REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITE AMAR TELIDJI
LAGHOUAT

FACULTE DES SCIENCES ET D'INGENIERIE
DEPARTEMENT DE GENIE INFORMATIQUE

PROJET DE FIN D'ETUDES
Pour l'Obtention Du Diplôme

D'INGENIEUR D'ETAT EN INFORMATIQUE

Option : intelligence artificielle

Thème

Application des algorithmes génétiques à la cryptographie (OTL modifié)

Présenté par :

- Ouled Djedid Lakhdar
- Benhaoua Kamel

Encadré par :

Encadreur :
Mr : Lagraa Nasr Eddine

Co-encadreur :
Mr :Boukhalfa kamel

N° d'ordre : …….. **/2005-PFE/DGI**

Webographie

[1]: http://www.zurich.ibm.com/Technology/Security/extern/internet/white-paper.html .

[2]: http://www.dmi.ens.fr/equipes/grecc/crypto/intro/intro.html .

[3]: http://www.securiteinfo.com .

[4]: http://www.menard.bourgogne.net.com/tipe98/algom/IDEA.htm#idea.

[5]: http://www.carefour.usherbrooke.ca/resultats2004/rsa_berube+5/rsa/rsa/introduction.html.

[6]: http://www.commentcamarche.com.

Bibliographie

[8]:David Goldberg .(1994).Les algorithms genetiques .Edition Wesley.

[9]:Vincent Magnin.Algorithmes Evolutionnaires et Algorithmes Génétiques.

[10] : Omary.F, Lbekkouri.A et Tragha.A"Extension des applications des algorithmes évolutionistes " .Rapport interne N° 7 / 01/ 2004 .Département de mathématiques et informatique Faculté des sciences -Rabat.

[doc1] :A.ALI PACHA '' Sécurité des donnée par l'IDEA ''. Deuxième journées d'Informatique pour l'Entreprise .

[doc2] : Mémoire ''problème du voyageur de commerce a l'aide des algorithmes génétiques''

[doc3]:Emeric Jioan .Introduction à la cryptographie. Université de la réunion .

[doc4]:Christian Paquin.Les codes correcteurs quantique et leurs applications cryptographique. Faculté des arts et des sciences / Département d'informatique et recherche opérationnelle.

Remerciements

Après ces cinq , longues années qui viennent de s'écouler , jalonnées de bon ou de mauvais moments , voilà que vient la fin de mon cursus universitaire ou peut-être devrais-je dire le début d'une ère nouvelle pour moi et pour mes camarades de promotion .

Je partirais donc pour une nouvelle vie mais non sans avoir présenté mon profond respect et ma vive gratitude à tous ceux et celles qui m'ont accompagné tout au long de ces moments souvent éprouvant :

- A mon encadreur M^{r} Lagraa Nacer Eddine , pour sa disponibilité , son sens de l'écoute et sa patience à mon égard .
- A mes professeurs : M^{r} Ouinten Youcef , M^{r} Tahari abedlkarim , M^{r} Djoudi Mohamed , M^{r} Mokhtari , M^{r} Belabassi , M^{r} Derradji , M^{r} yagoubi ,M^{me}Kerrouche , M^{me} Guibage , M^{me} Bouzouade pour leurs aide précieuse et leurs conseils sans prix .
- Au président et aux membre du jury .

Enfin , je tiens a dire a tous ce qui mon aider de prés ou de loin par un mot gentil , un geste , un sourire , m'avez encouragé pour arriver jusqu'au bout sans encombres , je dis tout simplement : Merci .

Ouled Jedid Lakhdar

Dédicace

- à la mémoire de mon père (partis trop tôt pour un monde meilleur) qui auraient été tellement fière de moi.
- à ma mère pour le soutien et la compréhension qu'elle ma toujours apportés malgré vents et marées.
- à mes frères.
- à mes sœurs.
- à mes neveux et nièce..
- A mes oncles et tantes.
- A tous mes cousins et cousines paternels et maternels.
- A tous ceux et celles qui me connaissent et m'apprécient.
- A tous mes amis .

Je dédie ce modeste travail du fond de mon cœur.

Ouled Jedid Lakhdar

Remerciements

Après ces cinq , longues années qui viennent de s'écouler , jalonnées de bon ou de mauvais moments , voilà que vient la fin de mon cursus universitaire ou peut-être devrais-je dire le début d'une ère nouvelle pour moi et pour mes camarades de promotion .

Je partirais donc pour une nouvelle vie mais non sans avoir présenté mon profond respect et ma vive gratitude à tous ceux et celles qui m'ont accompagné tout au long de ces moments souvent éprouvant :

- A mon encadreur M^{r} Lagraa Nacer Eddine , pour sa disponibilité , son sens de l'écoute et sa patience à mon égard .
- A mes professeurs : M^{r} Ouinten Youcef , M^{r} Tahari abedlkarim , M^{r} Djoudi Mohamed , M^{r} Mokhtari , M^{r} Belabassi , M^{r} Derradji , M^{r} yagoubi ,M^{me}Kerrouche , M^{me} Guibage , M^{me} Bouzouade pour leurs aide précieuse et leurs conseils sans prix .
- Au président et aux membre du jury .
- Je tiens a remercier particulièrement mes grands-parents (la famille Bourezg) pour avoir supporté tout ce temps mes sautes d'humeur et parfois mon viascibilité sans fondement .
- Merci , aussi à mes parents pour leurs soutiens moral et matériel pour que je puisse mener à bien ce modeste travail .

Enfin , je tiens a dire a tous ce qui mon aider de prés ou de loin par un mot gentil , un geste , un sourire , m'avez encouragé pour arriver jusqu'au bout sans encombres , je dis tout simplement : Merci .

Benhaoua Mohamed Kamel

Dédicace

- à mon père Ilies , en espérant qu'il sera notre guide par sa sagesse et sa droiture .
- à ma mère pour le soutien et la compréhension qu'elle ma toujours apportés malgré vents et marées.
- à mes frères : Abdelkader, oualid , Habib , Mourad .
- à mes sœurs : Fouzia , Imene .
- à mes neveux et nièce
- à Maza (que Dieu me la garde) .
- à mes grands-parents de Laghouat (que Dieu me les garde).
- à la mémoire de mon grand père hadj Abdelkader et ma grand mère Ma-Tata (partis trop tôt pour un monde meilleur) qui auraient été tellement fière de moi.
- A mes oncles et tantes de Mascara, Bel Abbés, Oran, El-Amiria, Laghouat, Touggourt, Chicago.
- A Badr-Eddine et sa femme Aïcha , Saad et sa femme Fatima , gharbi abdallâh, saïdat Djamel , Faïza , Fatima.
- A tous mes cousins et cousines paternels et maternels.
- Au famille Benhaoua, Bourezg, Moubarik, Zekri, Méliani, Derkaoua, gharbi.
- A mes cousin Sofiane, abdeladhim.
- A tous ceux et celles qui me connaissent et m'apprécient.

Je dédie ce modeste travail du fond de mon cœur.

Benhaoua Mohamed Kamel

TABLE DES MATIERES

Chapitre I La Sécurité informatique

Chapitre II La cryptographie

Chapitre III Les *algorithmes génétiques.*

Chapitre IV La cryptographie évolutionniste

Chapitre V Implémentation et comparaison

Chapitre I :

La sécurité informatique

I.1) Introduction :

L'informatique et les réseaux de télécommunication sont devenus des outils de travail indispensables pour les tâches critiques de la vie professionnelle, ce sont des outils vitaux pour l'enseignement, la recherche et le pilotage de nos établissements. Nous devons en assurer le bon fonctionnement, pour garantir la liberté de communiquer.

Jusqu'au début des années 80, la centralisation des moyens informatiques (quelques gros serveurs par établissement par exemple) et la quasi absence de communication avec l'extérieur, permettaient de facilement garantir la sécurité grâce à une administration (équipe) centralisée bien identifiée. Les temps ont changé, le Système d'information (SI) d'un établissement sont aujourd'hui réparti, partout et inter connecté entre eux par le réseau. Le bon fonctionnement de cette informatique distribuée, peut être perturbée.

En effet, si la généralisation de la connexion à l'Internet de nos établissements et l'utilisation de plus en plus intense des systèmes communicants, offrent des possibilités nouvelles et prometteuses, elles introduisent également un certain nombre de risques. Ces risques , il faut prendre conscience, en mesurer les conséquences éventuelles, et en connaissance de cause prendre les mesures adéquates. C'est ce qui introduit la notion de sécurité informatique. La *sécurité informatique*, d'une manière générale, consiste à assurer que les ressources matérielles ou logicielles d'une organisation seront uniquement utilisées dans le cadre où il est prévu qu'elles le soient [1].

I.2) Les objectifs de la sécurité informatique:

La sécurité informatique a plusieurs objectifs, bien sûr liés aux types de menaces ainsi qu'aux types de ressources, etc. ... . Néanmoins, l'objectif principal de la sécurité informatique est d'empêcher :

- la divulgation non autorisée de données.
- la modification non autorisée de données.
- l'utilisation non autorisée de ressources réseau ou informatique de façon générale.

I.3) Services principaux de la sécurité informatique :

Pour remédier aux failles et pour contrer les attaques, la sécurité se base sur un certain nombre de services qui permettent de mettre en place une réponse appropriée à chaque menace. A ce niveau, aucune technique n'est encore envisagée; il ne s'agit que d'un niveau d'abstraction visant à obtenir une granularité minimale pour déployer une politique de sécurité de façon optimale.

Les attaques peuvent à première vue être classées en 2 grandes catégories :

- *Les attaques passives* : Consistent à écouter sans modifier les données ou le fonctionnement du réseau. Elles sont généralement indétectables mais une prévention est possible.
- *Les attaques actives* : Consistent à modifier des données ou des messages, à s'introduire dans des équipements réseau ou à perturber le bon fonctionnement de ce réseau. De plus, il n'y a généralement pas de prévention possible pour ces attaques, bien qu'elles soient détectables (permettant ainsi une réponse adéquate).

Les principaux services qui réalisent la protection contre les attaques sont :

- *confidentialité* : Assurer que seules les personnes autorisées ont accès aux ressources.
- *authentification* : Permet d'assurer l'origine des données (l'identification).
- *intégrité* : c'est-à-dire garantir que les données de l'émetteur n'on pas été modifié frauduleusement par un tiers durant la communication (c'est la fonction de hachage qui garantie l'intégrité des messages) .
- *contrôle d'accès :* Qui permet de limiter l'accès aux données, EX : accès aux serveurs limité aux personnes autorisées (mot de passe, par exemple).
- *non répudiation* : avec preuve d'émission ou avec preuve de réception.

Parmi les aspects importants qui répondent au domaine de la sécurité informatique on va illustrer dans tous notre travail le domaine qui permet d'échanger des données tout en préservant leur confidentialité c'est le domaine de *la cryptographie* .

Chapitre II :

La Cryptographie

II.1) Introduction :

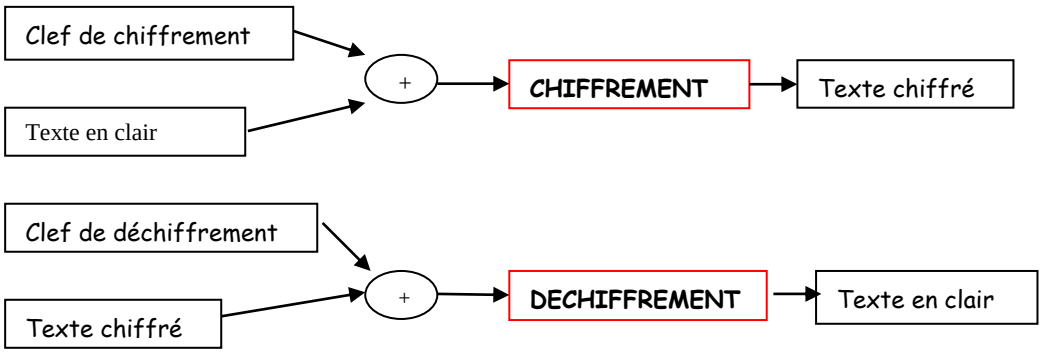
La cryptographie vient du grec kryptos qui veut dire cacher et de graphein qui signifie écrire , étant donné qu'une lettre ou une communication peuvent être interceptées par un individu mal intentionné, il est prudent de rendre le message incompréhensible .

La cryptographie est donc un ensemble de techniques permettant de protéger une communication au moyen d'un code secret.

Depuis Jules César, qui a été sans doute le premier à l'utiliser pour communiquer avec ses troupes, jusqu'à l'armée allemande qui s'est servie de machines électromécaniques lors de la deuxième guerre mondiale, la cryptographie a fait d'énormes progrès avec l'arrivée de l'informatique. Car la cryptographie se sert de la puissance des ordinateurs et des progrès mathématiques pour rendre tout message incompréhensible par un tiers.

La cryptographie traditionnelle (ancienne) est l'étude des méthodes permettant de transmettre des données de manière confidentielle. Afin de protéger un message, on lui applique une transformation qui le rend incompréhensible ; c'est ce qu'on appelle *le chiffrement*, qui, à partir d'un texte en clair, donne un texte chiffré ou cryptogramme. Inversement, *le déchiffrement* est l'action qui permet de reconstruire le texte en clair à partir du texte chiffré (**figure II.1**). Dans la cryptographie moderne, les transformations en question sont des fonctions mathématiques, appelées *algorithmes cryptographiques*, qui dépendent d'un paramètre appelé *clef* (**figure II.2).**

FigureII.1 : *Le principe de base de la cryptographie ancienne*



FigureII.2 : *Le principe de base de la cryptographie moderne*

Aujourd'hui avec l'essor d'Internet et des communications, l'usage de la cryptographie est devenu incontournable. Elle est principalement utilisée par les militaires pour assurer la confidentialité de leurs communications. Et même si les militaires utilisent toujours les techniques les plus modernes et les secrètes (symétrique ou asymétrique) il est important avant d'aborder ces techniques de présenter quelques méthodes anciennes de cryptographie [2].

II.2) La cryptographie ancienne :

Les origines de la cryptographie remontent à l'antiquité. On verra dans cette partie, à travers des exemples historiques, comment l'évolution des problèmes de cryptographie a abouti aux techniques modernes , en raison du nombre important de méthodes anciennes, nous nous limiterons aux méthodes suivantes [3] :

II.2.1) chiffrement par permutation :

- **le scytale** : utilise un ruban et un bâton appelé scytale .le ruban était enroulé autour du bâton , le message était écrit en colonne verticale le long du bâton .ainsi il fallait avoir un bâton de même diamètre pour retrouver le message à partir du ruban déroulé. dans ce cas, la clé de chiffrement est le scytale.

- **Chiffrement par transposition** :

  - **cryptage** : le message en clair **M** est écrit en **k** colonnes de **d** lignes tel que **M = d * k** pour former le message codé **C**. La clé de chiffrement dans ce cas est **d** (**figure II.3**) .

  - **décryptage** : connaissant la clé de déchiffrement **k = n/d**. On retrouve **M** à partir de **C** en écrivant le message en lignes de taille **k** et en lisant les colonnes.

    - **inconvénient** : ce chiffrement est facile à casser puisqu'il suffit d'essayer toutes les clés possibles (entier d inférieur à la taille du message), et de vérifier à chaque fois si le message obtenu a un sens.

  - **exemple** : **d** = 4 (nombre de lettres par colonnes)

    k = 6 (nombre de colonnes)

    n = 24 (taille du message)

    La clé de chiffrement est **d**.

Le message en clair est : M = voici le message secret

V	I		S		R
O		M	A	S	E
I	L	E	G	E	T
C	E	S	E	C	

Le message code est : C = vi s ro maseilegetcesec

II.2.2) chiffrement par substitution :

La technique de substitution consiste à changer les lettres suivant une règle précise.

- **Chiffrement de césar** :

  Il remplaçait chaque lettre du texte par celle qui se trouve trois places plus loin dans l'alphabet mis en cercle : A devient D , B devient E etc. il suffisait de supprimer les espaces entre les mots pour qu'il forme une suite de lettre incompréhensible. Dans ce codage, la clef était **k** = 3.

- **cryptage** : $c_i := m_i + k$ [26]

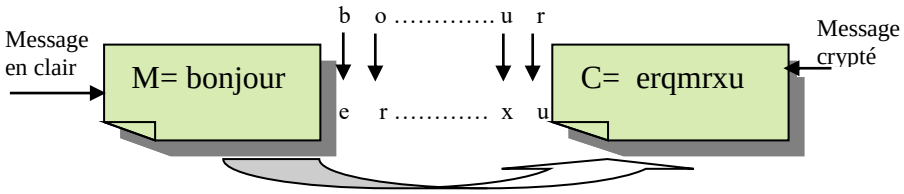
  c_i **:** c'est la position des caractères du texte crypté .

  m_i **:** c'est la position des caractères du texte en clair .

  k : représente la clé .jules césar a pris k = 3 .

- **décryptage** : $m_i = c_i - k$.

- **exemple** : avec k=3 .



- **inconvénient :** L'ennui d'un tel procédé est qu'un cryptanalyste peut retrouver le texte de départ en étudiant la statistique du texte crypté et ainsi retrouver le texte de départ . Ce type de codage peut être cassé en quelques secondes par un ordinateur actuel.

Chiffrement par substitution mono alphabétique : on remplace chaque lettre du message par la lettre correspondante dans une bijection µ avec un autre alphabet. le décryptage se fait par la bijection réciproque.

- **cryptage** : $c_i : = \mu(m_i)$

- **décryptage** : $m_i : = 1/\mu\ (c_i)$

- **inconvénient** : la méthode employée pour casser ce type de code a été découverte au IX ^{eme} siècle par le savant arabe AL-KINDI.il utilise les fréquences d'apparition des lettres des langages vivants .On retrouve ainsi facilement certaines lettres du message, et le reste se fait par tâtonnement en essayant de deviner les mots selon le contexte de l'information déjà obtenue.

- **Chiffrement par substitution poly alphabétique :** on remplace chaque lettre du message par une ou plusieurs lettres correspondantes dans un autre alphabet .le décryptage se fait par la fonction réciproque.

  - **inconvénient** : possible a casser avec des moyens statistiques plus élaborés.

- **Le code de Vigenére :** ce type de chiffrement est un raffinement du chiffre de césar, car l'alphabet de substitution change avec la place de la lettre à crypter dans le message et dépend d'un mot clé.

  La clé est un mot t_0 , t_1…… t_n . On note M=m_1 m_2. Et C=c_1c_2.

  - **cryptage** : $C_i = M_i + t_{i-l}[n] - 1$ [26].
  - **décryptage** : $M_i = C_i - t_{i-l}[n] + 1$ [26].

II.3) la cryptographie moderne (symétrique, asymétrique) :

Si le but traditionnel de la cryptographie est d'élaborer des méthodes permettant de transmettre des données de manière confidentielle, la cryptographie moderne s'attaque en fait plus généralement aux problèmes de sécurité des communications. Pour cela, on utilise un certain nombre de mécanismes basés sur des **algorithmes cryptographiques**.

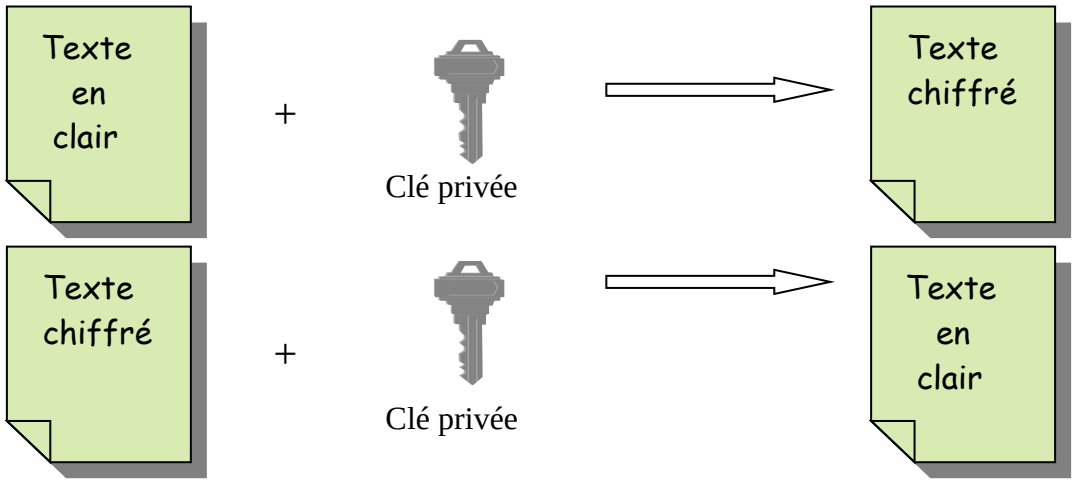
La sécurité des données chiffrées dépend des éléments suivants :

- La robustesse de l'algorithme cryptographique(difficile a casser).
- La confidentialité de la **clé**.

Il existe deux grandes familles d'algorithmes cryptographiques à base de clés : les algorithmes à **clé privée** ou algorithmes symétriques, et les algorithmes à **clé publique** ou algorithmes asymétriques.

II.4) Chiffrement symétrique ou à clé privée :

Dans la cryptographie conventionnelle, les clés de chiffrement et de déchiffrement sont identiques : c'est la clé privée , qui doit être connue exclusivement par les communicants. Le procédé de chiffrement est dit symétrique. On parle de cryptographie symétrique lorsque deux ou plusieurs personnes utilisent une même clé pour crypter et décrypter des messages (**figure II.3**).



FigureII.3 : *Principe du chiffrement symétrique*

- **Les inconvénients du chiffrement symétrique :**

Le principal inconvénient de ce système est le partage de cette clé unique entre les différentes personnes : Comment envoyer à tout le monde et de façon sécurisée cette clé unique qui permet de crypter et décrypter ?

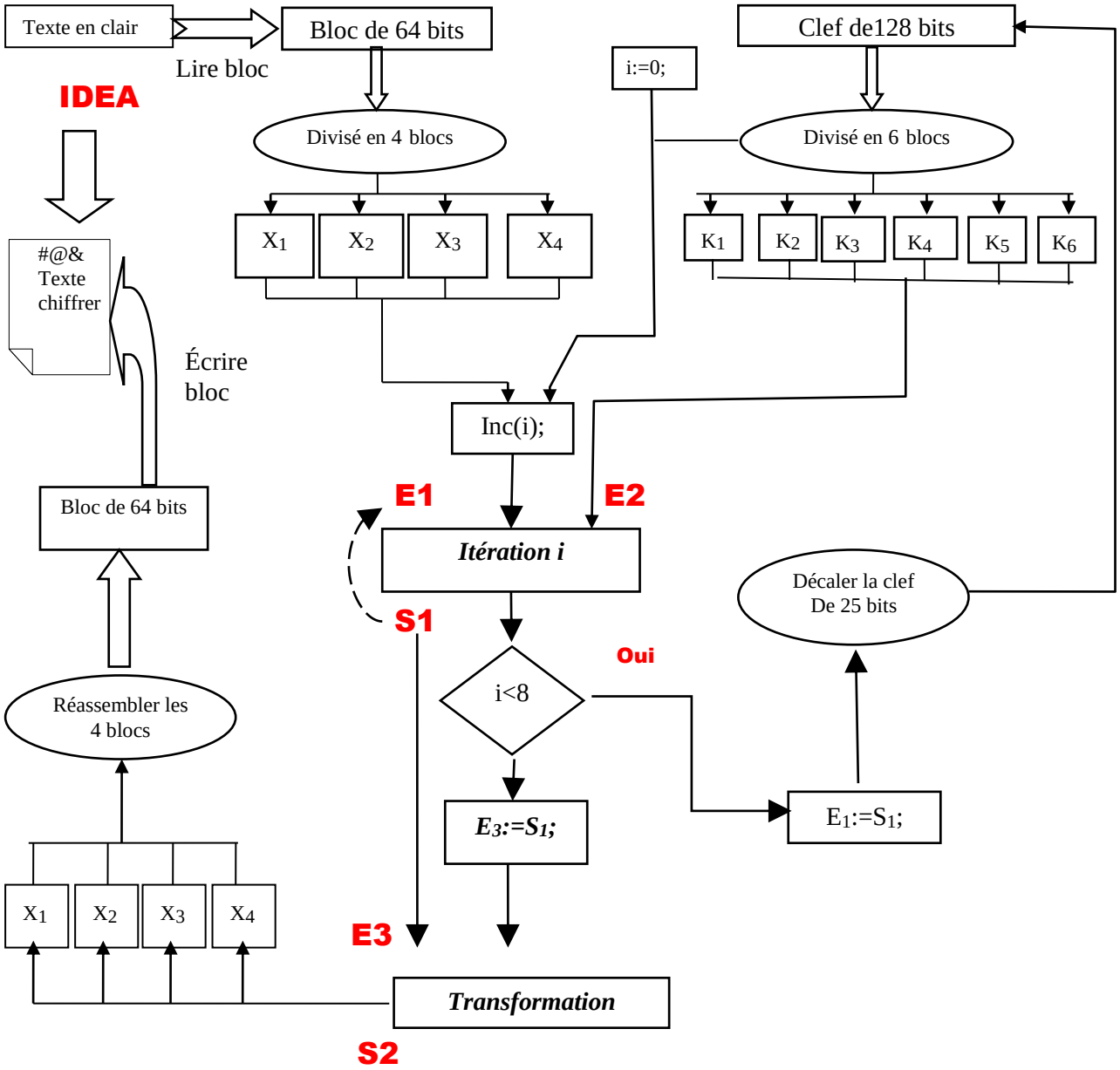
Parmi l'ensemble des algorithmes de chiffrement symétrique notre étude s'est portée sur l'algorithme d' **IDEA** (**International Data Encryption Algorithm**)

II.5) Algorithme symétrique IDEA(International Data Encryption Algorithm):

- **Principe :**

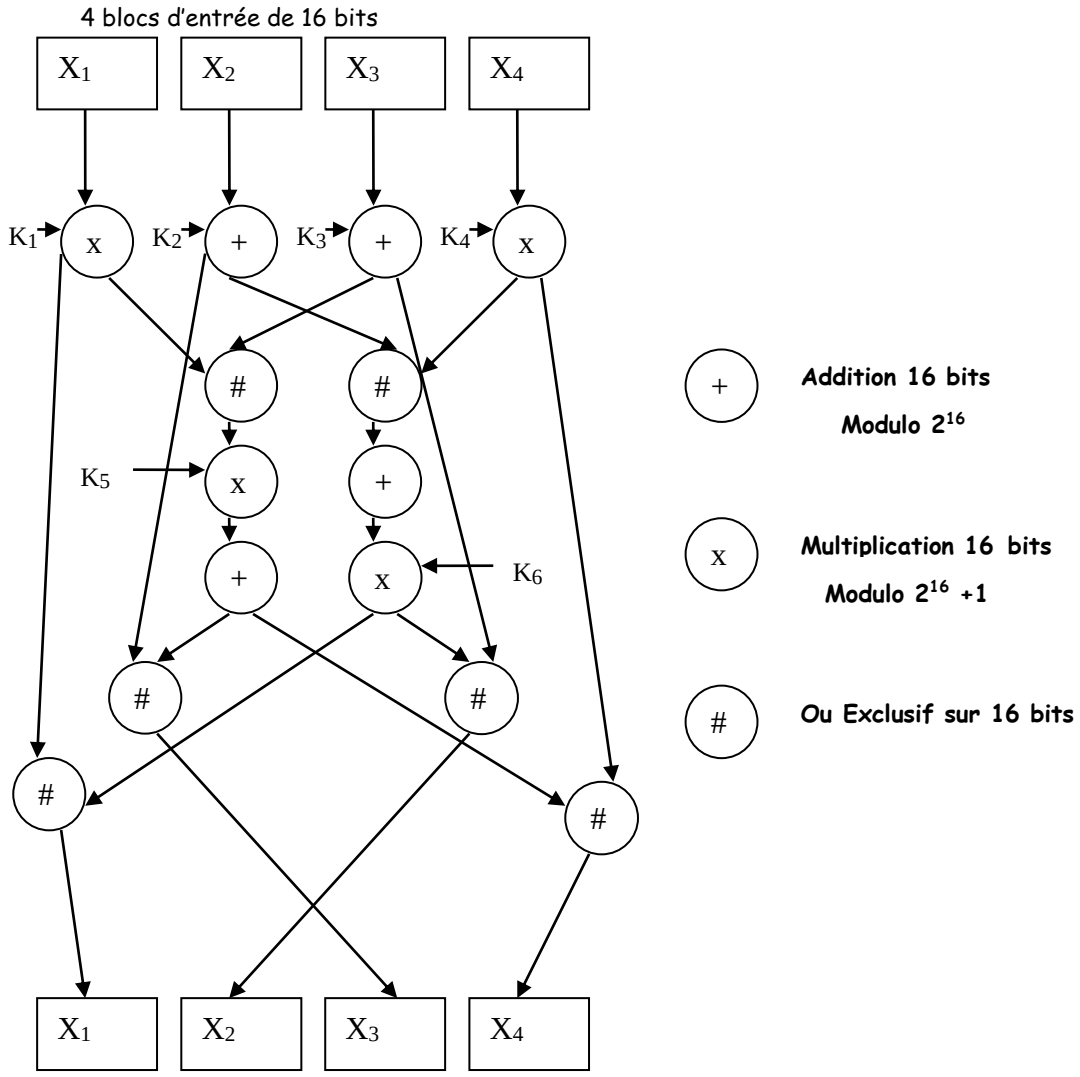
IDEA est un algorithme de chiffrement qui utilise une clé privée de 128 bits, et qui s'exécute sur 8 itérations. Cet algorithme n'utilise que trois opérations simples : le XOR, l'addition modulo 2^{16} et la multiplication modulo 2^{16+1}.

Pour l'**IDEA** Le texte est découpé en blocs de 64 bits, qui sont à leur tours redivisés en quatre blocs de 16 bits : X_1, X_2, X_3, X_4. La clé K de 128 bits est divisée en 8 blocs de 16 bits, puis ces fragments sont décalés circulairement sur la gauche de 25 bits, et redivisés, et ainsi de suite jusqu'à l'obtention de 52 clés. Ces clés formeront 8 groupes de 6 clés (un groupe par ronde) : K_1, K_2, K_3, K_4, K_5, K_6, et un groupe de 4 clés supplémentaires pour la ronde finale : K_1, K_2, K_3, K_4 (**figure II.4)** [4].

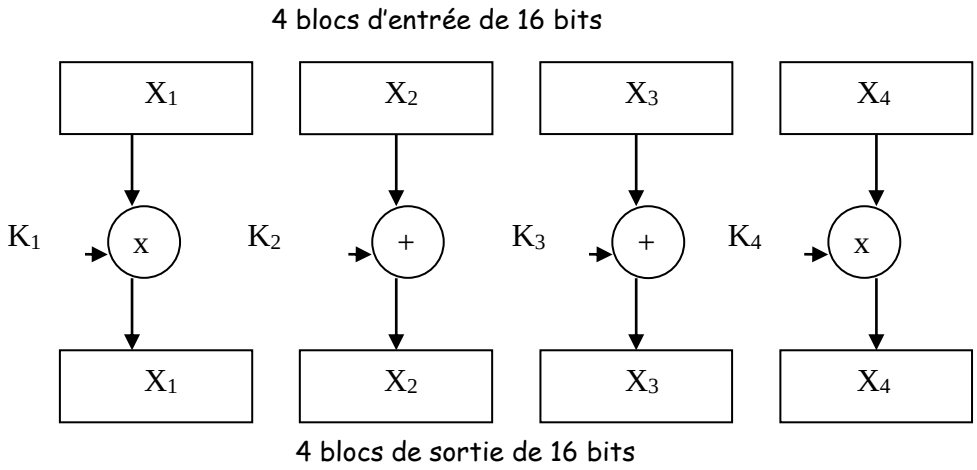


FigureII.4 : Fonctionnement de l'algorithme d'IDEA

La **figure II.5** et **figure II.6** montrent les différentes opérations combinant à l'aide de nos trois opérateurs (le XOR, l'addition modulo 2^{16} et la multiplication modulo $2^{16} + 1$) les 4 blocs de données aux blocs de la clé.



FigureII.5 : *L'organigramme d'une itération*



FigureII.6: détail de transformation

Chiffrement : Etapes des 8 rondes :

Code d'une itération :

- $Etape_1 = X_1 * K_1$
- $Etape_2 = X_2 * K_2$
- $Etape_3 = X_3 * K_3$
- $Etape_4 = X_4 * K_4$
- $Etape_5 = Etape_1$ XOR $Etape_3$
- $Etape_6 = Etape_2$ XOR $Etape_4$
- $Etape_7 = Etape_5 * K_5$
- $Etape_8 = Etape_6 + Etape_7$
- $Etape_9 = Etape_8 * K_6$
- $Etape_{10} = Etape_7 + Etape_9$
- $Etape_{11} = Etape_1$ XOR $Etape_9$ => X_1 de la ronde suivante
- $Etape_{12} = Etape_3$ XOR $Etape_9$ => X_3 de la ronde suivante
- $Etape_{13} = Etape_2$ XOR $Etape_{10}$ => X_2 de la ronde suivante
- $Etape_{14} = Etape_4$ XOR $Etape_{10}$ => X_4 de la ronde suivante

Pour finir, on applique une étape supplémentaire après la huitième ronde :

Code de transformation :

- $C_1 = X_1 * K_1$
- $C_2 = X_2 + K_2$
- $C_3 = X_3 + K_3$
- $C_4 = X_4 * K_4$

Les 4 blocs C_1, C_2, C_3, C_4, forment alors le message chiffré.

Déchiffrement :

Pour déchiffrer le texte, il faut d'abord inverser la dernière opération :

- $C_1 = C_1 * K_1^{-1}$
- $C_2 = C_2 - K_2$
- $C_3 = C_3 - K_3$
- $C_4 = C_4 * K_4^{-1}$

On applique alors les opérations suivantes selon 8 rondes, en utilisant les groupes de 6 clés en partant de la dernière à la première :

- $Etape_1 = C_1$ XOR C_3 ($Etape_5$ lors du cryptage)
- $Etape_2 = C_2$ XOR C_4 ($Etape_6$ lors du cryptage)
- $Etape_3 = Etape_1 * K_5$ ($Etape_7$ lors du cryptage)
- $Etape_4 = Etape_2 + Etape_3$ ($Etape_8$ lors du cryptage)
- $Etape_5 = Etape_4 * K_6$ ($Etape_9$ lors du cryptage)
- $Etape_6 = Etape_3 + Etape_5$ ($Etape_{10}$ lors du cryptage)
- $Etape_7 = C_1$ XOR $Etape_5$ ($Etape_1$ lors du cryptage)
- $Etape_8 = C_3$ XOR $Etape_5$ ($Etape_3$ lors du cryptage)
- $Etape_9 = C_2$ XOR $Etape_6$ ($Etape_2$ lors du cryptage)

- $Etape_{10} = C_4$ XOR $Etape_6$ ($Etape_4$ lors du cryptage)
- $Etape_{11} = Etape_7 * K_1^{-1} \Rightarrow C_1$ de la ronde suivante
- $Etape_{12} = Etape_8 - K_3 \Rightarrow C_3$ de la ronde suivante
- $Etape_{13} = Etape_9 - K_2 \Rightarrow C_2$ de la ronde suivante
- $Etape_{14} = Etape_{10} * K_4^{-1} \Rightarrow C_4$ de la ronde suivante

Les 4 blocs C_1, C_2, C_3, C_4 obtenus après la dernière ronde forment alors le message en clair.

L'IDEA est l'algorithme le plus sûr parmi les algorithmes de chiffrement symétrique.

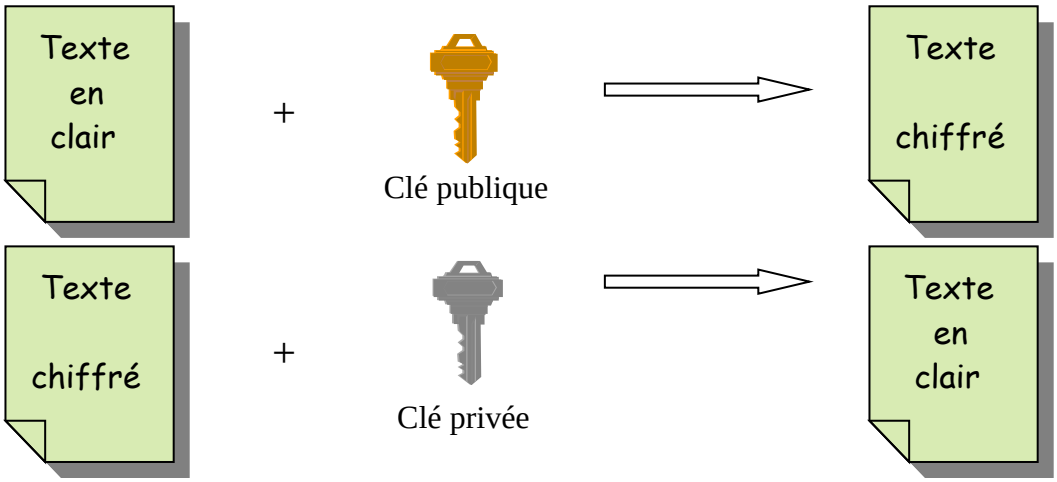
II.6) Chiffrement asymétrique ou à clé publique :

Dans ce type de cryptographie, chaque utilisateur utilise deux clés liées mathématiquement.

- La première est privée et différente pour chaque utilisateur qui doit être gardée secrète.
- La deuxième est publique et qui est connue par tous les utilisateurs.

Dans la pratique, la clé publique sert à crypter les messages, et la clé privée sert à les décrypter. Une fois le message crypté, seul le destinataire est en mesure de le décrypter.

Parmi l'ensemble des algorithmes de chiffrement asymétrique notre étude s'est portée sur l'algorithme du **RSA** (figureII.7) .



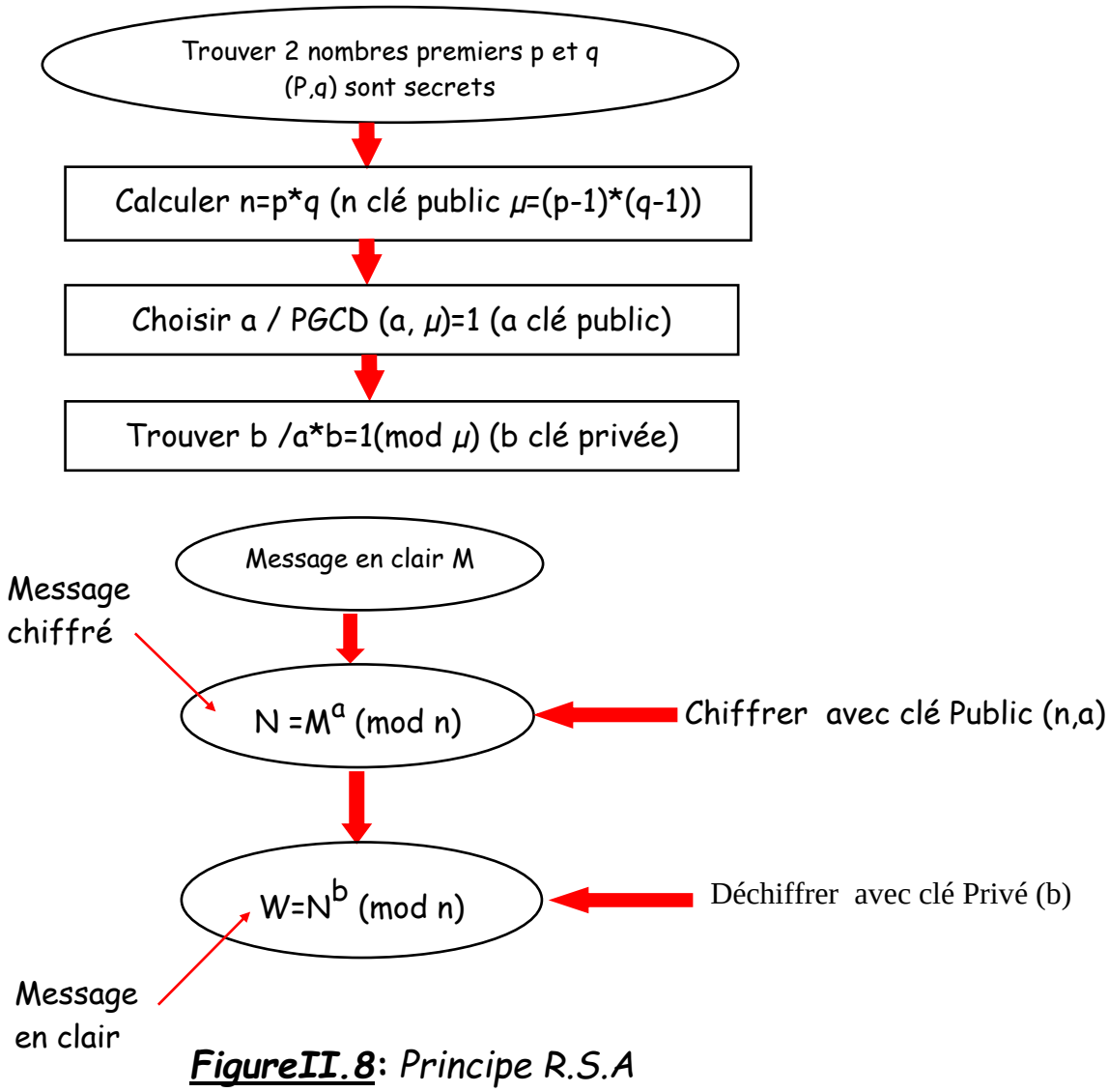
FigureII.7 : *Principe du chiffrement*

II.7) Algorithme asymétrique RSA (Ronald **R**ivest, Adi **S**hamir, et Leonard **A**dleman) :

- **Principe :**

Cette algorithme de **chiffrement** est une méthode à **clé publique** qui repose sur certains concepts de la théorie des nombres. Il a été nommé à partir des noms de famille de ses inventeurs (Ronald Rivest, Adi Shamir, et Leonard Adleman).

Les grandes lignes de l'algorithme RSA sont montrées dans la (**FigureII.8)**. Le principe est très simple et en voici les principales étapes [5]:



***FigureII.8*: Principe R.S.A**

Pour mieux comprendre voici un exemple simple qui décrit les étapes de cet algorithme décrit ci-dessus :

Exemple :

Soient les deux nombres premiers suivants (On commence par choisir deux nombres premiers quelconques) : p=79 , q=127

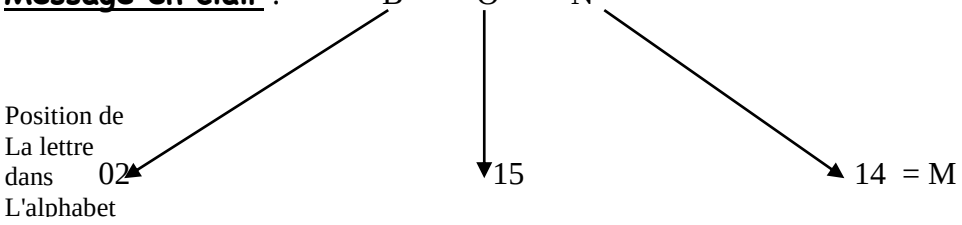
La Clé publique : est calculée de la façon suivante :

- n = p x q = 79 x 127 = 10033
- a = 97 tel que le PGCD (a,(p-1)*(q-1)) = 1

La clé privée : est calculée de la façon suivante :

- b = 25 tel que a*b = 1 (mod (p-1)*(q-1))
- (p-1)*(q-1) =78 * 126 = 9828.

Message en clair : B O N



Bloc1 Bloc2

0002 1514 devisé M en blocs de 4 caractères de droite vers gauche (On ajoute des 0 si M est non multiple de 4 caractères)

$1514^{97} \bmod (10033) = 2500$

$0002^{97} \bmod (10033) = 9208$

Chiffrer chaque bloc

Message codé est : 9208 2500

- **Les inconvénients de l'algorithme RSA** ***:***

La principale faiblesse de l'algorithme RSA provient des temps de calculs nécessaires au cryptage.

Donc , on ne les emploiera que pour transmettre des données courtes (de quelques octets) tels que les clés privées .

Les crypto systèmes à clés publiques tel RSA sont également moins performant au niveau du cryptage proprement dit. Une clé publique de 512 bits (utilisé pour le chiffrage RSA) offre approximativement le même niveau de sécurité qu'une clé de 40 bits utilisée pour un algorithme à clé privé (type IDEA).

II.8) Algorithme du PGP :

- **Le principe:**

PGP est un crypto système (système de chiffrement) inventé par Philip Zimmermann, un analyste informaticien.

PGP est une combinaison des fonctionnalités de la cryptographie de clé publique et de la cryptographie conventionnelle. PGP est donc un système de cryptographie hybride.

Lorsqu'un utilisateur chiffre un texte avec PGP, les données sont d'abord compressées. Cette compression des données permet de réduire le temps de transmission par modem, d'économiser l'espace disque et, surtout, de renforcer la sécurité cryptographique.

Ensuite, l'opération de chiffrement se fait principalement en deux étapes :

- PGP crée une clé secrète IDEA de manière aléatoire, et chiffre les données avec cette clé.
- PGP crypte la clé secrète IDEA et la transmet au moyen de la clé RSA publique du destinataire.

De même, l'opération de décryptage se fait aussi en deux étapes :

- PGP déchiffre la clé secrète IDEA au moyen de la clé RSA privée.
- PGP déchiffre les données avec la clé secrète IDEA précédemment obtenue.

Cette méthode de chiffrement associe la facilité d'utilisation du cryptage de clé publique à la vitesse du cryptage conventionnel. Le chiffrement conventionnel est environ 1 000 fois plus rapide que le cryptage de clé publique. Le cryptage de clé publique résout le problème de la distribution des clés. Utilisées conjointement, ces deux méthodes améliorent la performance et la gestion des clés, sans pour autant compromettre la sécurité. Il est très rapide et sûr ce qui le rend quasiment impossible à cryptanalyser [6].

II.9) Le hachage :

Une bonne cryptographie doit pouvoir offrir une garantie de l'intégrité des informations. En effet, il ne doit pas être possible de pouvoir modifier des informations cryptées de façon totalement transparente. Un processus de vérification de l'intégrité du message (crypté et en clair) doit être mis en place. Ce processus est réalisé par une fonction de hachage.

Une fonction de hachage (parfois appelée fonction de condensation) est une fonction permettant d'obtenir un condensé (appelé aussi haché) d'un texte, c'est-à-dire une suite de caractères assez courte représentant le texte qu'il condense.

Les algorithmes de hachage les plus utilisés actuellement sont :

- **MD5** (MD signifiant Message Digest),
- **SHA-1** (Secure Hash Algorithm 1),
- **SHA-2** (Secure Hash Algorithm 2).
- **RIPEMD-160** (Ripe Message Digest)

En expédiant un message accompagné de son haché, il est possible de garantir l'intégrité d'un message, c'est-à-dire que le destinataire peut vérifier que le message n'a pas été altéré (intentionnellement ou de manière fortuite) durant la communication.

Lors de la réception du message, il suffit au destinataire de calculer le haché du message reçu et de le comparer avec le haché accompagnant le document. Si le message (ou le haché) a été falsifié durant la communication, les deux empreintes ne correspondront pas [3].

II.10) Le certificat :

C'est un document électronique qui fait correspondre une clé à une entité (personne, entreprise, ordinateur...). Cette correspondance est validée par une autorité de certification (Certificate Authority : CA). Ces certificats sont utilisés pour identifier une entité. Ce certificat est normalisé (norme X.509). Concrètement, les données utilisateur (identité du propriétaire de la clé, la clé publique et l'usage de la clé) sont elles mêmes signées par l'autorité de certification, en y incluant certaines données propres (période de validité du certificat, l'algorithme de cryptage utilisé, numéro de série, etc...).

Les certificats sont des petits fichiers divisés en deux parties :

- La partie contenant les informations
- La partie contenant la signature de l'autorité de certification

La structure des certificats est normalisée par le standard X.509 qui définit les informations contenues dans le certificat :

- Le nom de l'autorité de certification
- Le nom du propriétaire du certificat
- La date de validité du certificat
- L'algorithme de chiffrement utilisé
- La clé publique du propriétaire

L'ensemble de ces informations est signé par l'autorité de certification, cela signifie qu'une fonction de hachage crée une empreinte de ces informations, puis ce condensé est chiffré à l'aide de la clé publique de l'autorité de certification, clé publique ayant été préalablement largement diffusée ou elle-même signée par une autorité de niveau supérieur.

Lorsqu'un utilisateur désire communiquer avec une autre personne, il lui suffit de se procurer le certificat du destinataire. Ce certificat contient le nom du destinataire, ainsi que sa clé publique et est signé par l'autorité de certification. Il est donc possible de vérifier la validité du message en appliquant d'une part la fonction de hachage aux informations contenues dans le certificat, en déchiffrant d'autre part la signature de l'autorité de certification avec la clé publique de cette dernière et en comparant ces deux résultats.

II.11) La signature digitale :

C'est un code électronique unique qui permet de signer un message codé. Cette signature permet d'identifier l'origine du message : elle a la même fonction qu'une signature "à la main". C'est la clé privée qui permet de signer, et la clé publique qui permet de vérifier cette signature.

Conclusion :

La cryptographie a donc connu une rapide évolution en s'enrichissant de nouvelles techniques de chiffrement et c'est en partie du à deux facteurs essentiels [doc1] :

1. L'irruption des mathématiques et de l'informatique .
2. le développement des moyens de télécommunication .

Qui a eu pour effet de multiplier ses activités .Avant elle n'avait pour rôle que de protéger un texte écrit ; maintenant elle s'étend dans différents domaines , pour cela Il faut que la cryptographie avance , c'est à dire qu'il faut toujours chercher de nouveaux algorithmes plus puissants même si ceux utilisés aujourd'hui sont fiables. Et dans ce cadre de recherche des méthodes cryptographique plus puissant , on va exploiter le mécanisme de la génétique pour créer un algorithme cryptographique performant .

Chapitre III :

Les algorithmes génétiques

III.1) Introduction aux algorithmes génétiques(AG):

Un algorithme génétique est, au premier lieu, un algorithme ; ce qui veut dire une méthode générale qui permet de résoudre un problème spécifique, en suivant un procédé bien précis ou un ensemble d'instructions, que l'on peut appliquer de la même façon, quelles que soient les données du problème.

Un nombre de biologistes ont utilisé les ordinateurs pour réaliser des simulations de systèmes génétiques dans le but de comprendre les systèmes naturels. Ces simulations ont mené à des résultats très importants.

Plusieurs expériences de ce genre et dans plusieurs domaines en vue de comprendre les phénomènes naturels ont vu le jour, mais sans soupçonner que l'algorithme d'exploration employé par la nature pourrait être utile pour les systèmes artificiels.

Alors, le vocabulaire employé dans les algorithmes génétiques est directement calqué sur celui de la théorie de l'évolution et de la génétique. C'est au début des années 1960 que john holland de l'université du Michigan, avec ses collègues et ses étudiants, a commencé à s'intéresser à ce qui allait devenir les algorithmes génétiques. Ses travaux ont trouvé un premier aboutissement en 1975 avec la publication de ***Adaptation in Natural an Artificial System*** .

John holland a développé des idées et des théories qui ont conduit à la création des nouveaux principes d'un algorithme, fondés sur les mécanismes de la sélection naturelle et de la génétique.

Simultanément, Holland a réalisé le rôle fondamental la sélection naturelle – la survie naturelle du plus adapté - .

Donc : **Qu'est ce qu'un algorithme génétique ?** Un algorithme génétique est une méthode de recherche itératif, dont le but est d'optimiser une fonction de coût (ou fitness) [doc2] . Il utilise à la fois les principes de la survie des structures les mieux adaptées et les échanges d'informations

pseudo aléatoires, pour former un algorithme d'exploration qui possède certaines caractéristiques de l'exploration humaine [8].

III.2) Les concepts importants des algorithmes génétiques :

Par analogie à la science les algorithmes génétiques ont pris les concepts importants de cette dernière. Ces algorithmes ont pris :

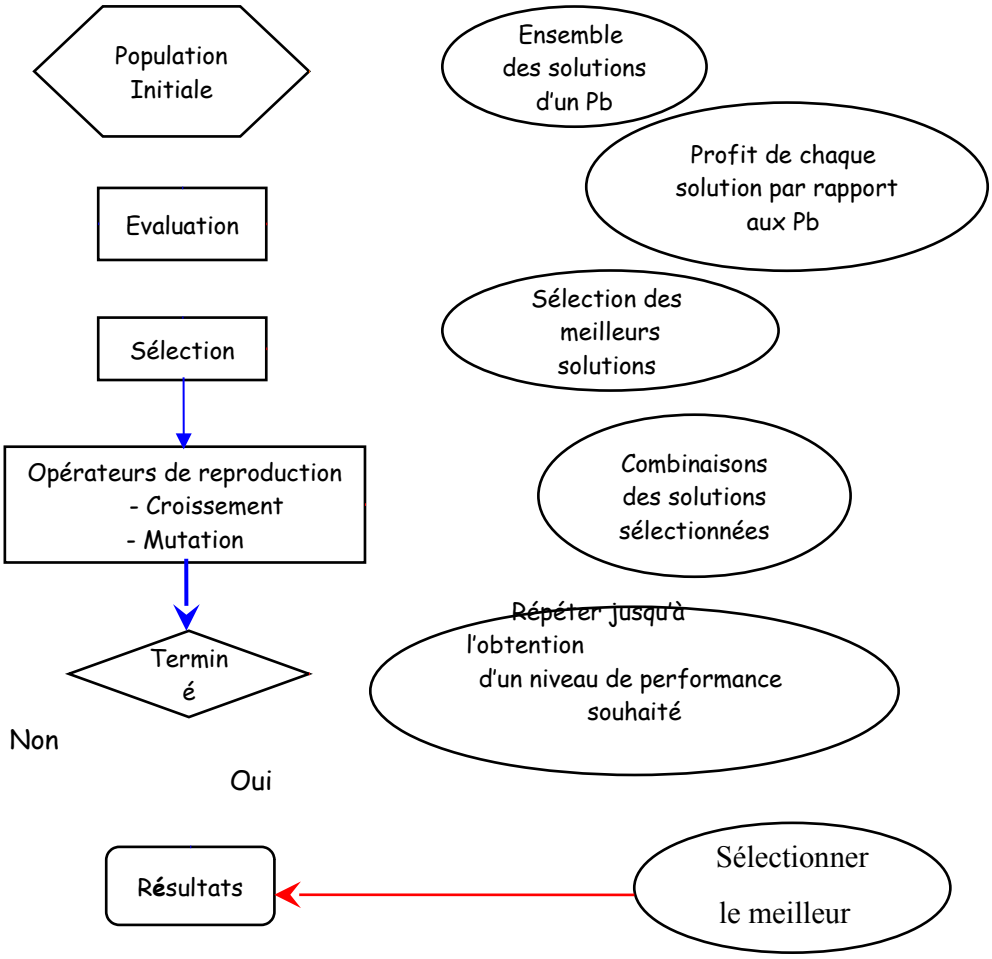
- **Individu** : entité représentée par son code génétique .
- **Population** : groupe d'individus.
- **Evaluation** : mesure de l'aptitude d'un individu à son environnement.
- **Sélection** : choix des individus pour la reproduction.
- **Reproduction** : croisement (crossover) + mutation.
- **Croisement (crossover)** : création d'individu à partir de deux individus parents.
- **Mutation** : modification (aléatoire) du code génétique d'un individu.
- **Fitness (coût)** : fonction à optimiser (fonction d'adaptation) .
- **Les chromosomes** : sont les éléments à partir des quels sont élaborés les solutions (mutation et croisement génétiques).
- **La population** (génération) : est l'ensemble des chromosomes

III.3) Applications : Les domaines d'applications sont multiples :

- optimisation de fonctions numériques difficiles (discontinues…)
- traitement d'image (alignement de photos satellites, reconnaissance de suspects…)
- optimisation d'emplois du temps.
- optimisation de design.
- contrôle de systèmes industriels.
- apprentissage des réseaux de neurones.

III.4) Principe :

IL s'agit de simuler l'évolution d'une population d'individus divers (généralement tirer aléatoirement au départ) à laquelle on applique différents opérateurs (recombinaisons, mutation) que l'on soumet à une sélection, à chaque génération .Si la sélection s'opère à partir de la fonction d'adaptation, alors la population tend à s'améliorer (**figure III.1**) [9].



FigureIII1 : *Organigramme d'un algorithme génétique simple*

Un algorithme génétique suit les étapes suivantes :

1. Générer aléatoirement une population de n chromosomes x;
2. Évaluer l'adaptabilité f(x) de chaque chromosome;
3. Créer une nouvelle population en :

   3.1 Sélectionner 2 parents chromosomes

   3.2 Croiser les deux parents pour obtenir un enfant

   3.3 Muter l'enfant obtenu avec une certaine probabilité.

   3.4 Placer l'enfant dans la population
4. Composer la nouvelle population
5. Si la nouvelle population n'est pas satisfaisante refaire les étapes à partir de **2**.

III.5) Mécanisme de déroulement d'un algorithme génétique :

Les mécanismes d'un algorithme génétique de base sont très simple, et ne mettent en jeu rien de plus compliqué que des copies de chaînes et des échanges de morceaux de chaînes.

Ainsi, un algorithme génétique commence avec une population de chaînes (individus) et génère par la suite des populations de chaînes successives. Et pour effectuer une recherche performante de structures de plus en plus intéressantes, il n'a besoin que des valeurs de la fonction à optimiser associées à chaque chaîne.

Il est maintenant nécessaire de définir un ensemble d'opérations simples qui permettent de produire, à partir d'une population initiale de la génération successive de plus en plus performante.

La génération d'une nouvelle population à partir de la précédente s'effectue suivant les étapes suivantes :

1. **Génération de la population initiale** :

Le choix de la population initiale d'individus conditionne fortement la rapidité de l'algorithme. Alors, il est préférable de démarrer avec une population initiale répartie de façon équitable sur l'espace de recherche ou avec des solutions approximatives assez bonnes qui ont été expérimentées auparavant.

2. **Evaluation :**

A chaque itération ou génération , une nouvelle population est créée avec le même nombre d'individus , cette nouvelle génération contient généralement des individus plus adaptés à l'environnement tel qu'il est représenté par la fonction d'adaptation. Seuls les individus qui codent les meilleurs solutions seront retenus.

3. **La sélection :**

La sélection permet d'identifier statiquement les meilleurs individus d'une population et d'éliminer les mauvais .Il y' a plusieurs méthodes de sélection, les plus utilisées sont :

a. *Roulette de casino :* C'est la sélection naturelle la plus employée pour l'AG binaire. Chaque chromosome occupe un secteur de roulette dont l'angle est proportionnel à son indice de qualité. Un chromosome est considéré comme bon aura un indice de qualité élevé, un large secteur de roulette et alors il aura plus de chance d'être sélectionné.

b. *" N/2 –élitisme " :* Les individus sont triés selon leur fonction d'adaptation, seule la moitié supérieure de la population correspondant aux meilleurs composants est sélectionnée, il est constaté que la pression de sélection est trop forte, donc il est important de maintenir une diversité de gènes pour les utilises dans la population suivante et avoir des populations nouvelles quand on les combine.

c. *"par tournoi " :* Choisir aléatoirement deux individus et on compare leur fonction d'adaptation (combattre) et on accepte le plus adapté pour accéder à la génération intermédiaire, et on répète cette opération jusqu'à remplir la génération intermédiaire (N/2 composants). Les individus qui gagnent à chaque fois on peut les copier plusieurs fois ce qui favorisera la pérennité de leurs gènes.

4. **Opérateur de croisement :**

Le phénomène de croissement est une propriété naturelle de l 'ADN. C'est par analogie qu'ont été conçus les opérateurs de croissement dans les algorithmes génétiques.

Le croissement (crossover) est donc la transposition informatique du mécanisme qui permet, dans la nature, la production de chromosomes qui héritent partiellement des caractéristiques des parents.

Le croissement a pour but d'enrichir la diversité de la population en manipulant la structure des chromosomes (individus). Pour effectuer des croissements sur des chromosomes constitués de M gènes, on tire aléatoirement une position dans chacun des parents. On échange

ensuite les deux sous chaînes terminales de chacun des deux chromosomes (P1, P2) , ce qui produit deux enfants (F1,F2) (**figure III.2) .** .

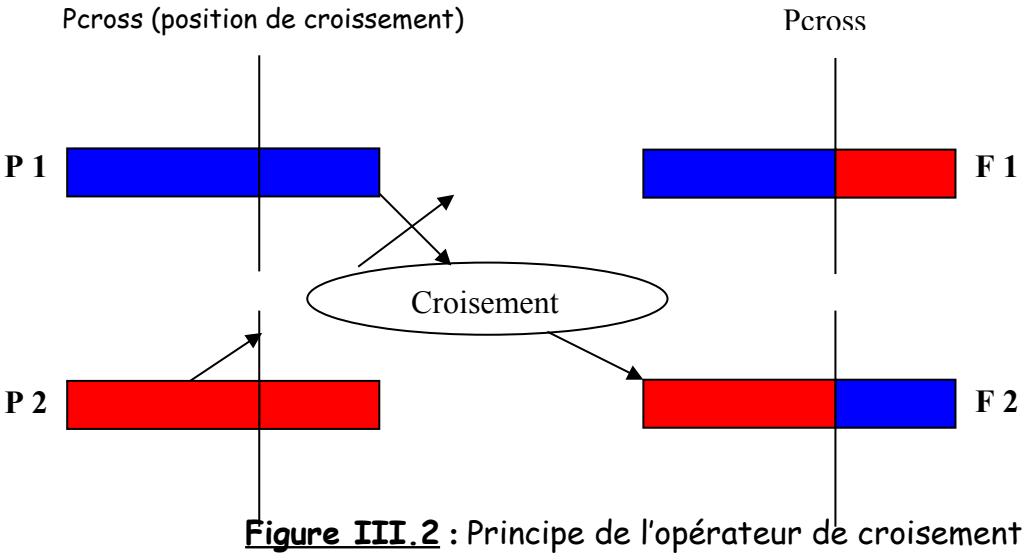


Figure III.2 : Principe de l'opérateur de croisement

5. **Opérateur de mutation** :

L'opérateur de mutation consiste, généralement, à tirer aléatoirement un gène dans le chromosome et le remplacer par une valeur aléatoire. Alors, la mutation est la modification aléatoire occasionnelle (de faible probabilité) de la valeur d'un caractère de la chaîne (**FigureIII.3**). Son rôle est de protéger le système de la perte de la matière génétique potentiellement utile provenant de la reproduction et le croisement.

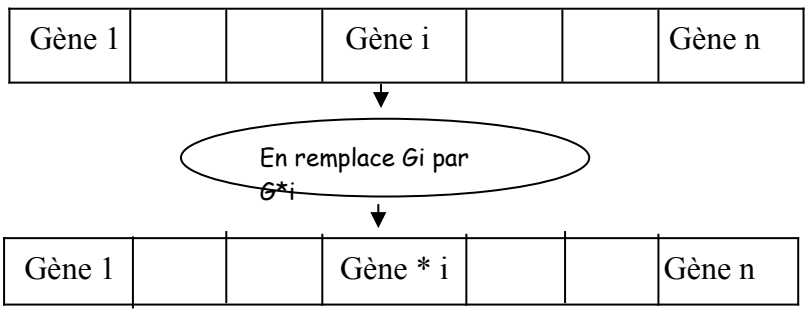


Figure III.3 : Principe de l'opérateur de

III.6) Conclusion :

Les domaines d'application des algorithmes génétiques s'étendent de plus en plus et l'avenir est à eux. Mais en dépit de leur simplicité, les algorithmes génétiques ne sont pas évidents dans leur application.

En fait, concevoir et réaliser un bon algorithme évolutionniste demande une bonne connaissance du problème, une bonne compréhension des mécanismes évolutionnistes et beaucoup de créativité.

Chapitre IV :

La cryptographie évolutionniste

IV.1) Introduction :

De nos jours, la cryptographie s'est imposée dans la vie courante notamment pour la protection (Ou Sécurité) des transactions. Malgré son antiquité (plus de 3000 ans) elle est toujours à l'état embryonnaire. En effet, les vrais algorithmes de chiffrement sont comptés sur les bouts des doigts (DES, IDEA (1977), RSA (1970) …). Et actuellement, le cryptosystéme dominant est le PGP (1992). Parallèlement, les domaines d'application des algorithmes évolutionnistes n'ont cessé de s'élargir grâce à leur grande efficacité (performance, rapidité,…).Leur succès a atteint son apogée dans le domaine de L'intelligence artificielle et recherche opérationnelle pour résoudre des problèmes d'optimisation. Certes, concevoir et réaliser un bon algorithme évolutionniste pour chiffrement peut avoir un grand Succès.

Vu le grand succès des algorithmes évolutionnistes dans les problèmes d'optimisations, nous les exploitons dans la phase principale de la cryptographie soit: le chiffrement. Dans ce chapitre nous avons étudié un algorithme de chiffrement évolutionniste OTL développé par (Fouzia **O**mary –Abderrahim **T**ragha –Aboubakr **L**bekkouri) [10]. Pour commencer, il faut coder le problème de chiffrement en le simulant (ou rapprochant) à un problème d'ordonnancement (notamment permutations). Après , on crée un codage adapté pour les chromosomes. Ensuite en définit la fonction d'évaluation adéquate, pour appliquer les opérateurs génétiques.

IV.2) Description d'algorithme de chiffrement OTL :

Soit M le message à chiffrer. M est une suite de N caractères. Soit C une clé secrète formée de p caractères. Après un brouillage initial (ce brouillage peut être effectué par combinaison de plusieurs méthodes simples comme les substitutions ou les transpositions matricielles .... à ce stade, nous devons disposer de clés secrètes symétriques.) permettant de masquer les caractères de M, nous décomposons ce dernier en m blocs B_1, B_2,.., B_m de même taille n.

Note : cette décomposition n'aura lieu que pour les messages de grandes tailles. Nous distribuons les m blocs sur m processeurs fonctionnant en parallèle .Chacun des processeurs applique l'algorithme **OTL**. Dorénavant B désigne le bloc courant affecté à l'un des processeurs et O.T.L algorithme de chiffrement.

- **Schématisation du problème**

Soit B le message à chiffrer (de taille n). a_1, a_2, ..., a_l les différents caractères de B (l<=255). Désignons par L_i ($1 \leq i \leq l$) la liste des différentes positions du caractère (a_i) dans B avant le chiffrement et par (occu) le nombre des occurrences de a_i dans B. le bloc B peut être représenté par le tableau ci-dessous (**figureVI.1**) :

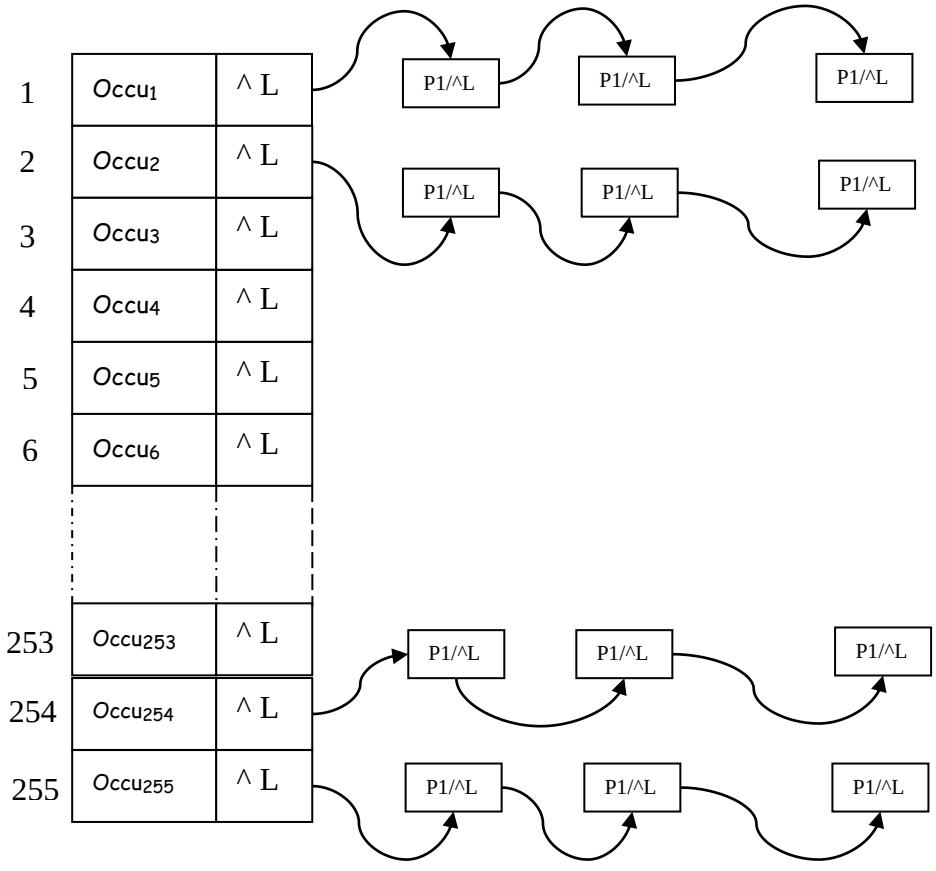


Figure IV.1 : La représentation du message à chiffrer (structures de données utilisées)

Désignons par :

- l'indic i de M-tab *:* le code ASCII de caractères ai de M .

- Li : la liste des positions du caractère a^i dans M.

- occu i : le nombre des occurrences de a^i dans M.

Le but de L'OTL est de créer le maximum de désordre dans les positions des caractères de B. de telle manière que la différence entre le cardinal de la nouvelle liste affectée à chaque caractère ai et le cardinal de la liste Li d'origine soit maximale. Nous sommes donc devant un problème d'optimisation.

IV.3) <u>Squelette de l'algorithme</u> :

- **<u>Etape 0</u>** : Définir un codage du problème.

- **<u>Etape 1</u>** : Créer une population initiale P0 de q individus{X1, X2, …, Xq}

- **<u>Etape 2</u>** : Evaluation des individus.

  Soit F la fonction d'évaluation. Calculer F(Xi) pour chaque individu Xi de la population Pi .

- **<u>Etape 3</u>** : Sélection

  Sélectionner les meilleurs individus (au sens de F(Xi)) et les grouper par paire.

- **<u>Etape 4</u>** : Application des opérateurs génétiques :

  1. <u>*Croisement*</u> : Appliquer l'opération de croisement aux paires sélectionnées.
  2. <u>*Mutation*</u> : Appliquer la mutation aux individus issus du croisement .

Ranger les nouveaux individus obtenus dans une nouvelle génération Pi+1.

Répéter les étapes 2,3 et 4 jusqu'à l'obtention du niveau de performance souhaité.

IV.4) l'algorithme OTL :

- **Le chiffrement :**

1. Le codage :

un individu (chromosome) : est une chaîne de caractère de taille l (l<=255) dont les gènes sont les caractères de M .

Ch- initial : chromosome initial dont les gènes sont (avec respect d'ordre) : a1, a2, ..., *al* Le i ème caractère d'un individu remplace Le i ème caractère de Ch-initial.

2. créer une population initial P0 de q individus {X1,X2,……Xq} :

Nous appliquons q bonnes permutation sur Ch-initial afin d'obtenir q individus distincts formant ainsi la population initiale constituée de q solutions potentielles du problème.

3. La fonction d'évaluation :

Soit X_k un individu de P_i (la taille maximale de la population est 30 individus) dont les gènes sont ak_1, ak_2 ,…,ak_l . Désignons par Card (ak_i) le nombre d'occurrence de caractère aki telle que :

Card (ak_i) = M_tab [pos (ak_i)].occu

Pos : fonction qui donne le code ASCII d'un caractère.

F la fonction d'évaluation sur l'ensemble X_k

$F(X_k) = \square \sum | \text{card}(ak_i) - \text{card}(a_i) |$

4. Le mécanisme de sélection :

Nous utilisons la méthode classique de la roulette, permettant de retenir les individus les plus forts.

5. <u>Les opérateurs génétiques</u> :

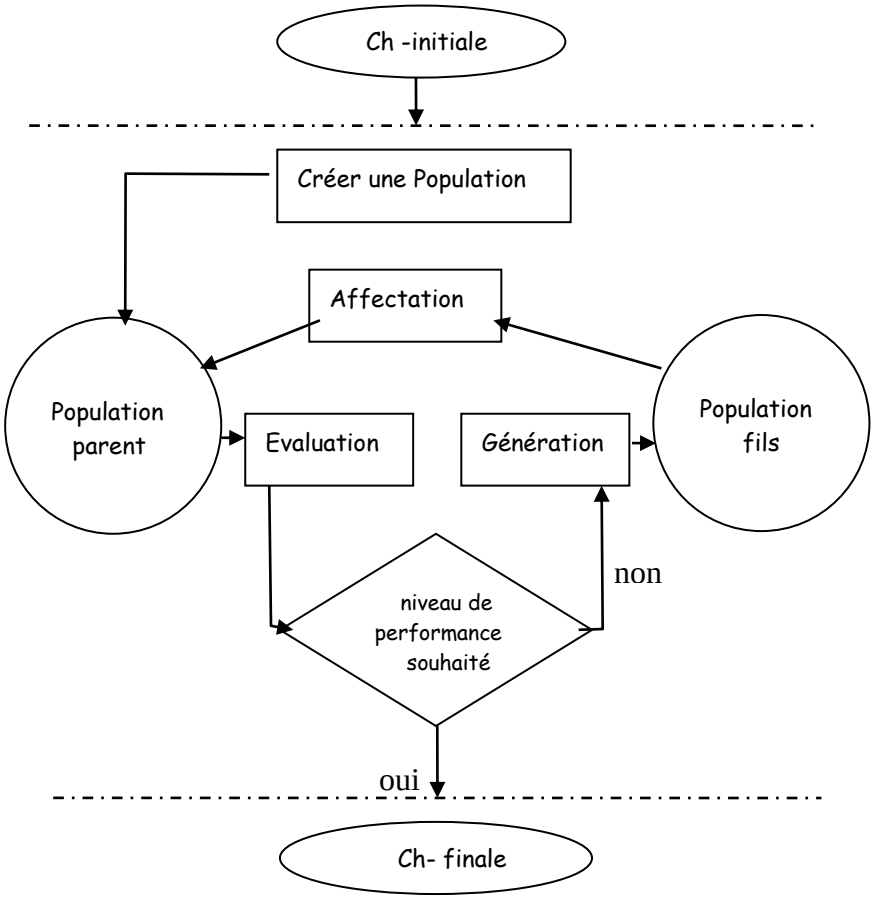
a- Croisement a un point. La probabilité du crossover est de 60% à 100%.

b- Mutation : permuter aléatoirement deux gènes d'un chromosome. La probabilité de la mutation est de 0.1% à 5%.

On applique le processus génétique jusqu'à l'obtention d'un niveau de performance souhaité.

6. <u>La phase finale de l'algorithme</u> :

A partir de la meilleure solution Ch-final obtenue par O.T.L nous constituons le bloc chiffré correspondant. Ensuite nous concaténons ces blocs chiffrés. Ceci nous donne le message chiffré M' du message initial M.Le schéma de la (**figureVI.2)** montre les différents étapes effectués par l'algorithme de chiffrement **OTL**.



<u>Figure IV.2</u> : Les étapes de chiffrement

- **Le déchiffrement :**

Le déchiffrement est effectué sans appliquer les AG, il suffit de déchiffrer le message avec la clef de session et déchiffrer le brouillage à l'aide de clés secrètes symétriques pour récupèrer le message en clair.

1. **Déchiffrement d'un bloc :**

Une fois la meilleure solution Ch- final est donnée par l'algorithme, nous identifions alors la permutation qui, partant de Ch- initial aboutit à Ch-final. Cette permutation servira de clé de session pour obtenir le chromosome initial Ch- initial. Ainsi à la fin de l'algorithme nous ajoutons une fonction dont le rôle est d'établir cette identification.

2. **Déchiffrement du brouillage :**

Toutes les fonctions ou opérations utilisées dans le brouillage admettent des réciproques permettant ainsi facilement le déchiffrement grâce aux clés symétriques.

IV.5) Conclusion :

Etant donné que la plupart des processus utilisés dans les algorithmes évolutionnistes sont aléatoires, ceci donne de la résistance à l' algorithme de chiffrement. En effet, l'aléatoire est un ennemi des cryptanalystes. D'autant plus, nous avons exploité ces processus pour déduire la clé de session en plus de clés secrètes symétriques utilisées initialement. Ces dernières renforcent la sécurité davantage. Cela dit, l'algorithme jouit de tous les avantages d'un algorithme évolutionniste, à savoir: simplicité, rapidité, coût faible des opérateurs génétiques et performance. Il suffit de choisir des paramètres (taux de population, taux des opérateurs,…) adaptés. Pour bien situer notre travail deux cas se présentent:

1-Si la clé de session peut être acheminée à travers un canal sûr (comme dans n'importe quel chiffrement symétrique à clé de session) alors nous avons établi un algorithme de chiffrement symétrique.

2-Si la clé de session est chiffrée par un algorithme asymétrique ensuite envoyée accompagnée du message chiffré nous pouvons dire que nous avons établi un système de chiffrement hybride (comme PGP).

Chapitre V :

Implémentation & Comparaison

I. Implémentation :

Dans ce chapitre nous allons présenter l'implémentation des principaux procédures utilisées lors du cryptage(**IDEA , RSA ,PGP , OTL, OTLmodifie**), et le fonctionnement du logiciel développé lors de ce travail. ainsi que les résultats obtenus après plusieurs essais .Ce logiciel est réalisé sous l'environnement Windows avec le langage de programmation Delphi.

1) Implémentation de l'IDEA :

Les procédures de l'IDEA sont les suivantes : shr25bit, generZ, transformation, iteration, crypte_bloc.

La procédure **shr25bit** réalise le décalage circulaire de 25 bits de la clef.

```
Procedure shr25bit (var cle :array of byte; var debutcle: byte);
var     test1,test2:byte;
        count2:integer;
begin
        debutcle:=(debutcle+3)mod 16 ; // le nombre de carctére de
                                       // la clé
        test1:=tableau1[15] and 1;
        count2:=0;
        while count2<=15  do
        begin
                test2:=tableau1[count2] and 1;
                tableau1[count2]:=tableau1[count2] shr 1;
                if (test1=1) then
                tableau1[count2]:=tableau1[count2]+ 128;
                test1:=test2;
                inc(count2);
        end;
```

Puis la clé est divisée en 8 sous-clés de 16 bits , décalées circulairement sur la gauche de 25 bits à l'aide de la procédure **generZ.** cette procédure est exécutée à chaque itération jusqu'à l'obtention 52 sous-clés c'est-à-dire 8 groupes de 6 sous-clés (un groupe par itération) : Z1, Z2, Z3, Z4, Z5, Z6, et un groupe de 4 sous-clés pour la transformation finale : Z1, Z2, Z3, Z4.

```
procedure generZ (var cle :array of byte; var debutcle; parcourcle:
                                                        byte);
var     round1:integer;
        bytevarlo, bytevarlo:byte;
        wordvar:word;
        procedure concat_2bytetoword;
        asm
        mov ah,bytevarlo;
        mov al,bytevarhi;
        mov wordvar,ax;
        end;
begin
                for round1:=0 to max then
        begin
        bytevarhi:=t2[parcourcle];varr1:=( parcourcle +1) mod 16;
        bytevarlo:=t2[parcourcle];varr1:=( parcourcle +1) mod 16;
        concat_2bytetoword;
        t1[round1]:=wordvar;
        if varr1=varr2 then
                begin
                shr25bit (cle, debutcle);
                parcourcle:= debutcle;
                end;
        end;
```

L'ensemble de ces sous clés seront combinées avec le bloc de données(X1,X2,X3,X4) à l'aide des deux procédures suivantes :

```
procedure transformation (var X,Z:array  Word );
var     round: integer;
begin
        X[1] := (X[1]+Z[1]) mod  n2E16;
        X[2] := (X[2]+Z[2]) mod  n2E16;
        X[3] := (X[3]+Z[3]) mod  n2E16;
        X[4] := (X[4]+Z[4]) mod  n2E16;
end;
```

```
procedure iteration (var X:array  Word ; var cle:array of byte);
var Etape :array[1..10] of  Word;
begin

        Etape[1] := (X[1]+Z[1])mod n2E16;
        Etape[2] := (X[2]+Z[2])mod n2E16;
        Etape[3] := (X[3]+Z[3])mod n2E16;
        Etape[4] := (X[4]+Z[4])mod n2E16;
        Etape[5] := Etape[1] XOR Etape[3];
        Etape[6] := Etape[2] XOR Etape[4];
        Etape[7] := (Etape[5] + Z[5])mod n2E16;
        Etape[8] := (Etape[6] + Etape[7])mod n2E16;
        Etape[9] := (Etape[8] + Z[6])mod n2E16;
        Etape[10]:= (Etape[7] + Etape[9])mod n2E16;
        X[1] := Etape[1] XOR Etape[9];
        X[3] := Etape[3] XOR Etape[9];
        X[2] := Etape[2] XOR Etape[10];
        X[4] := Etape[4] XOR Etape[10];
end;
```

La procédure **crypte_bloc** effectue le cryptage d'un bloc de données de 64 bits .

```
procedure crypte_bloc (var X:array  Word ; var cle:array of byte);
var round: integer;
begin
        generZ (Z,cle);
        for round:=1 to 8 do
        Begin
        Iteration (X, Z);
        generZ (Z,cle);
        end;
        transformation(X, Z);
end;
```

Présentation de l'interface d'IDEA :

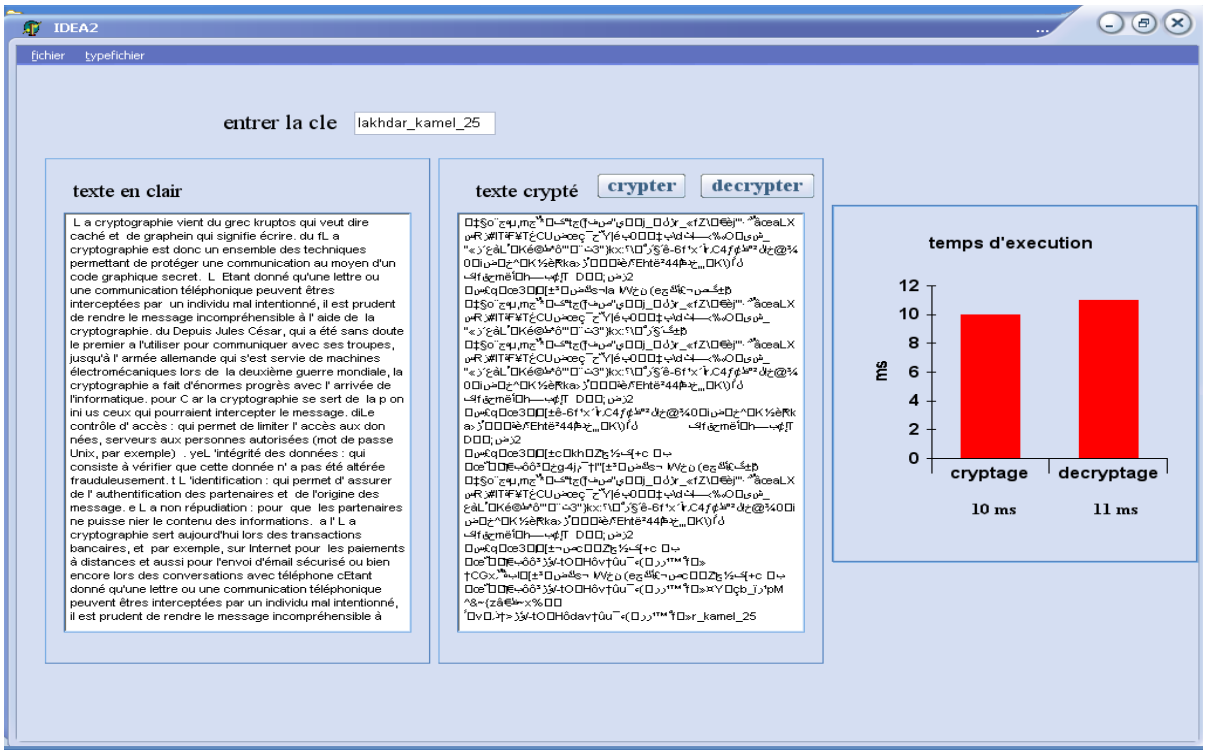


Figure V.1 : *Interface d'IDEA dans le cas du cryptage* d'un texte (11 ko)

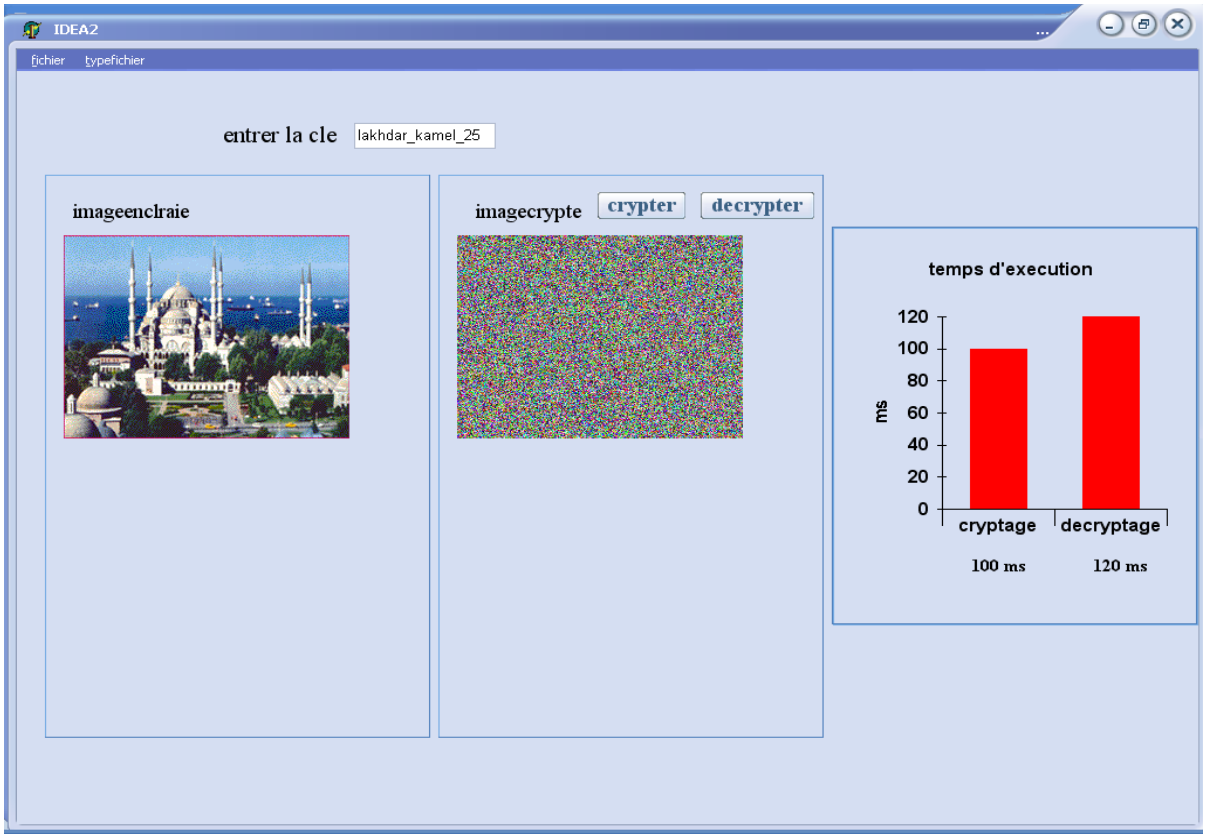


Figure V.2 : *Interface d'IEDA dans le cas de cryptage d'image* (45 ko)

Fonctionnement d'IDEA :

- **Pour coder** :

Premièrement il faut que l'utilisateur choisisse le mode de cryptage texte ou image (menu/typefichier) avant d'ouvrir le fichier adéquat et taper la clé de session (de 16 caractère de long) et pour lancer l'opération de cryptage il faut cliquer sur le bouton crypter.

- **Pour décoder :**

La personne voulant décrypter le fichier codé n'a pas besoin de taper la clé de session par ce qu'elle a été insérée dans ce fichier lors du cryptage (c'est notre idée afin d'acheminer la clé au destinataire d'une façon sécurisée) , donc il suffit de cliquer sur le bouton décrypter et le programme va premièrement récupérer la clef et en suite décrypter les données.

Résultats Expérimentaux :

➢ **Texte** :

Nous allons crypter le texte clair : **" there is only one God and Mohamed his prophet" .**

avec les clés : ' n_lagraa_b_kamel' , ' n_lagraa_lakhdar' **et** ' o_lakhda_b_kamel '.

Clé secrète	Texte crypté
n_lagraa_b_kamel	u {e Z æÆ¦,þº ÖÃC [¦—šêþ□Ðä[¦åM□U/□§Ñâ□"W×Õ¹ÊJ«ûI□«ôSf
n_lagraa_lakhdar	ðÊ÷ù‡"%õ¢œÀ¦ÄD]õ"sIî□-i ºlºóV9 M%Õ (^ ©žUÜ.H 8ïI É#w^B
o_lakhda_b_kamel	ÀÎgŬ4jã¯†I" †CGxöñúÉÈÇ□¤Y□çb_ïÑ¹pM ^&~{zâ€Ù~x%□ ðªv□öÐ†>

Tableau V.1 : Résultats obtenues avec l'IDEA

Nous remarquons que le changements des clés donne des résultats différents par contre le temps de cryptage reste constant .

- **Image** :

Nous allons crypter l'image BMP 244.188 avec la clé « cryptage_image_1» .

Image en claire

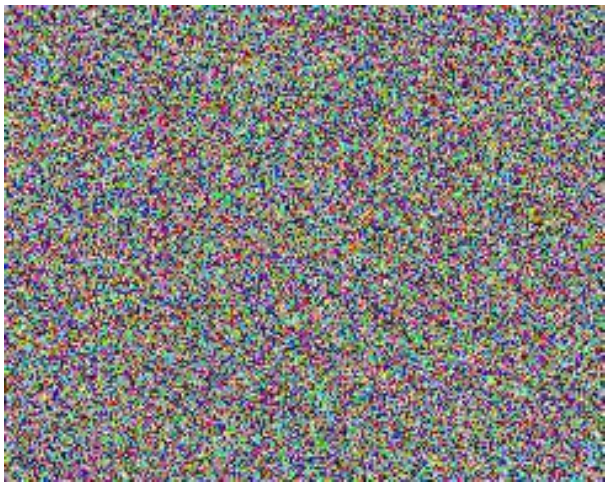


Image cryptée

2) Implémentation de l'RSA : Les procédures et fonctions constituant l'RSA sont :

premier ,premiersuiv, cleprive, clepublic, puissance, cryptemsg .

La fonction **premier :** tester si le nombre nbr est premier ou non .

```
function  premier(nbr:int64):boolean;
var       notexiste:boolean;
          nbrimper:integer;
begin
          notexiste:=true;
          nbrimper:=3;
          while (notexiste and (nbr div 3>= nbrimper)) do
          if (nbr mod nbrimper) =0      then notexiste:=false
                                        else nbrimper:= nbrimper+2;
          result:=notexiste;
end;
```

La fonction suivante donne le nombre premier qui suit le nombre nbr .

```
function  premier(nbr:int64):boolean;
var       notexiste:boolean;
          nbrimper:integer;
begin
notexiste:=true;
nbrimper:=3;
while (notexiste and (nbr div 3>= nbrimper)) do
if (nbr mod nbrimper) =0      then notexiste:=false
                              else nbrimper:= nbrimper+2;
result:=notexiste;
end;
```

La fonction **cleprive** est basée sur le théorème de bezout [doc3] et sert à calculer la clé privée dans le but de l'utiliser dans l'algorithme de chiffrement.

```
fonction  cleprive(v1,v2:int64):int64;
var       r,x,x1,x2,a,b:int64;
begin
          x:=1;     x1:=0;
          a:=v1;    b:=v2;
          while b>0 do
          begin
          r:=a mod b;
          x2:= x -(a div b)*x1;
          x:=x1;
          x1:=x2;
          a:=b;
          b:=r;
          end;
          cleprive:=x;
end;
```

La fonction **clepublic** comme son nom l'indique sert à calculer la clé publique [doc3].

```
fonction  clepublic(nbr:int64):int64;
var keyE:int64;
begin
 keyE:=nbr div 4;
 repeat
 premiersuiv(keyE);
 until  (nbr mod keyE<>0);

 clepublic:=keyE;
end;
```

La fonction suivante sert à calculer la puissance d'un nombre ,à l'aide de l'exponentiation modulo Nvar par algorithme dichotomique, cette fonction est utilisée pour le cryptage et le décryptage [doc3].

```
fonction puissance( msg2,a1, Nvar:int64):int64;
 var     boucle,msgvar,mul:int64;
begin
msgvar:=1;
mul:=msg2;
boucle:=a1;
while boucle<>0 do
if (boucle mod 2)<>0 then
begin
boucle:=boucle-1;
msgvar:=msgvar*mul mod Nvar;
end
else
begin
boucle:=boucle div 2;
mul:=mul*mul mod Nvar;
end;
puissance:=msgvar;
 end;
```

On crypte les messages **à l'aide de la procedure suivante :**

```
procedure cryptemsg(var msg:int64);
    begin
     msg:=puissance(msg,e);
    end;
```

Présentation de l'interface d'RSA :

Figure V.3 : *Interface d'RSA*

Fonctionnement d'RSA :

Tout d'abord il faut choisir deux nombres premiers différents p et q à l'aide du bouton suivant qui nous permet de choisir un nombre premier puis appuyer sur le bouton 'calculer' pour générer les clés de (dé)codage (privée/ publique) et le groupage c à d le nombre de lettres à coder à chaque passage (le groupage est inférieur au nombre de chiffre de p*q) .Dans ce cas , la personne voulant décrypter le message doit posséder les deux clés alors que les autres ne connaissent que la clé publique.

- **Pour coder :**

  Il suffit de taper la clé publique (fixée par la personne qui veut décoder) cliquer ensuite sur le bouton ' crypter '.

- **Pour décoder :**

  Taper la clé publique et la clé privée , placer le texte crypté dans le champ ' textenclaire ' , et appuyer ensuite sur le bouton décrypter.

Résultats Expérimentaux :

Nous allons crypter le même texte en claire employé avec l'IDEA , à chaque essai on utilise un couple (privée/ publique) différent :

Les Nbrs premier	Clé privée	Clé publique	groupage	Tps d'exe	Texte crypté
P=103 q=13	N=1339 E=311	D=551	1	5 ms	049301040277068002770843054703960843125007391271106308431250073902770843090812501199084302580739119908431013125001040258046302771190843010405470396084312660680125012660104027704930843084308430843084308430000000000000000000000
P= 1087 q=1361	N=1479407 E=369247	D= 263743	2	7 ms	125297103175330655465075295003439671224695007759203747940444367114810910723751219521051763613604131475350070889306952390550315116284700892051117956024976 5
P=104327 q=13591	N=1417908257 E=354447589	D=265835689	3	10 ms	104391230013963270620604073620087056168511456340440849465526085024887904183708470460259670018254469204364683551178963961023791672901206759770893980837

Tableau V.2 : *Résultats obtenues avec l'RSA*

3) Implémentation du PGP :

PGP est un système de chiffrement à clé publique qui combine les avantages de l'IDEA de RSA : IDEA est 1000 fois plus rapide que RSA et il est pratiquement impossible de percer la clé secrète RSA. Touts les procédures utilisées par l'IDEA et l'RSA sont exploitées par le PGP.

Présentation de l'interface de PGP :

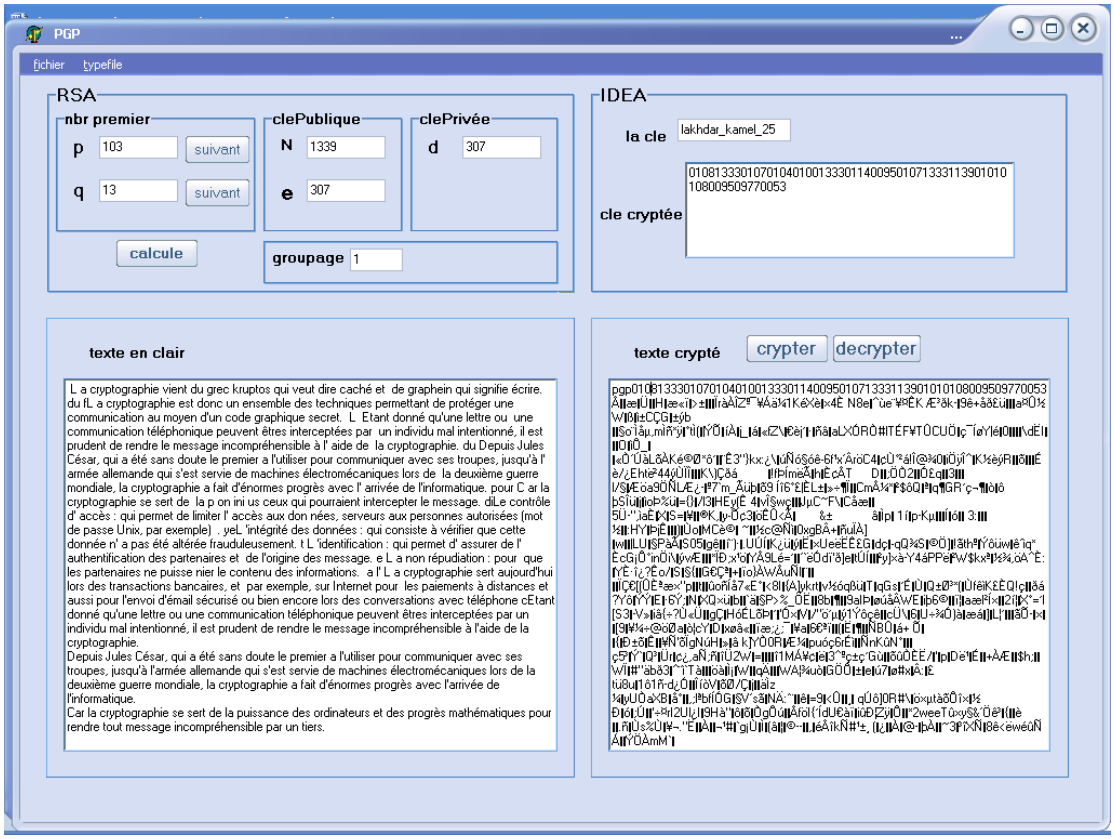


Figure V.4 : *Interface de PGP*

Fonctionnement du PGP:

Pour les clés (privée/publique) utilisées par le PGP ,on les génére de la même façon que l'RSA , tel que la clé de session d'IDEA est donnée par l'utilisateur .

- **Pour coder** :

Lorsque vous utilisez PGP pour coder des données (texte ou image) il faut tout d'abord entrer la clé publique (dans le champs RSA) donnée par le destinataire et taper la clé de session (dans le champs IDEA). une fois vous cliquez sur le bouton crypter les étapes suivantes sont automatiquement déclenchées :

a. PGP utilise l'algorithme IDEA pour chiffrer les données avec la clé de session.
b. Il emploie l'algorithme RSA pour chiffrer la clé de session avec la clé publique du destinataire.
c. Il sauvegarde, la clé de session chiffrée et les données chiffrées dans le même fichier.

Pour décoder : le processus de décodage se déroulera comme suite :

a. PGP emploie RSA pour déchiffrer la clé de session avec la clé privée de destinataire.
b. Il utilise de nouveau IDEA pour déchiffrer enfin le message avec la clé de session.

Résultats Expérimentaux :

L'exemple suivant montre le contenu du fichier crypté par PGP. on a utilisé le même texte employé auparavant par l'IDEA et RSA :

Clé privée	Clé publique	clé de session	Texte crypté
D=307	N=1339 E=307	Lakhdar_kamel_25	pgp010813330107010401001333011400950107133311390101010800950977005 3'□ø?R□cëw†KÁ§□®Jÿ&)ØGb<l °9ç ~(ÎAÝ9Xêx-^Œu´ÓÕ¼¼/

Tableau V.3 : Résultats obtenues avec le PGP

On remarque que le texte crypté commence par les trois lettres 'PGP' pour indiquer que ce dernier a été crypté par PGP , ensuite on trouve les chiffres qui représentent la clé de session , suivie par les données cryptées par l'IDEA.

4) **Implémentation de l'OTL :** L'ensemble des procédures et fonctions utilisées par l'OTL sont les suivantes : select, Fonctadapt, crossover, Generation, algorithm_genetique et cryptage.

La fonction **select** donne l'index qui correspond à l'individu sélectionné dans la population « pop ». L'algorithme utilisé pour la sélection est celui de la roue de lottrie [8].

```
function  select(pop:Tpopulation):byte;
var       rand,partsom:real;
          comp:byte;
          c:integer;
 begin
          partsom:=0;  c:=0;
          rand:=random*somfadapt;
          repeat
          c:=c+1;
          partsom:=partsom+pop[c].Fadapt;
          until ((c>=tailpop)or(rand<=partsom));
          select:=c;
end;
```

La fonction **Fonctadapt** calcule la valeur d'adaptation de l'individu indiv qui est le score qui prendra l'individu.

```
fonction  Fonctadapt(indiv:Tindivedu):integer;
 var      count: byte ;
          som: integer;
begin
          som:=0;
          for count:=1 a length(indiv) do
          som:=som+abs(ch_init[ord(str[count])].occu-
                         ch_init[ord(indiv_init[count])].occu);
          Fonctadapt:=som;
end;
```

Les opérateurs de croisement (crossover et mutation) sont programmés afin d'explorer l'espace de recherche du problème ,ces operateurs utilisent la fonction *flip[8]* (la pièce biaisée) pour décider s'il faut faire l'exploration ou non, cette fonction prend comme paramètre d'entrées les probabilités (pcross,pmut) et les considère comme face.

La procedure **crossover** prend comme paramètre deux parents *par1,par2 et* donne naissance à deux fils ,*fils1,fils2* (crossover en 1 point) *;*

```
procedure crossover (var par1,par2,fils1,fils2:Tinvidu ; pcross:real);
var       pointc,compt, pointcross:byte;
begin
          if flip(pcross) then
                    begin    tailchrom:=length(pa1);
                    repeat   pointc:=random(tailchrom)+1; until pointc>1;
                             pointcross:=pointc;
                             fils1:=copy(par1,1,pointc-1);
                             fils2:=copy(par2,1,pointc-1);
                    for compt:=pointc to length(par1) do
                     Begin  if  pos(par2[compt],fils1)<=0 then
                                      fils1:=fils1+par2[compt]
                     else     if pos(par1[compt],fils1)<=0 then       ar1[compt]
                               else
                             fils1:=fils1+par1[poszonecros(fils1,par1,pointc)];
                    if  pos(par1[compt],fils2)<=0 then
                             fils2:=fils2+par1[compt]
                    else
                     if pos(par2[compt],fils2)<=0 then
                             fils2:=fils2+par2[compt]
                     else     fils2:=fils2+par2[poszonecros(fils2,par2,pointc)];
          end;
                     end else
                     begin
                    fils1:=par1;
                    fils2:=par2;
                    end;
end;
```

la procedure **mutation** effectue une permutation aléatoire des deux gènes d'un fils .

```
procedure mutation(var indiv:Tindividu;pmut:real);
var      pointm1,pointm2,nbr:byte;
         permut: char;
begin
if flip(pmut) then
         begin
         nbr:=length(indiv);
          pointm1:=random(nbr)+1;
         repeat pointm2:=random(nbr)+1; until pointm1<>pointm2;
         permut:= indiv [pointm1];
         indiv [pointm1]:= indiv [pointm2];
         indiv [pointm2]:= permut;
         end;
end;
```

La procédure '**Generation'** permet la création d'une population *popfils1* à partir de la population *popparent1* en utilisant les operateurs de croisement et mutation .

```
procedure Generation(var  popfils1, popparent1 :Tpopulation);
var      sature,par1,par2,dotcross:byte;
         p1,p2,f1,f2:string;
begin
         sature:=1;
         repeat
                  par1:=select(popparent1);    par2:=select(popparent1);
                  p1:=pop1[par1].chrom;        p2:=pop1[par2].chrom;
                  //*   croisement  des parents sélectionnée  p1,p2 *//
                        crossover(p1,p2,f1,f2, pcross);
                  //*   mutation  des  fils  f1,f2 *//
                        mutation(f1,pmut);     mutation(f2,pmut);
                  with popfils1 [sature] do
                  begin chrom:=f1; part1:=par1; part2:=par2; end;
                  with popfils1 [sature+1] do
                  begin chrom:=f2; part1:=par1; part2:=par2; end;

                     sature:=sature+2;
          until (sature>=tailpop);
end;
```

Avec la procédure '**Evaluation'** on évalue l'adaptation de chacun des individus de la population *pop*, et on calcule la variable *somfadapt* qui est utilisée dans la roue de lottrie et dans d'autres statistiques telle que *la convergence de l'adaptation* des individus .

```
procedure Evaluation(var pop:Tpopulation;tailpop1:byte;var
                     som,max,bestmax:integer;var meil:byte);
var     count:byte;
begin
        max:=0; som:=0;
        for count:=1 to tailpop1 do
        begin
                pop[count].Fadapt:=Fonctadapt(pop[count].chrom);
                        som:=som+pop[count].Fadapt;
                if max <pop[count].Fadapt then
                begin
                        max:=pop[count].Fadapt;
                        meil:=count;
                end;
        end;
        if bestmax<max then bestmax:=max;
end;
```

La procédure 'algo_gene' sélectionne à partir d'une population initiale une population de solutions, cette procédure se répète jusqu'à ce que la compteur de génération dépasse un maximum *maxgen* dont le but est de trouver la meilleure solution.

```
procedure  algo_gene (var indiv_init, indiv_fin:Tindividu; popfils,
                                                popparent :Tpopulation);
var     gen :byte ;
begin
        bestmaxfadapt :=0 ;
        creation_popparent(indiv_init,popparent);
           Evaluation(popparent,tailpop,somfadapt,maxfadapt,
                                                bestmaxfadapt,meilleur);
        gen:=0;
        repeat
        Generation(popparent,popfils);
        Evaluation(popfils,tailpop,somfadapt,maxfadapt,bestmaxfadapt,
                                                        meilleur);
        popparent:=popfils ;
        inc(gen);
        until gen>=maxgene;.
        indiv_fin2:=popfils[meilleur].chrom;
end;
```

La procédure de chiffrement **cryptevolutionnist**, concatène touts les caractères constituant le texte à crypter pour former l' *indiv_init* puis fait appel au AG pour trouver l'*indiv_fin*, qui représente la solution qu'on cherche. La concaténation de ces individus constitue la clé de session à utiliser lors de décryptage, enfin chaque caractère de l' *indiv_init* sera substitué par leur correspondant dans l' *indiv_fin* .

```
procedure  cryptage(var buffer:array of char;var cle_session:string);
begin
        indiv_init
        charsfile (indiv_init);
        resltat de la
        algo_gene (indiv_init,indiv_fin);
                        cle_session:=indiv_fin+indiv_init;
        cryptfile;
end:
```

Présentation de l'interface de l'OTL :

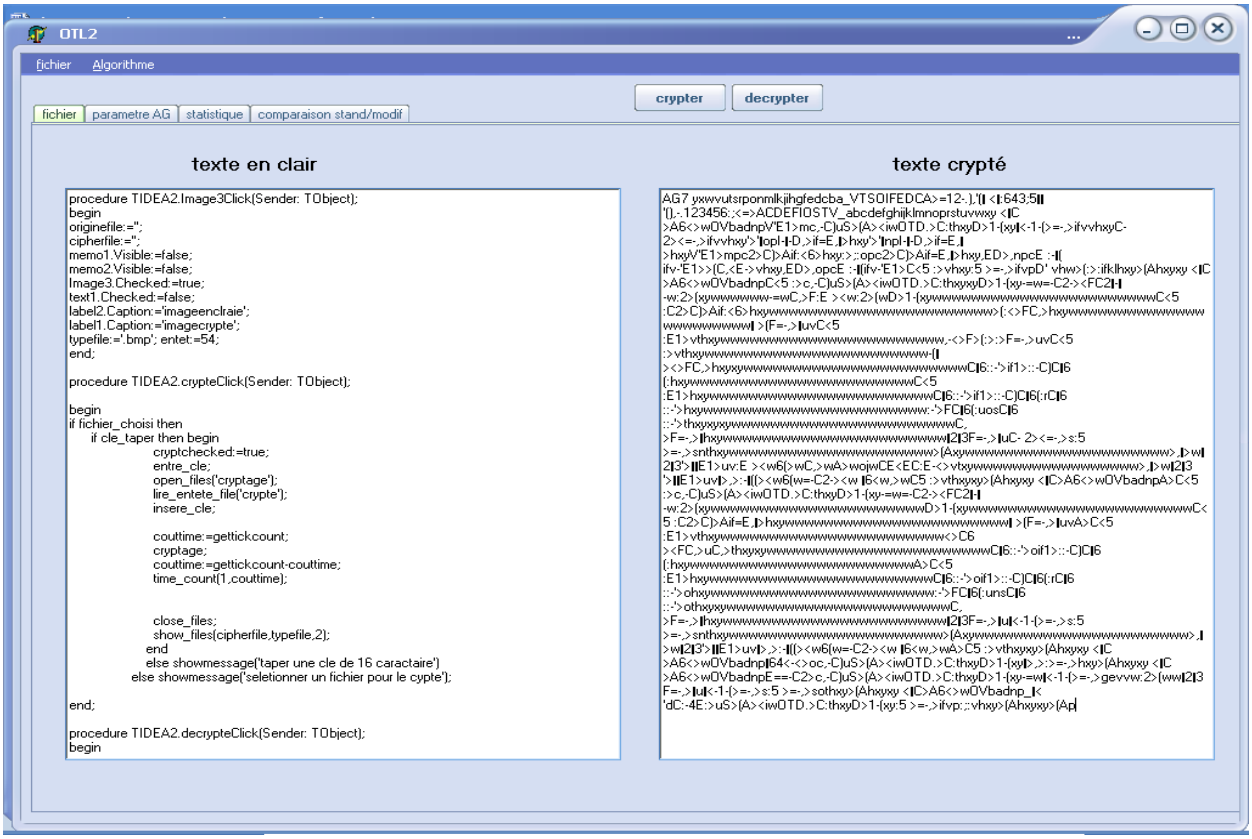


Figure V.5 : *Interface des fichiers de L'OTL*

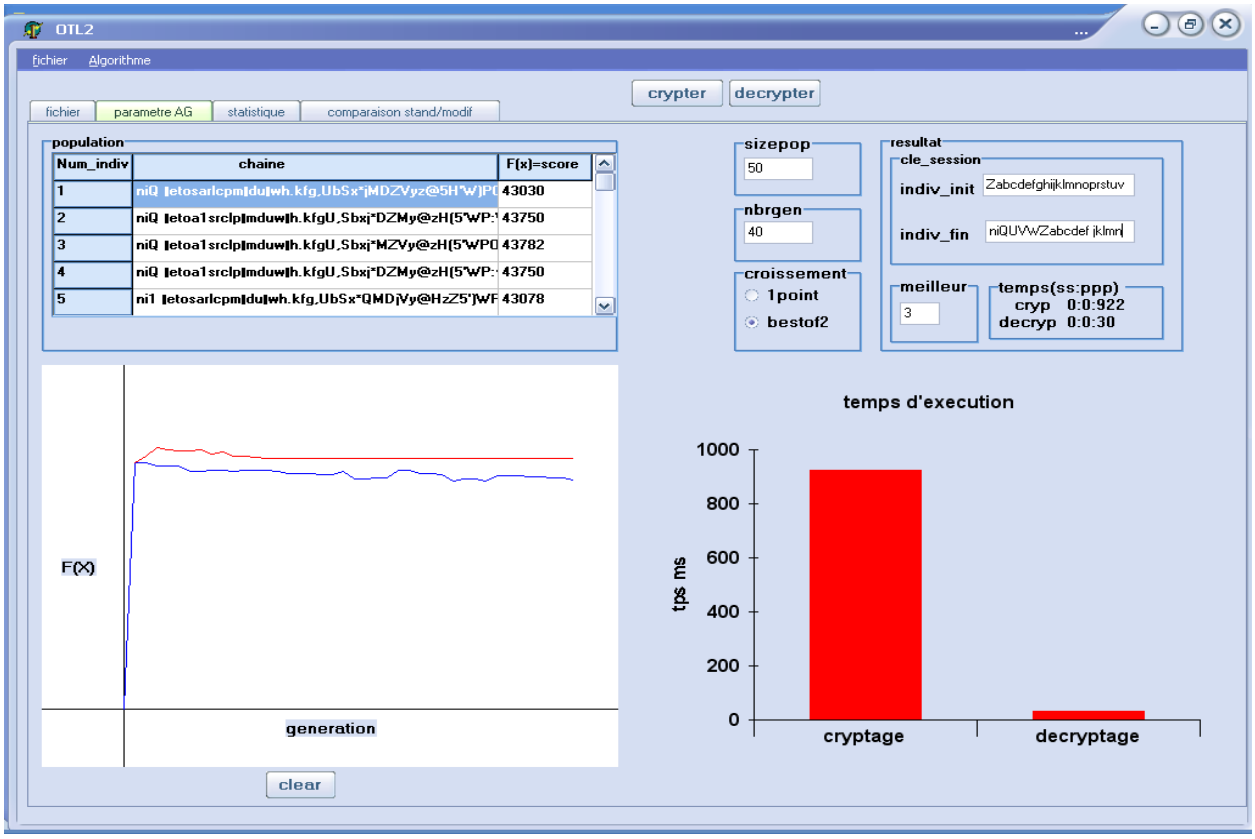


Figure V.6 : Interface des paramètres de l'AG de l'OTL

Fonctionnement :

Avec les algorithmes qu'on a vu, avant le cryptage l'utilisateur fait entrer la clé de session au programme ,mais dans le cas de l'OTL l'opération est inversée c'est le programme qui génère la clé après le cryptage pour l'utiliser lors de décryptage .

- **Pour coder** :

Il suffit de fixer les valeurs des paramètres de l'algorithme génétique comme la taille de la population *sizepop* et le nombre de génération *nbrgen* et de choisir le type de croissement *1point* ou *bestof2* (notons que le croissement *bestof2* est notre propre idée), Le cryptage sera lancé par un simple clic sur le bouton crypter et les informations suivantes sont affichées :

1. les individus de chaque population avec leurs fonction d'adaptation et l'indice du meilleur individu.
2. les individus *indiv_init* et *indiv_fin.*
3. le graphe du *convergence d'adaptation* des individus et le temps de (dé)cryptage.

- **Pour décoder** :

La clé de session est insérée dans le fichier crypté donc il suffit de cliquer sur le bouton décrypter pour que le programme récupère la clé et lance le décryptage.

Résultats Expérimentaux :

L'exemple suivant montre le cryptage du texte précédent avec l'OTL .

Paramètres AG		**clé de session**	**Texte crypté**
size pop	**nbrgen**		
10	5	*ytsrmlihgeda ponadeghilmnoprsty*	AG *ytsrmlihgeda pon adeghilmnoprsty*olr yipydehnyderymdsytesygdltgrsylipya dalrol

Tableau V.4 : Résultats obtenues avec l'OTL

Le fichier crypté débute toujours par 'AG' pour dire que ce dernier est crypté à l'aide des algorithmes genetiques , suivi par la clé de session et le texte.

Le fonctionnement des deux méthodes de croissement qui sont le *crossover en 1 point* , et le *crossover bestof2* qui est notre propre idée est représenté dans la figure V.7 . La deuxième méthode consiste à créer un enfant à partir de deux parents, de telle façon que l'enfant comporte seulement les meilleurs gènes de ses parents ,c'est à dire le iéme géne de l'enfant correspond au meilleur iéme géne de ses parents, mais avec le *crossover en 1 point* l'enfant peut contenir les mauvais et les meilleurs gènes , le fonctionnement des deux méthodes est présenté par les schémas suivants :

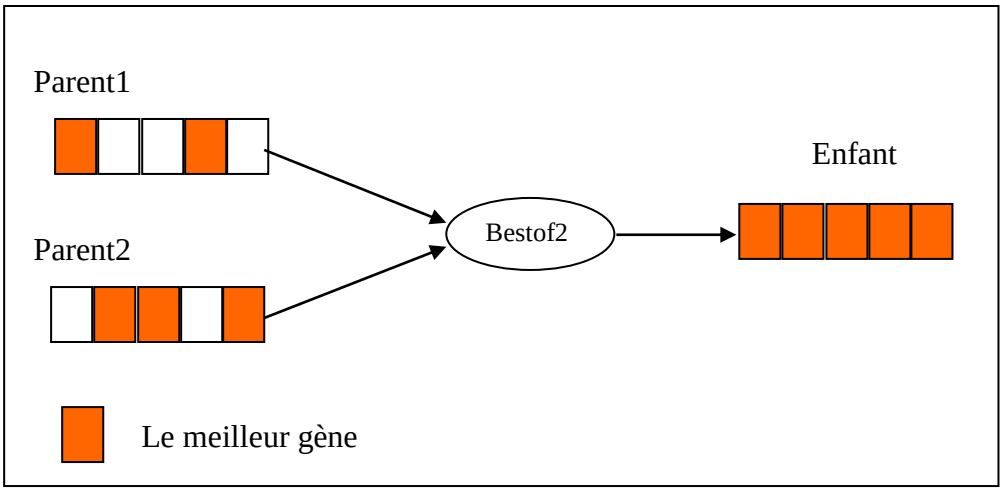


Figure V.7 : *Croisement par la méthode bestof2*

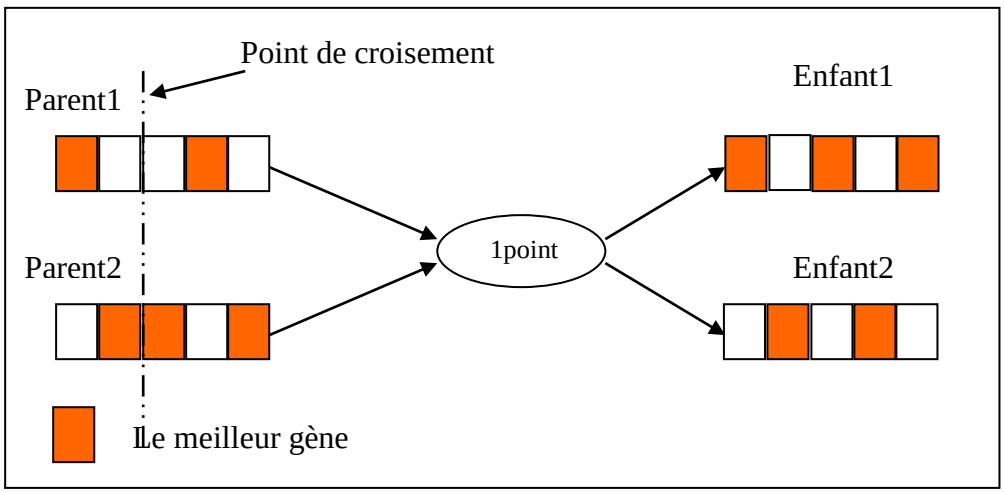


Figure V.8 : *Croisement en 1 point*

On constate que la convergence vers la solution est rapide et satisfaisante si on utilise le croissement *bestof2* figures V.8 et on déduit que ce croisement favorise une exploration rapide de l'espace de recherche du problème et donne des résultats meilleurs que celles de la méthode 1point.

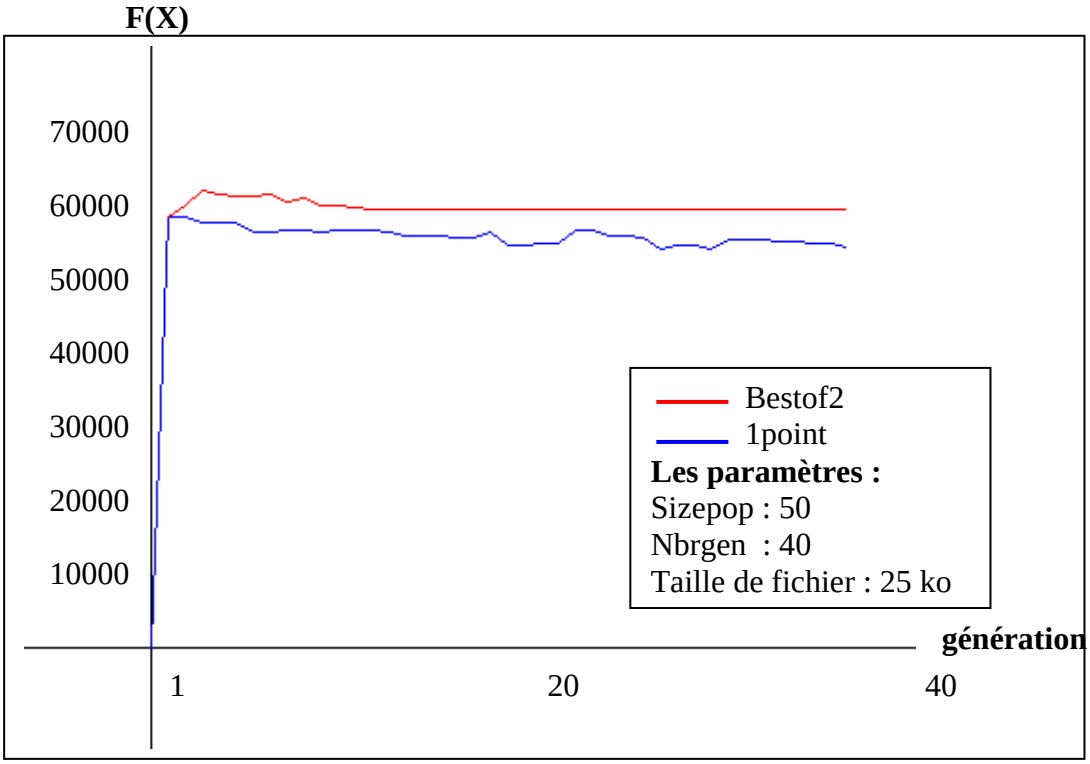


Figure V.9 :*Représentation graphique de la convergence vers la meilleure solution*

5) Implémentation de l'OTL modifié:

La différence majeure entre l'OTL et l'OTL modifié réside dans la substitution avec l'OTL un caractère est remplacé par un autre mais avec l'OTL modifié on peut substituer un caractère par un déterminant (mot fréquemment utilisé dans la langue française) ou l'inverse. Donc on peut dire que l'OTL modifié est l'extension de l'OTL standard.

Résultats Expérimentaux :

Le tableau suivant montre le cryptage d'un texte en clair par l'OTL modifié .

Texte en clair : " ***Si la confidentialité est mise hors la loi, seul les hors la loi profiteront de la confidentialité*** ".

Paramètres AG		**clé de session**	**Texte crypté**
size pop	**nbrgen**		
10	5	*éutsrpiedcaS hfmnol Sacdefhilmnoprstué*	AG éutsrpiedcaS hfmnolSacdefh ilmnoprstué udmsSidrpS les dtcd les flad de p e est de mc d de phcne est de mc d est id les p est S les oms SidrpS les dtcd les fé la

Tableau V.5 : Résultats obtenues avec l'OTL modifié

On remarque l'apparition des déterminants dans le texte ce qui indique qu'il y a eu des substitutions des caractères par ces déterminants et vice versa .

II. Comparaison :

II.1 Comparaison entre les algorithmes modernes et l' algorithme évolutionniste :

Le tableau de la **figure1** montre les tps de cryptage et décryptage des différents algorithmes d'un texte de taille de 16 kilo obtenus)

Les paramètres utilisés pour chaque algorithme :

IDEA : n'a aucun paramètre qui influe sur le tps du calcul .

RSA : le paramètre qui influe sur le tps du calcul est la valeur des nombres premiers p et q (le tps augmente suivant la valeur de p et q).

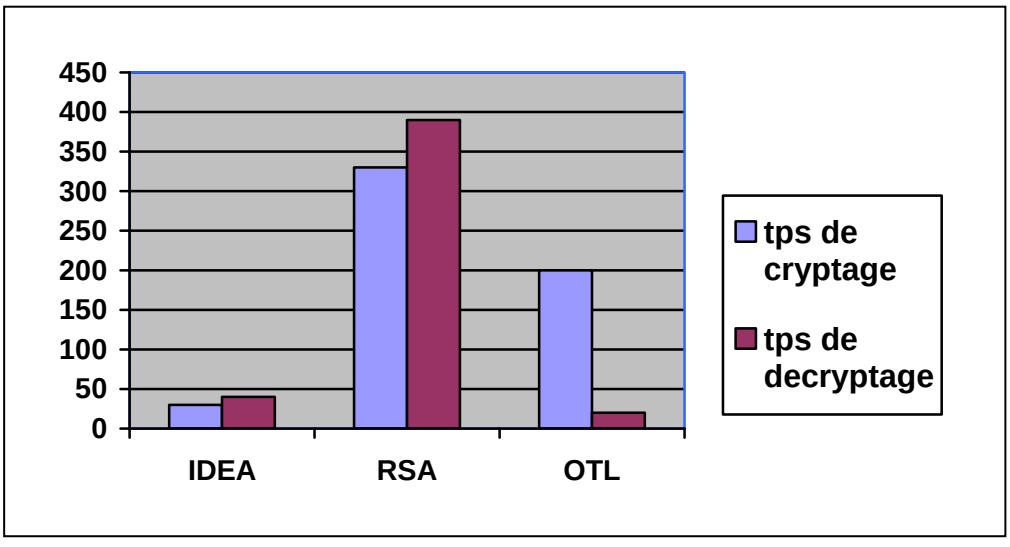
on a pris q= 103 , p= 13 .

OTL : les paramètres qui influes sur le tps du calcul est le nombre de génération et la taille de la population .

On a choisit taillepop =10 , nbrgene=10 .

	IDEA	RSA	OTL
tps de cryptage	30 ms	330 ms	200 ms
tps de décryptage	40 ms	390 ms	20 ms

Tableau V.6 :tps de cryptage et décryptage



FigureV.10 : *Représentation graphique de temps de cryptage et décryptage*

Discussions : On a vu que :

a. L'IDEA a un temps de cryptage et décryptage très rapide par rapport aux deux autres algorithmes mais l' inconvénient qui reste dans les algorithmes symétriques est comment faire circuler la clef privée de manière sûr et confidentielle . c'est un algorithme performant et difficile à casser .

b. RSA a le plus long temps de cryptage et décryptage par rapport aux deux autres algorithmes on peut le considérer comme un inconvénient, mais c'est un algorithme robuste sa clef privée est difficile à casser .

c. L'OTL a un temps de cryptage moyen par rapport au temps de l'IDEA et RSA , et un temps de décryptage plus rapide des deux autres . ce n'est qu'un algorithme de substitution mais ce qui le rond robuste et performant est le brouillage initial effectué sur le texte en clair à l'aide de l'un des algorithmes symétriques ou asymétriques . on peut dire que c'est un algorithme amélioré pour avoir un niveau de sécurité plus adéquat .

II.2 Comparaison entre l'OTL standard et modifié :

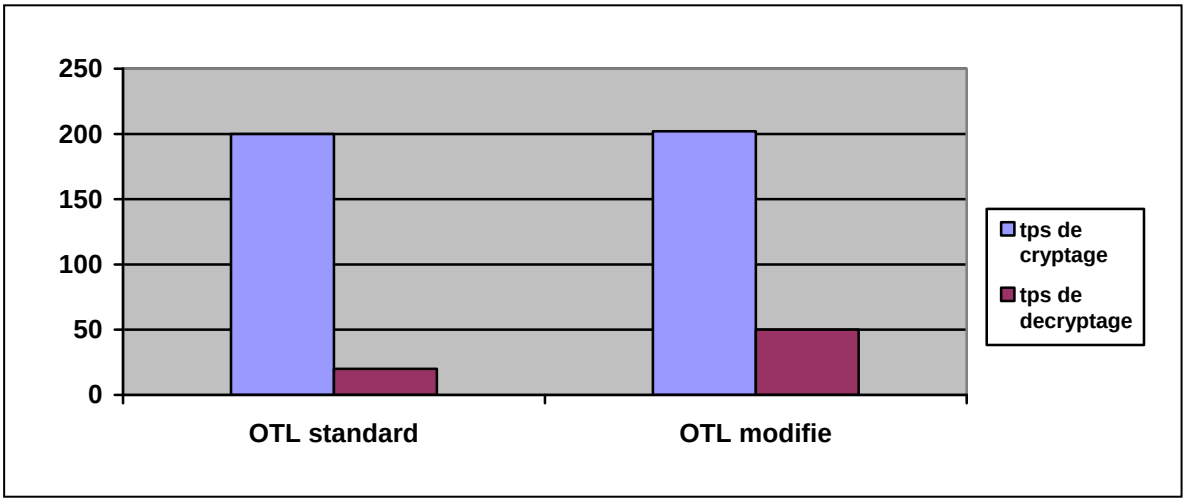
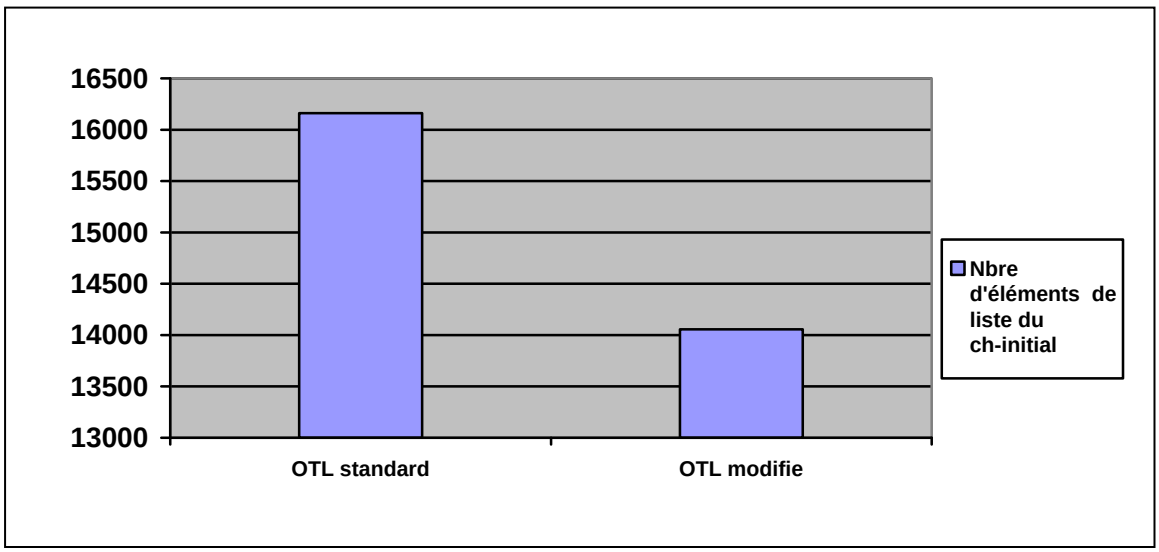
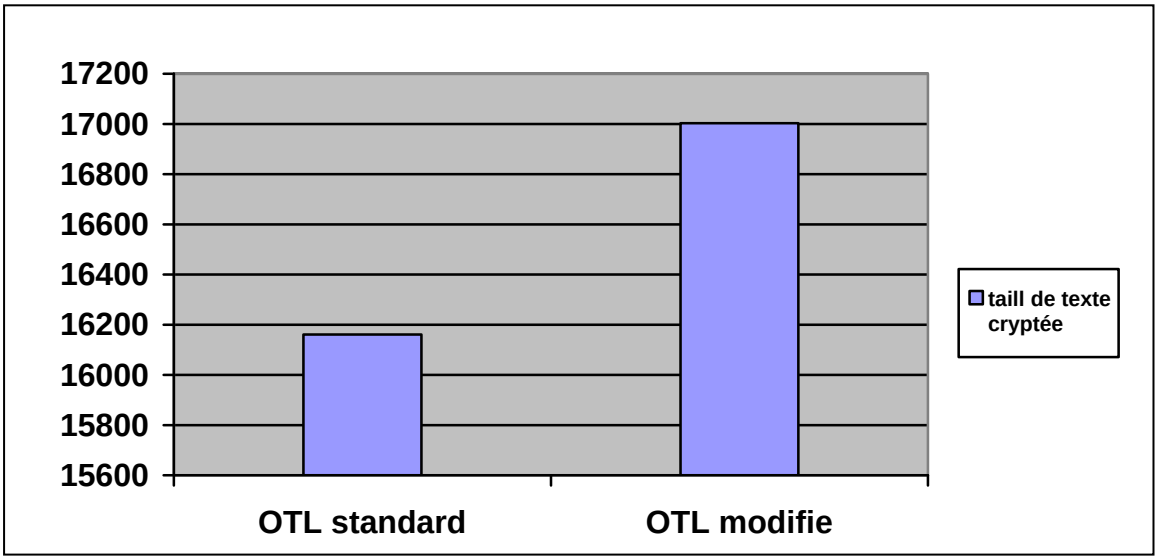


Figure V.11 : *Représentation graphique du temps de cryptage et décryptage de l'OTL Standard et modifié*



FigureV.12 : *Représentation graphique de nombre d'éléments de liste du ch-initiale de l'OTL standard et modifié.*

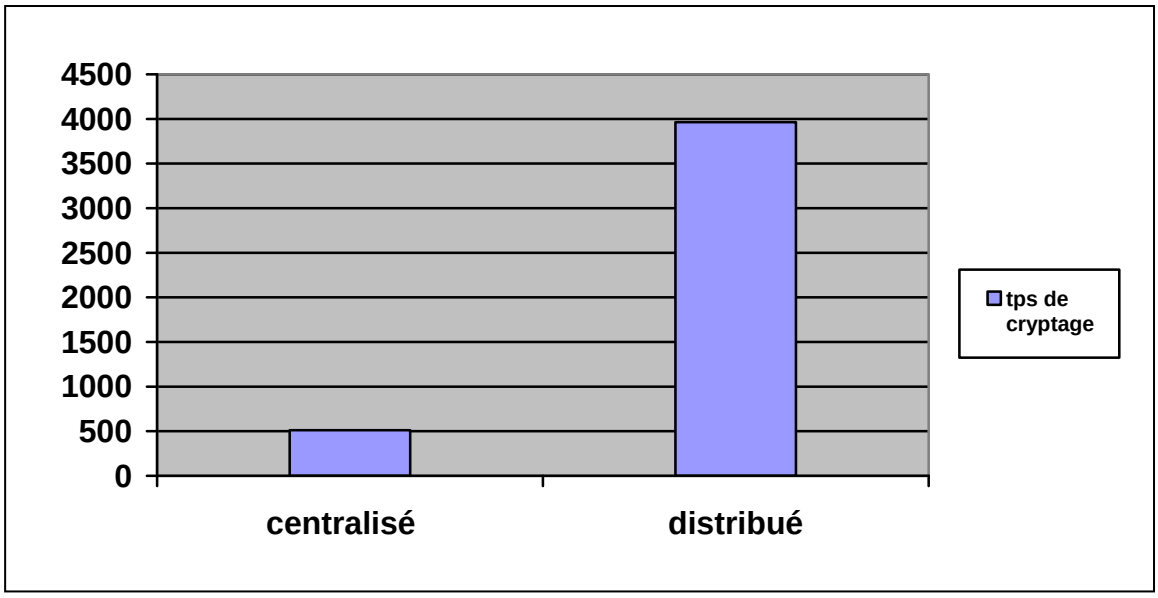


FigureV.13 : *Représentation graphique de la taille du texte crypté par l'OTL Standard et modifié*

II.3 Comparaison du cryptage centralisé et distribué par l'OTL :

Le but du cryptage distribué est de minimiser le temps nécessaire pour le chiffrement C'est à dire au lieu de chiffrer un texte par un seul ordinateur dans un temps Ts= k ms, on utilise n ordinateurs pour chiffrer le même texte (cryptage distribué). Le temps nécessaire est $Td=k/n+T_{tr}$ (T_{tr} est le temps de transfert) , pour bien exploiter le cryptage distribué il faut utiliser un nombre de postes de telle façon que le temps de transfert Ttr soit négligeable par apport au temps de traitement k/n .

On a utilisé deux ordinateurs pour le cryptage distribué figure (V.14)



FigureV.14 : *Représentation graphique du temps de cryptage centralisé et distribué par l'OTL*

Conclusion Générale :

Conclusion générale

Dans le contexte numérique, la cryptographie devient nécessaire à tous, particuliers, entreprises et administrations, afin de sécuriser et de préserver la confidentialité des échanges d'informations. Les progrès de l'informatique, logicielles ou matérielles ont permis de développer des solutions de cryptographie extrêmement puissantes. Depuis longtemps les mathématiciens se sont appropriés à la cryptographie pour en élaborer les bases théoriques.
Vu le grand succès des algorithmes évolutionnistes dans les problèmes d'optimisation, les chercheurs ont essayé de les combiner avec la cryptographie afin de réaliser un algorithme de chiffrement jouit de tous les avantages d'un algorithme évolutionniste.

Tout au long de notre travail nous avons :

1. Réalisé des cryptosystemes :
   - Modernes « IDEA, RSA, PGP » ce qui nous a permis d'enrichir nos connaissances sur un thème important de la sécurité informatique.
   - Evolutionniste « OTL (centralisé,distribué) ». qui nous a permis d'aborder à un sujet promettant de l'intelligence artificielle qui est les algorithmes génétiques et qui commence à être appliqué dans plusieurs domaines.

2. Proposé les approches suivants :
   - Une méthodes permettant d'acheminer d'une façon sécurisée la clé des cryptosystemes symétrique (tel que l'IDEA),elle consiste à caché la clé dans le message crypté pour cela en utilisant une fonction mathématique qui nous donne les positions des caractères de la clé dans le message crypté ,et lors de décryptage il suffit de récupérer la clé à partir de ce message (en utilisant la même fonction) et puis le décrypter.
   - Dans le domaine des algorithmes genetiques ,nous avons donné naissance à une méthode de croisement nommé « *le bestof2* » , autre que les méthodes standard tel que le croisement « en 1 point » qui manipule juste les chromosomes or notre approche s'intéresse au niveau des gènes.
   - L'algorithmes OTL modifier qui prend en conte les permutation entre les caractères et les articles .

3. Effectué différentes comparaisons ,entre:
    - Le croisement en 1 point et le bestof2.
    - Les cryptosystemes modernes et évolutionniste.
    - L'OTL standard et modifié.
    - L'OTL centralisé et distribué.

Les perspectives qu'on propose pour les étudiants voulant continué notre thème sont :

1. Simulation de fonctionnement de l'autorité de certification et les transactions entre plusieurs utilisateurs par le PGP .
2. Dans le futur il est possible de minimiser le temps de calcul par l'exécution des algorithmes dans un environnement distribué avec des algorithmes parallèles et d'améliorer l'OTL modifié.
3. Réalisation de la fonction de hachage .

Enfin comme information avec l'arrivée de l'informatique quantique dans un futur proche [doc4] (union entre l'informatique et la mécanique quantique) qui met en péril les cryptosystemes les plus robustes actuellement tel que (RSA) vu qu'ils sont capables d'effectuer des calculs assez grands dans un temps raisonnable, il est indispensable de penser à sécuriser en plus les systèmes de cryptographie et de penser à la cryptographie quantique.