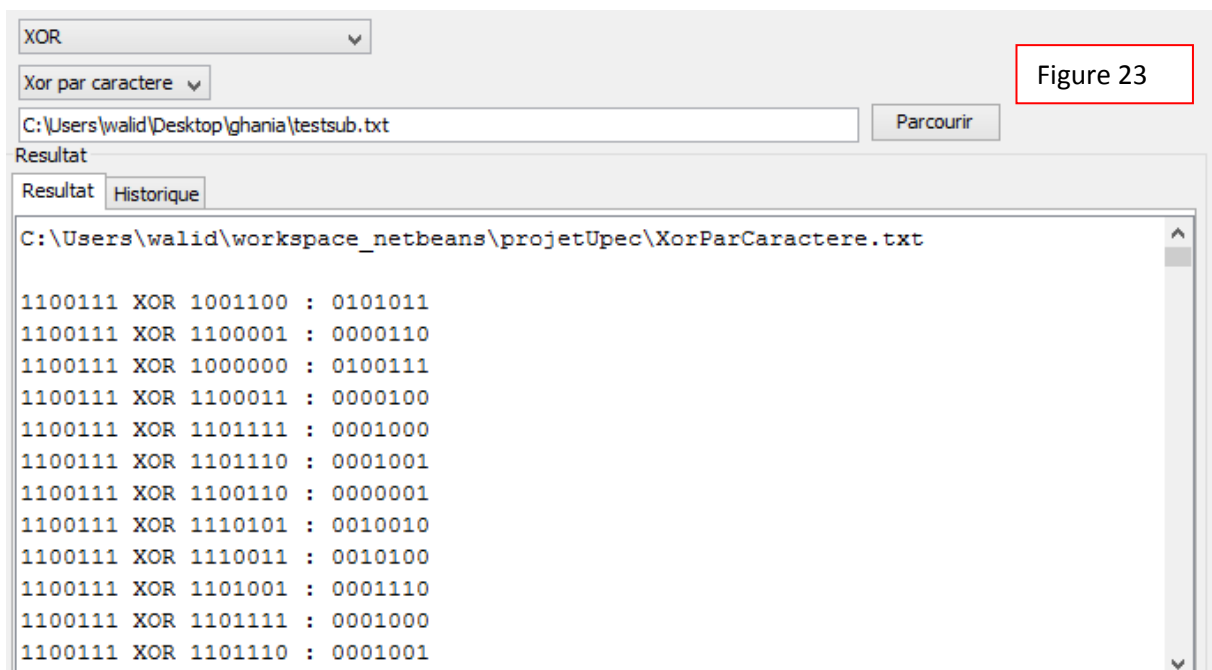
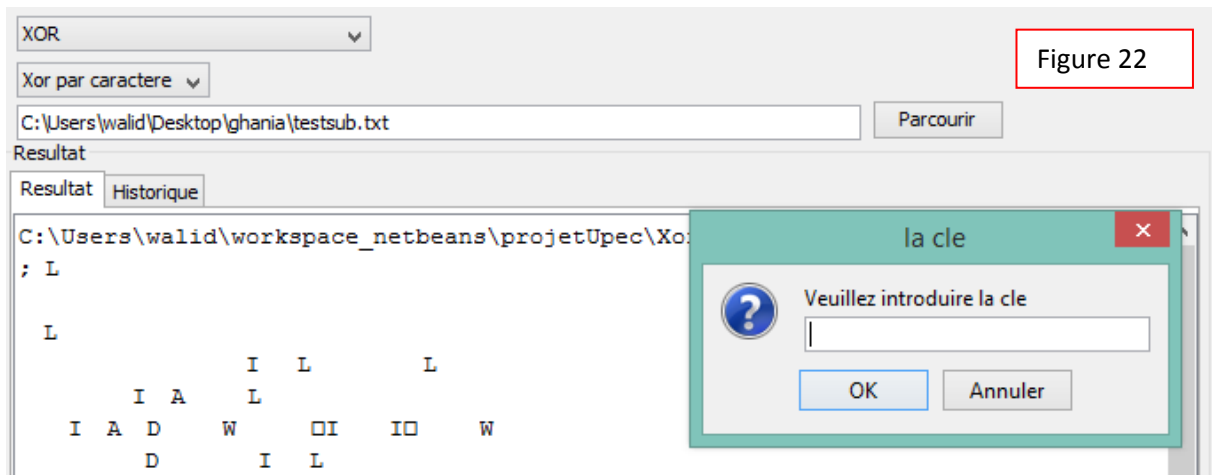
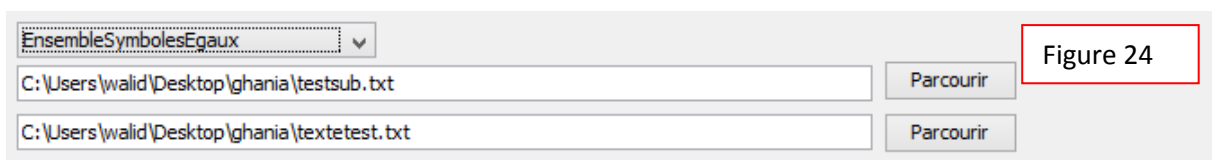


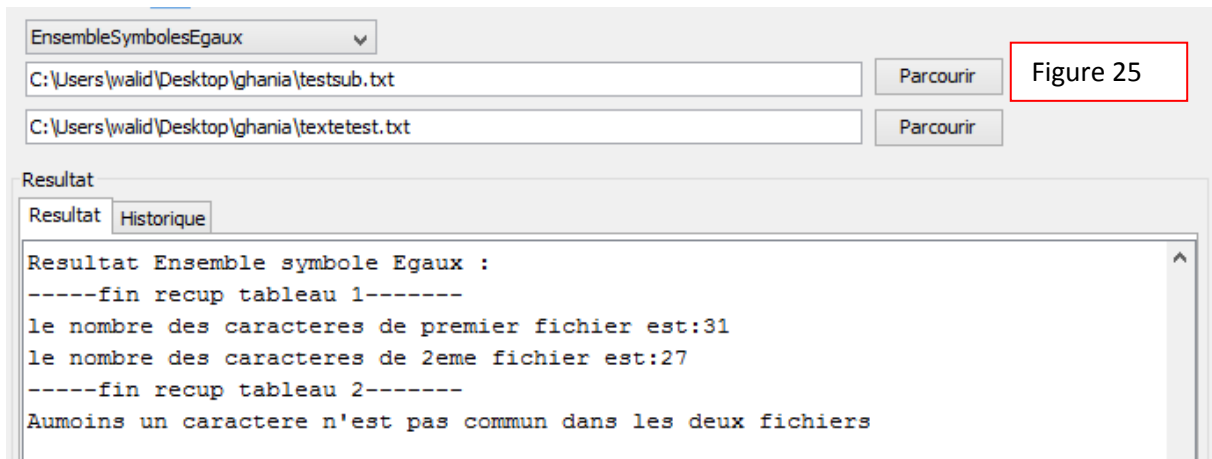
Figure 21

h.2) Xor par caractère : Même principe que le Xor par Bloc (voir la Figure 22 et 23)



i) EnsembleSymbolesEgaux : Cette fonctionne prend deux fichiers en entrée et elle vérifie si y'a des similitudes (voir La Figure 24 et 25).

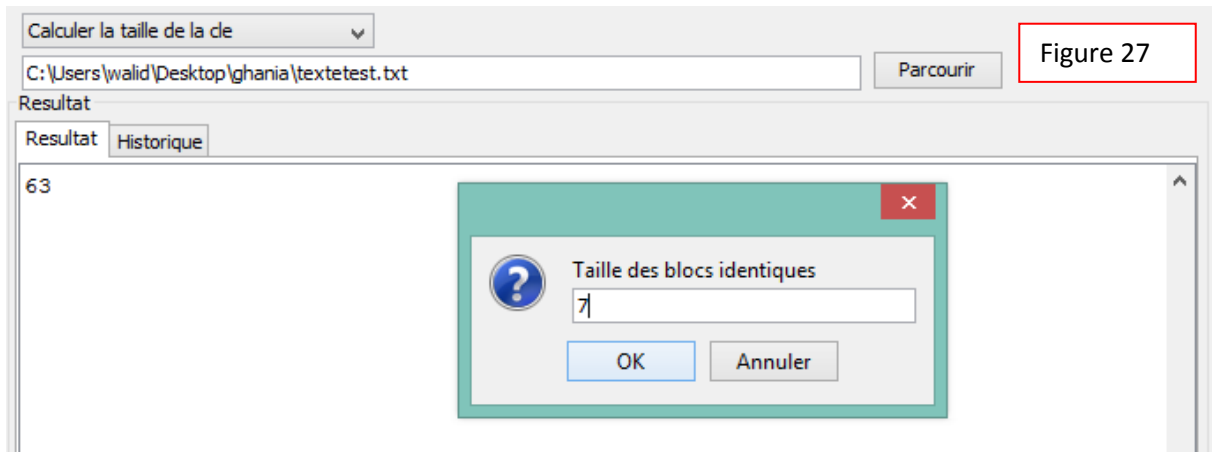




j) LeByteMaximum : une fois le fichier sélectionner et exécuter , la fonction nous rend le plus grand byte du fichier (Figure 26).



K) Calculer la taille de la cle : Dans cette fonctionne on doit sélectionner un fichier et indiquer la taille des blocs identiques elle nous rend la taille de la clé (Figure 27 et 28).

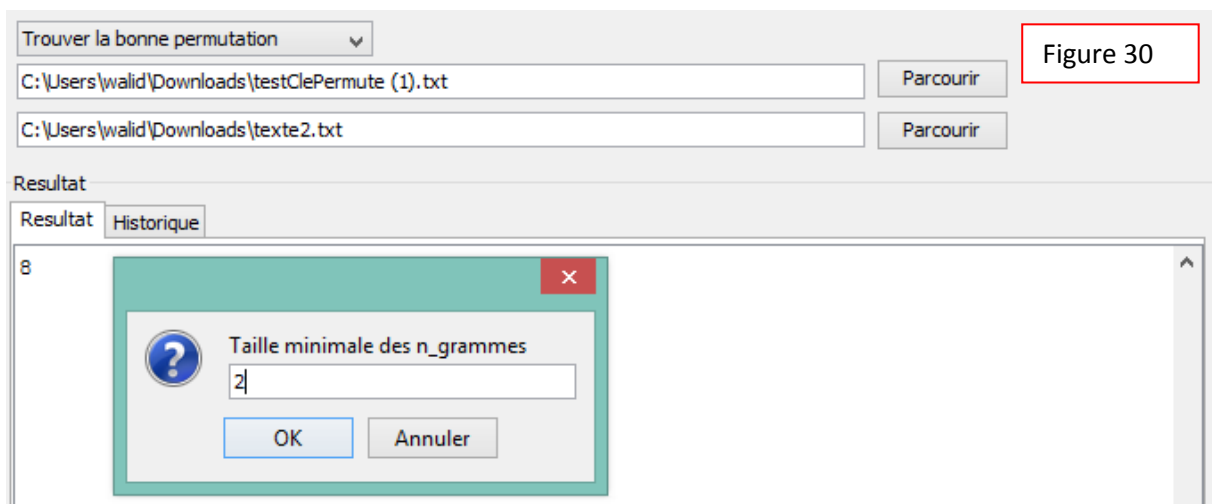


L) Trouver la bonne permutation : Dans cette fonction on sélectionne deux fichier (cryptogramme et n_gramme) ensuite on introduit les paramètres suivants :

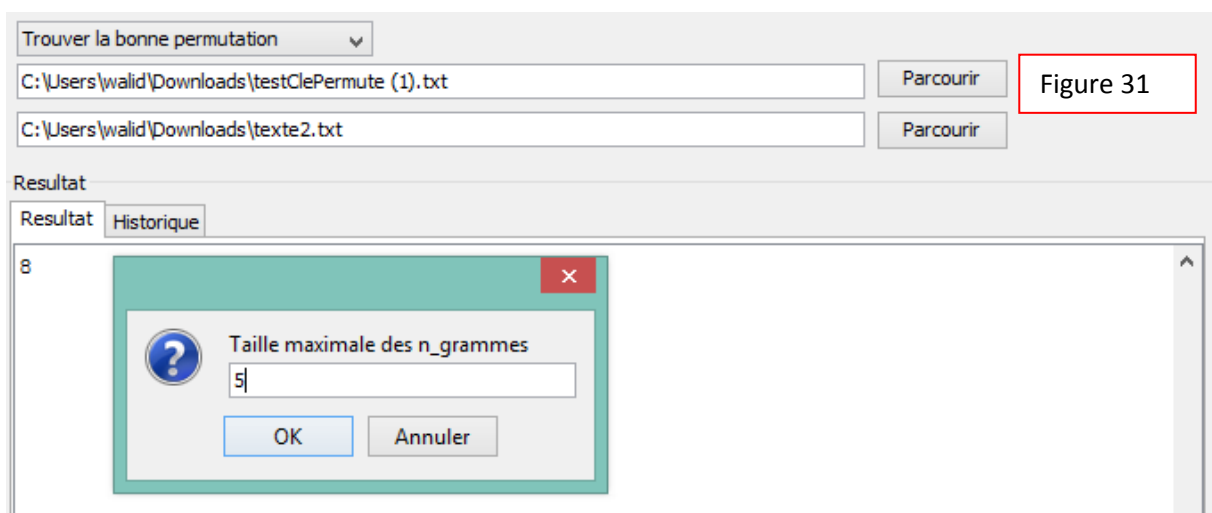
Taille minimale des n_grammes ; Tailles maximale des n_grammes et le nombre des n_grammes pour chaque taille donnée ensuite on indique la taille des blocs identiques qu'elle doit chercher (figures 29, 30, 31, 32, 33) après calcule la bonne permutation s'affiche (Figure 34)



The screenshot shows the 'Trouver la bonne permutation' window. It has a dropdown menu at the top with the text 'Trouver la bonne permutation'. Below it, there are two text input fields for file paths. The first field contains 'C:\Users\walid\Downloads\testClePermute (1).txt' and the second field contains 'C:\Users\walid\Downloads\texte2.txt'. To the right of each field is a 'Parcourir' button. A red box on the right side of the window is labeled 'Figure 29'.



The screenshot shows the 'Trouver la bonne permutation' window with the same file paths as Figure 29. Below the file selection area, there is a 'Resultat' section with two tabs: 'Resultat' and 'Historique'. The 'Resultat' tab is active, and it displays the number '8'. A dialog box is open in the foreground with the title 'Taille minimale des n_grammes'. The dialog box has a question mark icon, a text input field containing the number '2', and two buttons: 'OK' and 'Annuler'. A red box on the right side of the window is labeled 'Figure 30'.



The screenshot shows the 'Trouver la bonne permutation' window with the same file paths as Figure 29. Below the file selection area, there is a 'Resultat' section with two tabs: 'Resultat' and 'Historique'. The 'Resultat' tab is active, and it displays the number '8'. A dialog box is open in the foreground with the title 'Taille maximale des n_grammes'. The dialog box has a question mark icon, a text input field containing the number '5', and two buttons: 'OK' and 'Annuler'. A red box on the right side of the window is labeled 'Figure 31'.

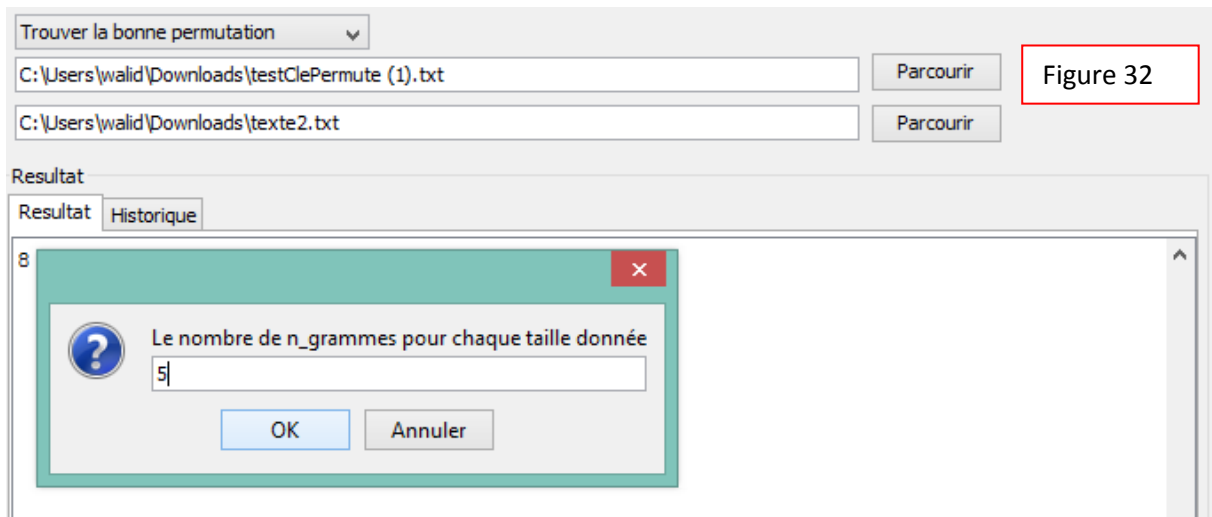


Figure 32

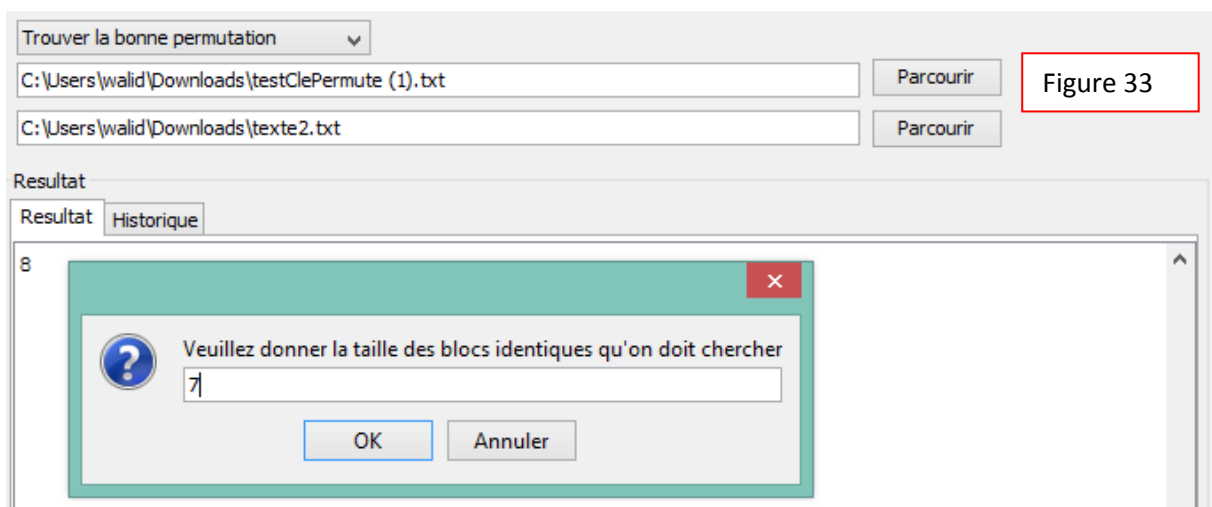


Figure 33

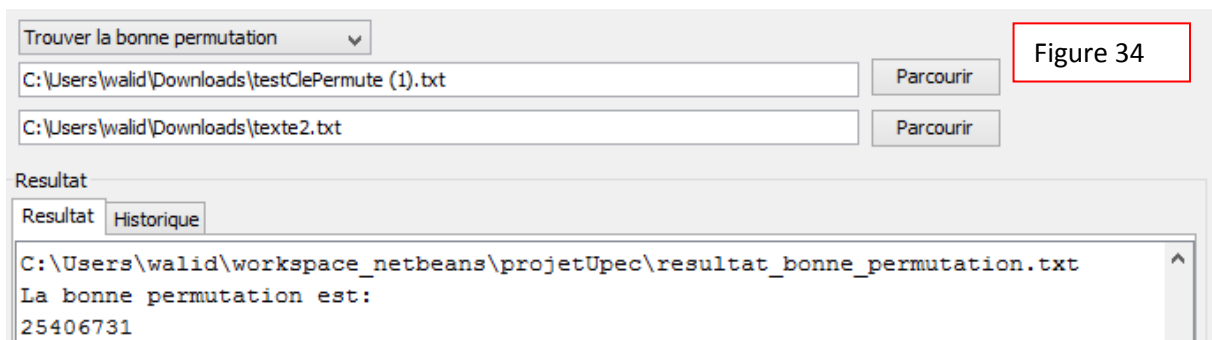


Figure 34

m) LigneCompar : Cette fonction prend en entrée deux fichiers et elle va comparer les lignes. La fonction propose deux cas : soit la ligne est dans le fichier 1 ET le fichier 2, Ou la ligne est dans le fichier 1 mais pas dans le fichier 2 (Figure 35, 36 et 37).

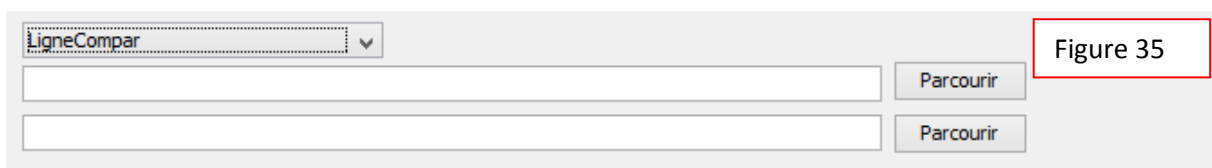


Figure 35

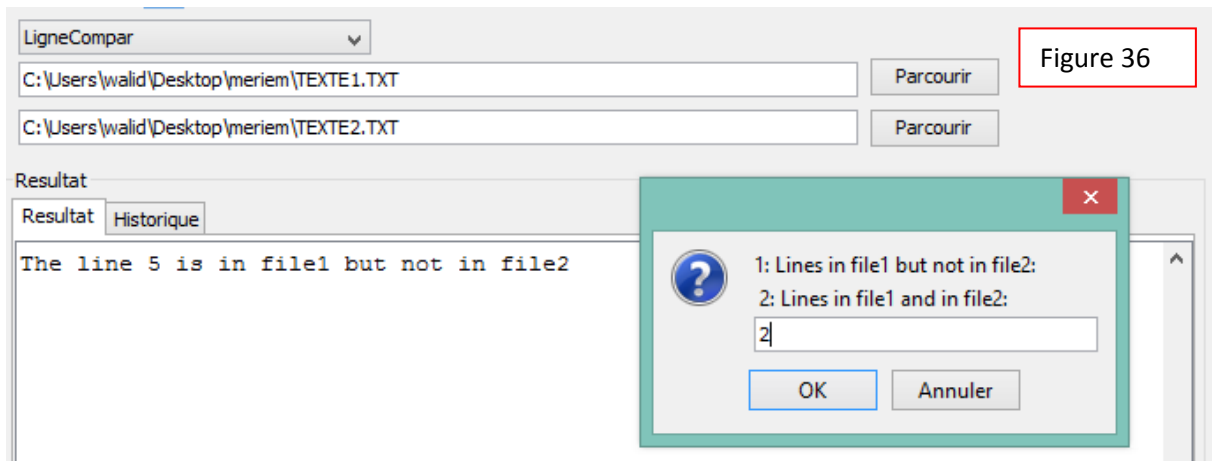


Figure 36

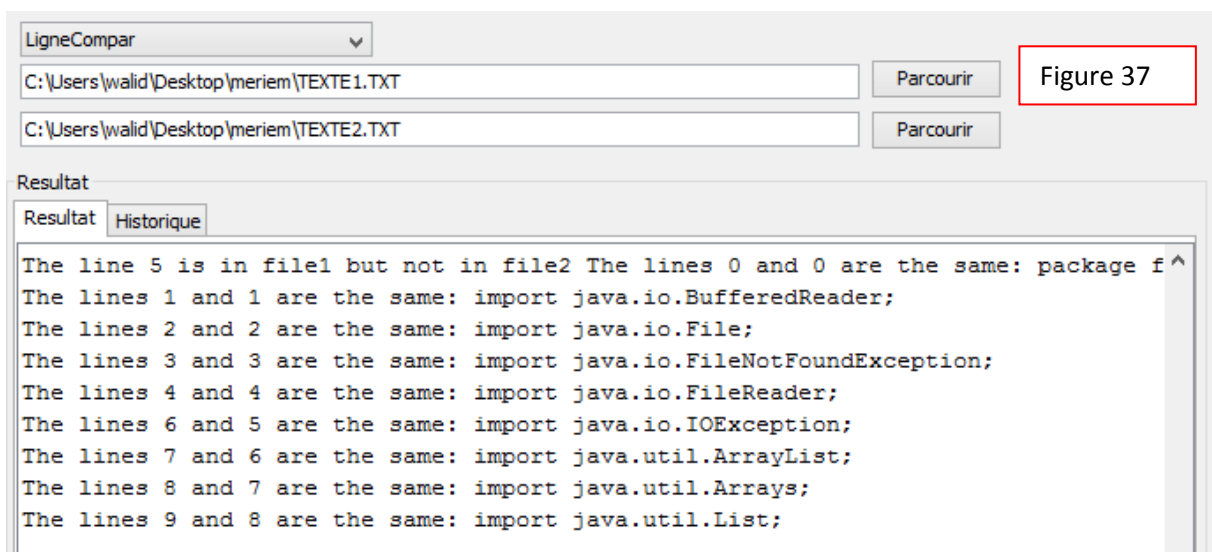


Figure 37

n) CoincidenceFinal : Cette fonction prend un fichier et rend le float IC «Indice de coïncidence » (Figure 38).



Figure 38

o) N_grammes : Après avoir choisi le fichier la fonction nous propose 3 menu :

1) choisir entre Glissant ou non Glissant ;

2)

Si l'analyse est effectué que sur les lettres, \n" +

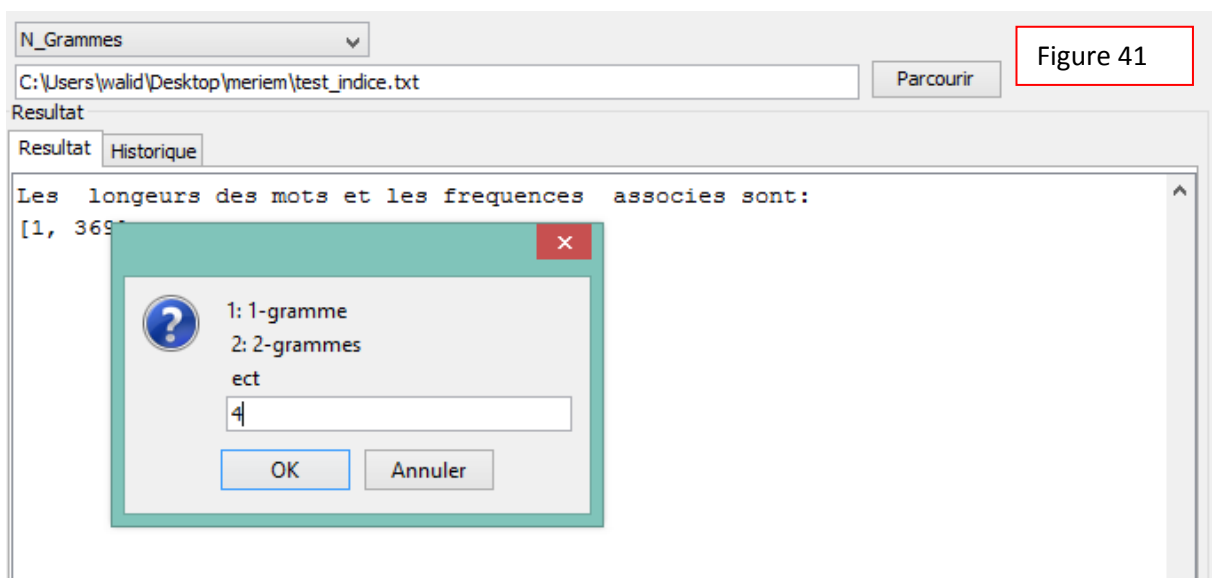
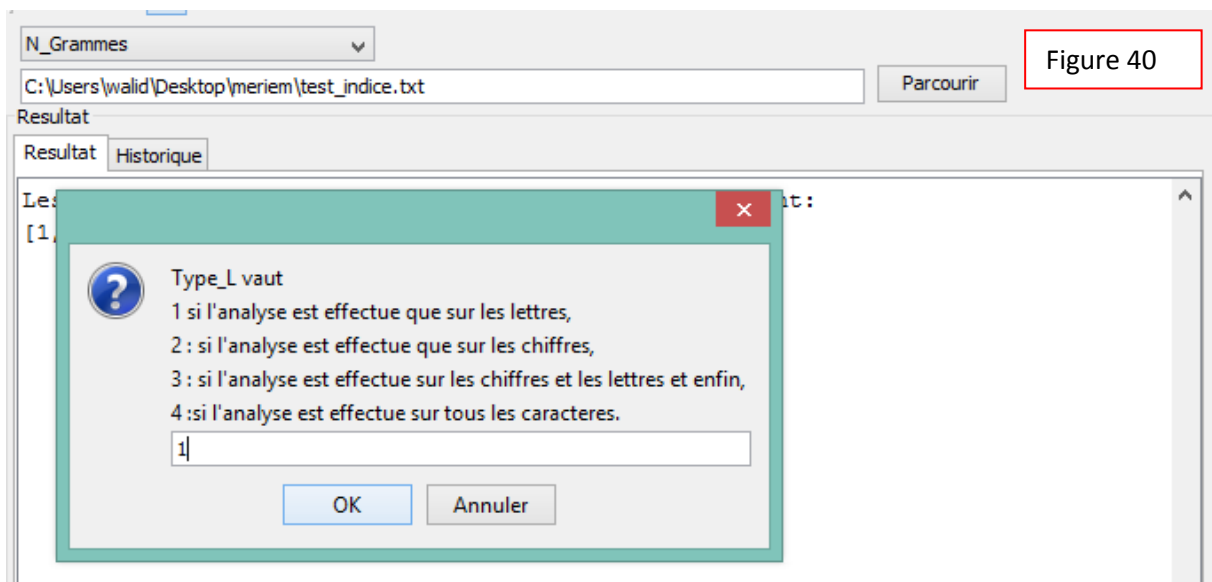
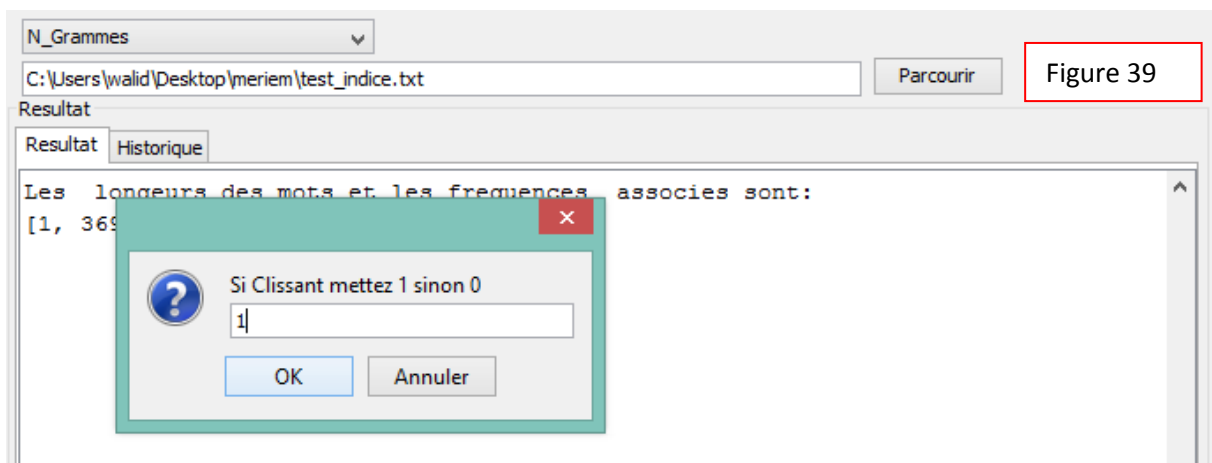
Si l'analyse est effectué que sur les chiffres, \n" +

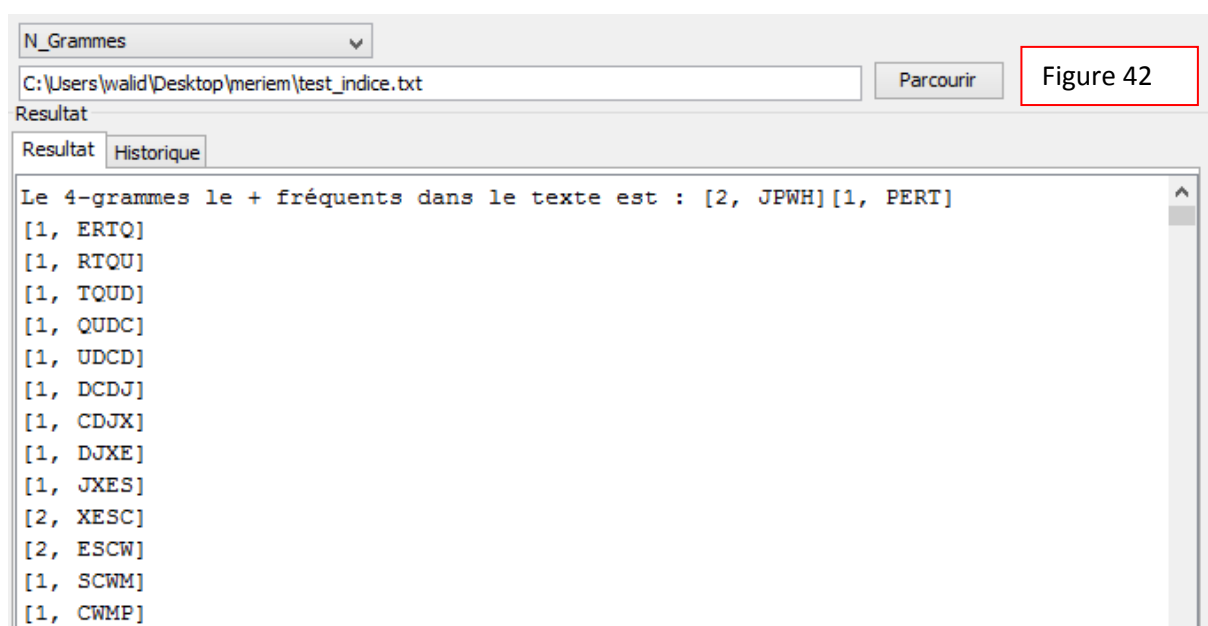
Si l'analyse est effectué sur les chiffres et les lettres et enfin, \n" +

Si l'analyse est effectué sur tous les caractères."

3) choisir le N_gramme correspondant (1_gramme, 2_grammes,... etc)

(Voir Les figures 39, 40,41) le résultat s'affiche sur (la Figure 42)





3.2. Les fonctions principales

3.2.1. La fonctionnalité du modulo:

Elle consiste à chercher le plus grand byte contenu dans un message.

Dans cette fonctionnalité on donne en entrée un fichier texte d'une longueur donnée et on aura en sortie le byte le plus grand contenu dans ce fichier et cette nous indiquera la valeur du modulo avec lequel le cryptosystème peut bien travailler c'est à dire savoir si notre notrecryptosysteme travaille avec un modulo caché. Comme le chiffrement de césar, vigenere.

C'est la classe "taille" dans notre programme.

3.2.2. La substitution:

Elle consiste à remplacer un caractère par un autre caractère dans un texte donné, elle prend en entrée le texte (fichier / tableau), le caractère que l'on veut remplacer et le caractère par lequel on le remplace. En Sortie on aura un texte dans lequel la substitution a été faite.

Utilisation : Un cryptanalyste qui souhaite attaquer un cryptosystème qui utilise le chiffrement par substitution, où il connaît la substitution d'un caractère chiffré pendant ses analyses (par exemple l'analyse de fréquence), il pourra utiliser cette fonction pour éliminer les caractères chiffrés dont il connaît leurs équivalents en clair.

3.2.3. SubstitutionParPositionParBloc:

. Elle consiste à remplacer un caractère à une position donnée dans un bloc, en découpant le texte en blocs de taille T donnée en entrée, vérifier si le caractère se trouvant à la

position souhaité dans ce bloc est égal au caractère que l'on veut remplacer, si c'est le cas, une substitution est faite sur ce caractère.

Utilisation : Cette fonction peut être utilisée pour les cryptogrammes qui ont été chiffrés, avec des systèmes de chiffrement polyalphabétique, qui sont des chiffrements par substitution, mais une même lettre du message clair peut, suivant sa position dans celui-ci, être remplacée par des lettres différentes. Donc le cryptanalyste choisit le caractère à remplacer selon la position qu'il occupe dans le cryptogramme.

3.2.4. CalculeTaille: Elle permet de calculer la taille de la clé si le chiffrement utilisé est un chiffrement par bloc et que tous les blocs sont chiffrés de la même manière. Cette fonction calcule les distances entre chaque deux blocs identiques et rend la bonne taille de la clé par le fait que cette dernière est le plus grand diviseur commun des distances calculées.

Utilisation: Dans la plupart des fonctions cryptographiques, la longueur des clés est un paramètre sécuritaire important. Etant donné un très long cryptogramme, la première des choses à calculer est la taille de la clé avec laquelle le cryptogramme a été chiffré, afin de découper le texte en autant de sous-textes de taille égale à celle de la clé et d'analyser ces derniers. Cette fonction peut être utilisée pour calculer la taille de la clé du chiffrement de Vigenère, Affine par bloc, Hill...etc.

3.2.5. HauteFrequenceAt : L'idée c'est de chercher le caractère le plus fréquent à une position donnée et ceci en découplant le texte en blocs de taille donnée en entrée et chercher le caractère le plus répété à une position p dans ces blocs et c'est donc ce dernier qui a la plus haute fréquence à la position p.

3.2.6. HauteFrequence: Elle permet de trouver les caractères les plus fréquents aux différentes positions dans le bloc de taille T. En faisant appel à la méthodeHauteFrequenceAt pour chaque position dans le bloc.

Utilisation : Dans certains chiffrements par bloc, où chaque caractère dans le clair est chiffré par une simple substitution monoalphabétique selon sa position, il est possible de faire une analyse de fréquence pour chaque position du bloc, comme dans le cas du système de Vigenère, Hill, Affine par bloc...etc.

3.2.7. XORParBloc: Cette fonctionnalité prend en entrée une clé et un texte, puis découpe le texte donné en entrée en blocs de taille égale à celle de la clé et effectue un XOR logique pour chaque bloc avec la clé. Elle rend en sortie les résultats du XOR des blocs.

Utilisation : Cette fonction elle pourra être appliquée pour les cryptogrammes obtenus avec un chiffrement utilisant l'opération XOR. Un cryptanalyste voulant appliquer un XOR aux blocs du cryptogramme, il donne en entrée à cette fonction une de ses clés candidates.

3.2.8. BonnePermutation: L'idée c'est de chercher la bonne clé de permutation d'une façon automatique, et ceci en générant toutes les clés possibles et trouver la bonne (attaque par force-brute). À chaque fois qu'une clé est générée avec la méthode **GenerationEtTest**, elle permute le texte avec cette clé de permutation puis elle calcule le score du résultat avec la méthode **Score** en utilisant l'ensemble des n-grammes les plus fréquents qui ont été calculés au préalable. Cet ensemble de n-grammes est calculé en utilisant la méthode **SetOfMostFrequentGrammes** qui prend comme paramètre un texte clair, la taille minimale des n-grammes, la taille maximale, ainsi que le nombre de n-grammes pour chaque taille, elle renvoie un ensemble contenant les n-grammes les plus fréquents dans le texte clair. À la fin, quand tous les scores des clés ont été calculés, la fonction renvoie la clé qui détient le plus grand score c'est à dire la clé de permutation dont son résultat contient le plus grand nombre de n-grammes fréquents.

Utilisation : Dans certains cas, le cryptanalyste n'aura pas d'autres solutions pour calculer une clé de permutation à part de tester toutes les clés possibles. Cette fonction lui permet automatiquement de lui renvoyer la bonne clé de permutation en utilisant les fréquences des n-grammes dans un texte choisi.

3.2.9. CalculSetClePermute : C'est le même problème que le précédent, mais cette fonctionnalité permet de faire une attaque à texte clair connu. Elle prend le message clair et le message chiffré et elle renvoie les clés de permutations candidates (si les caractères du bloc sont distincts elle renvoie qu'une). C'est un calcul très rapide puisque elle permute le seul bloc chiffré et renvoie les clés qui donnent un bloc identique au bloc clair passé en paramètre.

Utilisation: Comme l'attaque par force brute est très lente, l'attaquant tente par les moyens qu'il dispose, d'avoir une paire de message (clair/chiffré), puis fait appel à cette fonction qui lui renvoie un nombre négligeable de clés qui ont pu être utilisés pour permuter le message clair en chiffré (un nombre qui dépasse pas la taille de la clé).

3.2.10. Conversion de l'Unicode au ASCII UTF-8

Cette fonctionnalité (UnicodeToAscii) prend en entrée un fichier texte en Unicode et elle renvoie un fichier dans le même emplacement qui contient le même texte en ASCII UTF-8.

Utilisation : Le cryptanalyste utilise cette fonction car il existe des différents algorithmes de chiffrement qui utilisent la conversion en Unicode (exemple les mots de passe), alors le cryptanalyste a besoin de faire la transformation inverse et tester s'il y a des transformations ou non pour trouver le message clair.

3.2.11. La comparaison des ensembles des caractères des deux fichiers

Cette fonctionnalité (**EnsembleSymboleEgaux**) prend en entrée deux fichiers texte et elle renvoie une chaîne de caractères qui indique est-ce que ils ont le même ensemble des caractères et elle renvoie aussi le nombre des caractères de chaque fichier.

Utilisation : Le cryptanalyste utilise cette fonction dans le but à partir de deux fichiers textes il vérifie est-ce qu'ils ont les mêmes caractères, tel que le but est de tester est-ce qu'on reste dans le même domaine des caractères si on effectue un sur-chiffrement d'un chiffré donné.

3.2.12. L'analyse de fréquences des N-grammes

Cette fonctionnalité (Ngramme) permet de faire l'analyse de fréquences des N-grammes elle prend en entrée un fichier texte et trois entiers qui expriment selon le choix de l'utilisateur en mode : **glissant ou non glissant** et on peut effectuer l'analyse selon le choix soit tous **les caractères de texte ou juste les lettres ou juste les numéros** ou pour **les lettres et les numéros ensemble**. Et on aura en sortie un hashtable qui contient Les N-grammes les + fréquents associés à leurs fréquences et tous les N-grammes présents dans le texte avec leurs fréquences.

Utilisation : C'est une méthode de cryptanalyse consistant à examiner la fréquence des N-grammes employés dans un message chiffré. Cette méthode est utilisée pour décoder des messages chiffrés par substitution, on a fait l'appel à cette fonction pour réaliser la fonction qui détecte est-ce que la langue du texte donné est le français ou non et aussi pour réaliser une deuxième fonctionnalité qui consiste à trouver la bonne permutation (la clé).

Un petit exemple pour expliquer le mode glissant : pour N=2

Pour une chaîne de caractères ABCDEF, l'analyse des bi-grammes glissants donnera 1 fois AB, 1 fois BC, 1 fois CD, 1 fois DE et 1 fois EF

3.2.13. Longueurs des mots

Cette fonctionnalité (LongueurMots) prend en paramètre un fichier texte et on aura en sortie un hashtable qui contient les longueurs des mots et les fréquences associées.

Utilisation : l'utilité de cette fonctionnalité est d'effectuer l'analyse des fréquences des longueurs des mots pour un texte donné qui permet à un cryptanalyste de savoir comment le texte est découpé pour effectuer d'autre traitement pour avoir le clair.

3.2.14. Indice de coïncidence

Cette fonctionnalité (CoincidenceFinal) prend en entrée un fichier texte et on aura en sortie un float qui correspond à l'indice de coïncidence 'IC'

Utilisation : L'indice de coïncidence est une technique de cryptanalyse elle permet de savoir si un texte français a été chiffré avec un chiffre mono-alphabétique ou un chiffre poly-alphabétique en étudiant la probabilité de répétition des lettres du message chiffré (exemple : En français, l'indice de coïncidence vaut environ 0,0778).

Avec un IC aussi proche du français(0,0778), on peut faire l'hypothèse que ce cryptogramme est le résultat d'un chiffrement mono-alphabétique.

Avec l'IC aussi différent du français (0.0778), on peut déduire que le chiffre utilisé est poly-alphabétique.

3.2.15. La longueur de la clé probable avec l'indice de coïncidence (casser le chiffrement de Vigenère)

Cette fonctionnalité(LongueurC) elle prend en entrée un fichier texte et l'indice de coïncidence de la langue utilisée (exemple la langue française 0.078) on aura en sortie un array liste qui contient les moyennes de IC de chaque longueur de clé et la longueur de clé probable.

Utilisation : l'utilité de cette fonctionnalité c'est lors de la vérification automatique d'un déchiffrement, en particulier lors d'une recherche exhaustive de toutes les clés (exemple pour un chiffré d'un texte français). Les clés incorrectes fournissent un indice très proche de l'indice de données aléatoires uniformément distribuées(0,038). Une clé donnant un indice plus élevé a donc des chances d'être la bonne.

Exemple de fonctionnement :

Soit le message suivant, chiffré avec Vigenère

PERTQ UDCDJ XESCW MPNLV MIQDI ZTQFV XAKLR PICCP QSHZY DNCPW
EAJWS ZGCLM QNRDE OHCGE ZTQZY HELEW AUQFR OICWH QMYRR UFGBY
QSEPV NEQCS EEQWE EAGDS ZDCWE OHYDW QERLM FTCCQ UNCPP QSKPY
FEQOI OHGPR EERWI EFSDM XSYGE UELEH USNLV GPMFV EIVXS USJPW HIEYS
NLCDW MCRTZ MICYX MNMFZ QASLZ QCJPY DSTTK ZEPZR ECMYW OICYG

UESIU GIRCE UTYTI ZTJPW HIEYI ETYYH USOFI XESCW HOGDM ZSNLV QSQPY
JSCAV QSQLM QNRLP QSRLM XLCCG AMKPG QLYLY DAGEH GERCI RAGEI
ZNMGI YBPP

On va considérer les sous-chaînes obtenues en prenant les lettres à intervalle donné:

Intervalle de 1: PERTQ UDCDJ XESCW MPNLV ... (texte original)

Intervalle de 2: PRQDD XSWPL ... et ETUCJ ECMNV ...

Intervalle de 3: PTDJS MLIIQ ..., EQCXC PVQZF... et RUDEW NMDTV

... .

On calcule ensuite les IC pour toutes ces sous-chaînes.

Intervalle	Indice de coïncidence
------------	-----------------------

1=>	0.0456107
-----	-----------

2=>	0.0476954, 0.0443098
-----	----------------------

3=>	0.044249, 0.0494469, 0.0426771
-----	--------------------------------

4=>	0.0465839, 0.0453894, 0.0449116, 0.0425227
-----	--

5=>	0.0799704, 0.0925583, 0.0836727, 0.0795282, 0.0684932
-----	---

6=>	0.0512956, 0.0407192, 0.0371585, 0.0382514, 0.0661202, 0.0431694
-----	--

On remarque que la moyenne de l'indice de coïncidence quand l'intervalle est de 5, l'IC correspond plus ou moins avec l'IC caractéristique du français (en tout cas, c'est cette ligne qui s'approche le plus de 0.074, les autres lignes étant plutôt proches de 0.038). La longueur de la clef utilisée est donc probablement 5

3.2.16. Recherche les lignes différentes entre deux fichiers

Cette fonctionnalité(LigneCompar) prend en paramètre deux fichiers et on aura en sortie un fichier et elle fait appel à trois fonctions :

- ✓ La fonctionnalité (fileToLines) qui prend en paramètre un fichier et on aura en sortie une liste des lignes
- ✓ La fonctionnalité (differentLines) prend en entrée deux listes des lignes et on aura en sortie un fichier qui contient les numéros des lignes et les lignes qui sont présentes dans le fichier1 mais absentes dans le fichier deux
- ✓ La fonctionnalité (sameLines) prend en entrée deux listes des lignes et on aura en sortie un fichier qui contient les numéros des lignes et les lignes communs au fichier1 et au fichier 2.

Utilisation : c'est pour comparer deux fichiers rapidement pour avoir une idée de la façon de chiffrement.

3.2.17. Tester si un texte donné est écrit en langue française

Cette fonctionnalité prend en paramètre un fichier texte et elle fait appel à la fonction N-gram et on aura en sortie un booléen qui indique si « vrai » le texte est écrit en la langue française sinon elle renvoie faux.

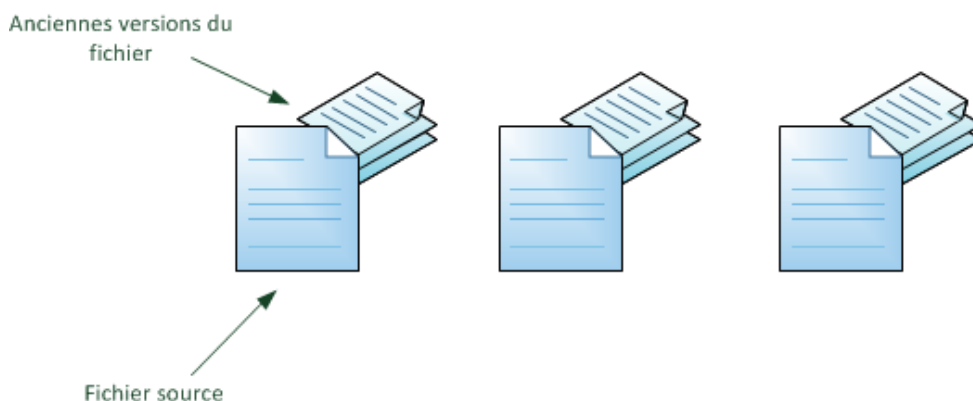
Utilisation : détecter est ce qu'un texte donné est écrit en la langue française en comparant le bi-gramme le plu fréquent de ce texte donné en paramètre avec les plus fréquents dans la langue française(ES, DE, LE, EN).

4. Gestionnaire de version

4.1 Présentation du gestionnaire de version

Les logiciels de gestion de versions sont utilisés principalement par les développeurs ; ce sont donc bel et bien des outils pour *programmeur*. En effet, ils sont quasi exclusivement utilisés pour gérer des codes sources, car ils sont capables de suivre l'évolution d'un fichier texte ligne de code par ligne de code. Ces logiciels sont fortement conseillés pour gérer un projet informatique.

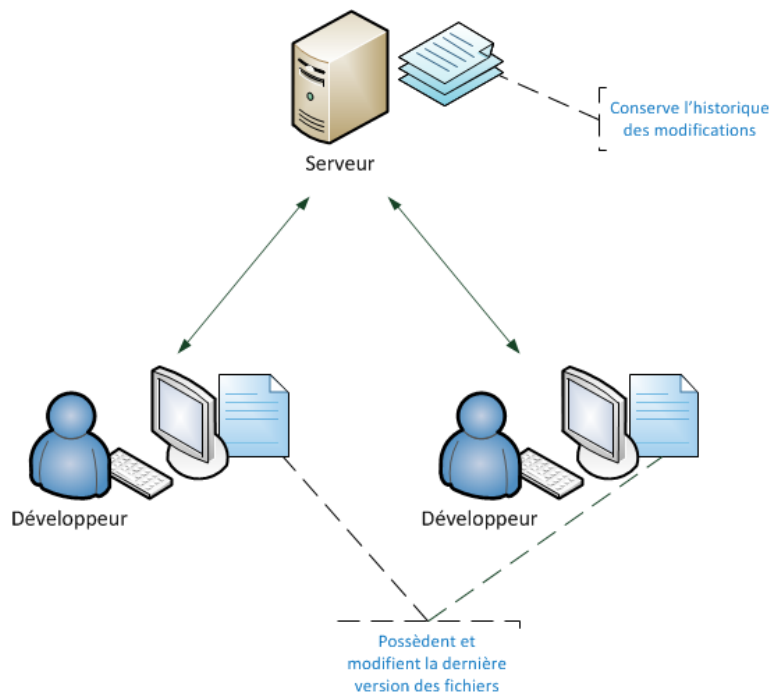
Ces outils suivent l'évolution des fichiers source et gardent les anciennes versions de chacun d'eux.



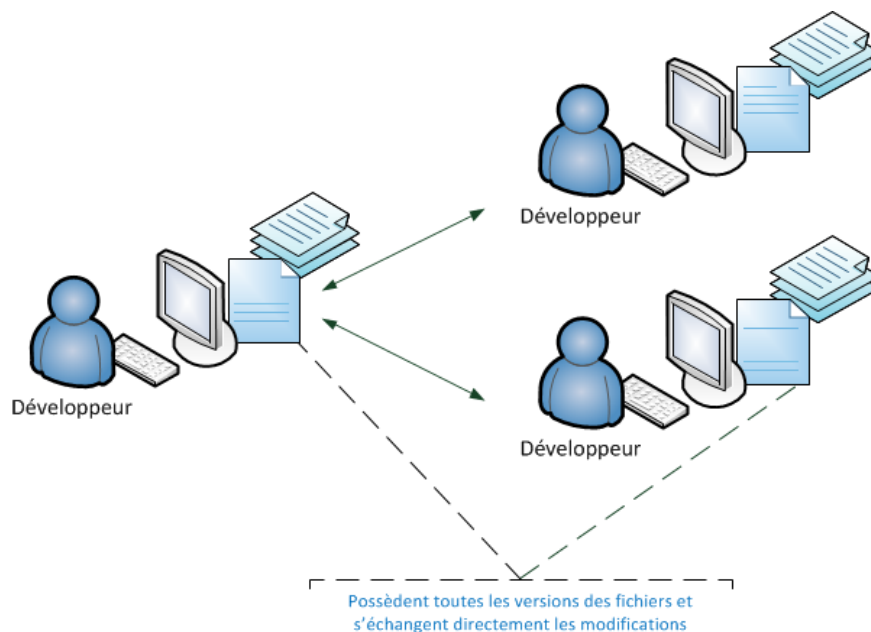
4.2. Types principaux de logiciels de gestion

Il existe deux types principaux de logiciels de gestion de versions :

- **Les logiciels centralisés** : un serveur conserve les anciennes versions des fichiers et les développeurs s'y connectent pour prendre connaissance des fichiers qui ont été modifiés par d'autres personnes et pour y envoyer leurs modifications.



- **Les logiciels distribués** : il n'y a pas de serveur, chacun possède l'historique de l'évolution de chacun des fichiers. Les développeurs se transmettent directement entre eux les modifications, à la façon du *Peer-to-Peer*.



5. Description du plan de travail

5.1. Problèmes rencontrés : il y a eu quelques petits soucis pour déterminer la durée de Certaines tâches ainsi que leurs répartitions, comme certaines fonctions sont fortement reliées entre elles mais elles sont effectuées séparément par chacun de nous Il fallait alors mettre en

commun ce qui a causé quelques problèmes. Nous avons aussi eu quelques difficultés à programmer certaines fonctionnalités ça nous a pris un peu plus de temps que prévu.

5.2. La répartition des tâches

Pour atteindre les objectifs fixés, nous développons un plan d'action. Notre travail de groupe, consiste en une étude des différentes fonctionnalités qu'un cryptanalyste peut utiliser pour casser n'importe qu'elle crypto-système et les implémenter sous forme d'une boîte à outil avec une interface graphique et un guide d'utilisation de cette boîte.

Le travail est partagé comme suit :

HARROUCHE Ghania:

J'ai fait de diverses recherches sur la cryptographie et la cryptanalyse en général afin de pouvoir tirer des idées sur les fonctionnalités qui seront utiles à programmer, ainsi que la rédaction du rapport et la préparation de la présentation en faisant des slides.

Et j'ai réalisé quelques méthodes qui permettent d'attaquer un cryptosystème qui sont décrites ci-dessous:

La fonction du modulo: elle consiste à chercher le plus grand byte contenu dans un message

La substitution: cette fonctionnalité permet de remplacer un caractère par un autre dans un (fichier ou un tableau). Elle prend en entrée le texte et le caractère que l'on veut remplacer avec le caractère par lequel on le remplace et elle renvoie le texte dans lequel la substitution a été faite.

SubstitutionParPositionDansLeBloc: L'idée c'est de remplacer un caractère à une position donnée dans un bloc. Elle prend en entrée un texte la taille des blocs le caractère à remplacer avec le caractère par lequel on désire le remplacer. Elle renvoie un texte dans lequel la substitution a été faite.

CalculeTaille: Elle permet de calculer la taille de la clé si le chiffrement utilisé est un chiffrement par bloc. Elle prend en entrée un texte et la taille des blocs identiques à chercher puis elle renvoie la taille de la clé.

HauteFrequenceAt: cette fonctionnalité permet de chercher le caractère le plus fréquent à une position donnée dans le bloc. Elle prend en entrée un texte la taille des blocs et une position p et elle retourne le caractère le plus à cette position p.

HauteFrequence: elle permet de trouver les caractères les plus fréquents aux différentes positions dans les blocs. Elle prend en entrée un texte la taille des blocs et elle retourne les caractères les plus fréquents.

XORParBloc: Cette fonctionnalité prend en entrée un texte et une clé de taille t puis elle découpe le texte en blocs de taille t et elle rend en sortie le résultat du XOR logique de ces blocs avec la clé donnée en entrée.

Bonne permutation: elle permet de trouver la bonne clé de permutation d'une manière automatique en faisant appel à différentes classes: SetOfMostFrequentGrammes qui permet de calculer les n_grmmes les plus fréquents, la méthode GenerationEtTest qui génère les différentes clés candidates et génère les fichiers associés à ces clés puis calcule leurs scores grâce à la fonction Score et elle renvoie la clé qui détient le plus grand score et c'est cette dernière qui est la bonne clé de permutation.

CalculSetClePermute: Quand l'attaquant dispose d'un couple clair/chiffré il fait appel à cette fonction pour renvoyer l'ensemble de clés qui ont pu être utilisées pour la permutation.

SAIDANI Meriem

Etudier et faire des recherches sur les différentes attaques classiques et s'inspirer de ces attaques pour trouver les différentes fonctionnalités pour la boîte à outil et les implémenter, et rédiger le rapport et effectuer une partie des slides pour présenter le travail final.

Les différentes fonctionnalités réalisées :

Conversion de l'Unicode au ASCII UTF-8 : Cette fonctionnalité (UnicodeToAscii) prend en entrée un fichier texte en Unicode et elle renvoie un fichier qui contient le même texte en ASCII UTF-8.

La comparaison des ensembles des caractères des deux fichiers : Cette fonctionnalité (EnsembleSymboleEgaux) permet à un cryptanalyste de vérifier si ce qu'on reste dans le même domaine des caractères si on effectue un sur-chiffrement d'un chiffré donné.

L'analyse de fréquences des N-grammes : Cette fonctionnalité (Ngramme) permet d'examiner la fréquence des N-grammes employés dans un message chiffré. Cette méthode est utilisée pour décoder des messages chiffrés par substitution, on a fait l'appel à cette fonction pour réaliser la fonction qui détecte si ce que la langue du texte donné est le français ou non et aussi pour réaliser une deuxième fonctionnalité qui consiste à trouver la bonne permutation (la clé).

Longueurs des mots : Cette fonctionnalité (LongueurMots) permet d'effectuer l'analyse des fréquences des longueurs des mots pour un texte donné qui permet à un cryptanalyste de savoir comment le texte est découpé pour effectuer d'autres traitements pour avoir le clair.

Indice de coïncidence : Cette fonctionnalité (CoincidenceFinal) permet de savoir si un texte français a été chiffré avec un chiffre mono-alphabétique ou un chiffre poly-alphabétique en étudiant la probabilité de répétition des lettres du message.

La longueur de la clé probable avec l'indice de coïncidence (casser le chiffrement de Vigenère) : Cette fonctionnalité(LongeurC) permet d'effectuer la vérification automatique d'un déchiffrement, en particulier lors d'une recherche exhaustive de toutes les clés (exemple pour un chiffré d'un texte français). Les clés incorrectes fournissent un indice très proche de l'indice de données aléatoires uniformément distribuées(0,038). Une clé donnant un indice plus élevé a donc des chances d'être la bonne.

Recherche les lignes différentes entre deux fichiers : Cette fonctionnalité(LigneCompar) permet de comparer deux fichiers rapidement pour avoir une idée de la façon de chiffrement.

Tester si un texte donné est écrit en langue française : permet de détecter est ce qu'un texte donné est écrit en la langue française.

.

SOUALAH Youba

Mise en place d'un gestionnaire de version (Github) qu'on utilise pour mettre à jour nos code et que nous consultant pour voir l'avancement du projet, création du mode d'emploi de l'application sous forme de page html avec des liens et des images, implémentation de quelque différentes fonctionnalités pour la boîte à outil qui vise à essayer de faciliter au cryptanalyste le déchiffrement d'un message , rédiger le rapport et effectuer une partie de la diaporama pour le travail final.

Calcule du Xor : La fonction effectue la lecture d'un caractère et la récupération d'un fichier dans le workspace, puis utilise un Xor entre le caractère et chaque caractère du fichier sélectionné :

Exemple : Saisissez un caractère :

a

Saisissez une chaîne :

E »'é »'t'(ret'(-« '(« '(é »)(ç'(-ç_(-àè(ç-è(-'

Vous avez saisi le caractère : a en binaire c'est : 1100001

1100001 XOR 1100101 : 0000100

1100001 XOR 1110010 : 0010011

1100001 XOR 1100101 : 0000100

1100001 XOR 1110010 : 0010011

1100001 XOR 1101010 : 0001011

1100001 XOR 1100101 : 0000100

Découpage en bloc : La fonction qu'on a réalisé consiste à lire un fichier texte en entrant un chiffre qui est le nombre de caractère que doit contenir un bloc puis le découper en petit bloc de taille précise et le stocker dans un fichier texte.

Analyse de fréquence : elle lit un fichier texte puis rend la fréquence (nombre de caractère existant dans le texte) de chaque caractère sous forme d'une matrice.

MAMOUNI Walid

Ma principale tâche dans ce projet a été de réaliser une interface graphique en langage Java. Cette interface a pour but de faciliter l'utilisation des différentes fonctionnalités développées par mes collègues.

Les environnements utilisés sont les suivants :

- a) Eclipse (en intégrant le plugin "WindowsBuilder")
- b) Netbeans
- c) JRE8 Et JDK8 (dernières versions)

J'ai regroupé toutes les fonctionnalités dans trois classes différentes :

classeRegroupef.java

classeRegroupef2.java

bonnePermutation.java

La classe principale est : principale.java

Je me suis occupé notamment de la génération de la javaDoc.

Et de la partie « La boîte à outils » du rapport.

Conclusion :

L'un des problèmes fondamentaux de la cryptographie est d'évaluer la sécurité des systèmes de chiffrement avec les techniques puissantes, il ne faut pas se reposer sur une sécurité jugée bonne à un moment donné, car de nombreuses attaques génériques sont possibles qui mènent à casser cette confidentialité, C'est pourquoi, la construction des algorithmes de chiffrement doit s'appuyer sur des problèmes mathématiques difficiles.

Ce projet nous a permis d'approfondir nos connaissances sur le JAVA, sur les différentes attaques génériques, sur cryptographie et la cryptanalyse en général. Comme appris à faire des applications plus attrayantes et plus orientées

Bibliographie

- [1] http://www-igm.univ-mlv.fr/~desar/Cours/M1-1_Codage_Cryptographie/chap3.pdf
- [2] http://laure.gonnord.org/pro/teaching/MIF30/YG_CoursCrypto_MIF30.pdf
- [3] http://www.univ-orleans.fr/lifo/Members/Yohan.Boichut/Enseignement/Securite/Securite_2_codes_et_chiffres_2008_2009.pdf
- [4] http://www.hypo-theses.com/docs/user62_TPE_cryptographie.pdf
- [5] docnum.univ-lorraine.fr/public/UPV-M/Theses/2007/LeHoai.Minh.SMZ0754.pdf
- [6] <http://www.bibmath.net/crypto/index.php?action=affiche&quoi=ancienne/transposition>
- [7] http://www.unilim.fr/pages_perso/christophe.chabot/doc/cours5.pdf
- [8] M. Matsui, "The first experimental cryptanalysis of the data encryption standard," in *CRYPTO*, pp. 1–11, 1994.
- [9] http://www-igm.univ-mlv.fr/~desar/Cours/M1-1_Codage_Cryptographie/chap3.pdf
- [10] <http://www.montefiore.ulg.ac.be/~herbiet/crypto/02-Cryptographie%20classique.pdf>
- [11] <http://www.uqtr.ca/~delisle/Crypto/cryptanalyse>
- [12] <http://www.matthieuamiguet.ch/media/documents/MA-CRYPTO-05-Cryptanalyse.pdf>
- [13] <http://deptinfo.unice.fr/twiki/pub/Linfo/PlanningDesSoutenances20032004/blanc-degeorges.pdf>
- [14] <http://info-rital.developpez.com/tutoriel/java>