ROYAUME DU MAROC
Université Mohammed V
- RABAT -
جامعة محمد الخامس
- الرباط -
Faculté des sciences
كلية العلوم

*CENTRE D'ETUDES DOCTORALES - SCIENCES ET TECHNOLOGIES*

N° d'ordre : **3522**

# THESE

En vue de l'obtention du **DOCTORAT**

**Structure de Recherche** : Intelligent Processing Systems & Security (IPSS)
**Discipline** : Informatique
**Spécialité** : Sécurité Informatique

**Présentée et soutenue le 25/09/2021 par :**

**Mohammed BOUGRINE**

## *Nouvelles Contributions Pour La Sécurité Informatique Par La Performance Du Chiffrement Evolutionniste*

**JURY**

| | | | |
|---|---|---|---|
| Fouzia BENABBOU | PES, | Université Hassan II, Casablanca. Faculté des Sciences Ben M'Sik | Présidente |
| Abderrahim TRAGHA | PES, | Université Hassan II, Casablanca. Faculté des Sciences Ben M'Sik | Rapporteur / Examinateur |
| Soumia ZITI | PH, | Université Mohammed V, Rabat. Faculté des Sciences | Rapporteur/ Examinatrice |
| Abdellah IDRISSI | PH, | Université Mohammed V, Rabat. Faculté des Sciences | Rapporteur / Examinateur |
| Ahmed EL YAHYAOUI | PA, | Université Mohammed V, Rabat. Faculté des Sciences | Invité |
| Fouzia OMARY | PES, | Université Mohammed V, Rabat. Faculté des Sciences | Directrice de thèse |

Année Universitaire : 2020/2021

# *Dédicace*

A Dieu le tout puissant

A mon prophète Mohammed (bénédiction et paix sur Lui)

A l'âme de mon chère inoubliable Papa,

A ma chère maman, que Dieu la protège,

A ma chère femme Salima et mes chers enfants Yahya et Ahmed,

A mes chères bonnes parant Khalil et Fatima

A ma chère Professeur Fouzia Omary

Aucun mot ne pourra exprimer mes sentiments envers vous.

A mes chères sœurs,

A mes chers frères,

Je ne sais comment vous remercier pour tout ce que vous avez fait pour moi.

A toute ma famille.

A tous ceux qui m'aiment.

A tous les musulmans à travers le monde, je dédie ce travail...

Mohammed

# *Remerciement*

Ce travail a été réalisé au sein de l'équipe Intelligent Processing Systems & Security (IPPS) à la Faculté des Sciences de l'Université Mohammed V de Rabat.

Je tiens à remercier Madame Fouzia BENABBOU, Professeur de l'enseignement supérieur à l'Université Hassan II-Mohammedia, Faculté des Sciences Ben M'Sik de Casablanca, d'avoir accepté de présider le jury. Je tiens à vous exprimer mes respectueuses considérations.

Mes remerciements les plus sincères vont également à Monsieur. Abderrahim TRAGHA, Professeur de l'enseignement supérieur à l'Université Hassan II-Mohammedia, Faculté des Sciences Ben M'Sik de Casablanca, d'avoir accepté d'examiner et de rapporter ce travail.

J'exprime mes salutations et mes remerciements à Madame Soumia ZITI, Professeur habilité à la Faculté des Sciences de Rabat, pour son soutien, sa motivation, son encouragement et d'avoir accepté de rapporter et de juger ce travail.

Mes vifs remerciements et mes respects s'adressent également à Monsieur Abdellah IDRISSI, Professeur habilité à la Faculté des Sciences de Rabat, pour son accompagnement, son soutien, son encouragement et pour avoir accepté d'être parmi le jury de cette thèse ainsi que d'examiner et de rapporter ce travail.

Je tiens à remercier chaleureusement Monsieur Ahmed EL YAHYAOUI, Professeur assistant à la Faculté des Sciences de Rabat, d'avoir accepté d'être parmi le jury de cette thèse ainsi que pour son soutien et sa disponibilité.

Aussi je tiens à remercier ma directrice de thèse, Madame Fouzia OMARY, Professeur de l'enseignement supérieur à la Faculté des Sciences de Rabat, qui m'a fait confiance et m'accepté au sein de son équipe de recherche et qui m'a accompagnée durant ces années de la préparation de cette thèse. Merci Professeur pour vos précieux conseils, votre temps et votre soutien moral, et surtout pour votre humanité.

Mes remerciements vont également à ma chère femme Salima TRICHNI pour ses conseils, encouragements, aide, pour tous ce qu'elle a fait pour moi, Merci encore une fois.

# *Résumé*

Dans la présente thèse, de nouvelles variantes du système de chiffrement SEC sont proposées. Ce dernier étant le premier système de chiffrement symétrique ayant comme outil de base les algorithmes évolutionnistes. Notre conviction sur la robustesse de ce système nous a incités à pousser nos recherches plus loin afin de trouver des résultats plus pertinents sur le chiffré et la clé générée par celui-ci. Le premier travail consiste en la conception et réalisation d'une nouvelle formule mathématique de la fonction d'évaluation étant donné l'importance qu'occupe cette étape dans un algorithme évolutionniste. Cette nouvelle fonction d'évaluation a permis de rendre nulle la probabilité pour qu'un caractère du message clair soit apparu dans ses positions initiales. Les expériences réalisées ont donné de très bons résultats en intégrant cette nouvelle fonction dans le système en question. Le deuxième travail présente la mise en place d'une nouvelle variante nommée ASEC, du système SEC qui a répondu largement aux attentes tracées au début de notre thèse. Et ce, en réalisant une reformulation de la majorité des outils du SEC en faisant appel à un problème difficile qui est le problème de Partition. Ainsi, la résistance du système vis-à-vis aux différentes attaques possibles a atteint un degré très élevé. La réussite de ce système est aussi prouvée par les expériences réalisées dans ce contexte. Notre dernière contribution aboutit à un nouveau système de chiffrement hybride qui utilise le système de chiffrement symétrique ASEC, et le système de chiffrement asymétrique RSA, les expériences réalisées ont donné des très bons résultats.

**Mots-clefs :** Sécurité, Confidentialité, Chiffrement hybride, Algorithmes évolutifs, Chiffrement symétrique, Cryptographie.

# *Abstract*

In this thesis, new variants of the SEC encryption system are proposed. The latter being the first symmetric encryption system with evolutionary algorithms as its basic tool. Our conviction on the robustness of this system prompted us to push our research further in order to find more relevant results on the cipher and the key generated by it. The first work consists of the design and realization of a new mathematical formula for the evaluation function given the importance of this stage in an evolutionary algorithm. This new evaluation function made it possible to nullify the probability that a character in the plain message appeared in its initial positions. The experiments carried out have given very good results by integrating this new function into the system in question. The second work presents the implementation of a new variant called ASEC, of the SEC system which largely met the expectations outlined at the start of our thesis. And this, by carrying out a reformulation of the majority of the tools of the SEC by appealing to a difficult problem which is the Partition problem. Thus, the resistance of the system to the various possible attacks has reached a very high degree. The success of this system is also proven by the experiences carried out in this context. Our last contribution results in a new hybrid encryption system which uses the ASEC symmetric encryption system, and the RSA asymmetric encryption system, the experiments carried out have given very good results.



**Keywords**: Security, Confidentiality, Hybrid Encryption, Evolutionary Algorithms, Symmetrical Encryption, Cryptography

# Table des matières

# Liste des tables

# Liste des figures

# Liste des abréviations

| | |
|---|---|
| ***RSA*** | Système de chiffrement asymétrique nommé par les initiales de ses trois inventeurs |
| ***DES*** | Data Encryption Standard |
| ***SEC*** | Symetrical Evolutionist-based Ciphering |
| ***AE*** | Algorithmes Évolutionnistes |
| ***ASEC*** | Advanced Symetrical Evolutionary Ciphering |
| ***Pgcd*** | Le Plus Grand Diviseur Commun |
| ***IDEA*** | International Data Encryption Algorithm |
| ***GPA*** | Générateur Pseudo-Aléatoire |
| ***POC*** | Problème d'optimisions combinatoire |
| ***NPC*** | Problème est NP-complet |
| ***TDES*** | Triple-Data Encryption Standard |
| ***XOR*** | Exclusive OR |

# INTRODUCTION

La naissance de la cryptologie remonte à l'antiquité : le plus vieux document chiffré retrouvé est une tablette d'argile datant du XVIe siècle avant J.-C. Les procédés de chiffrement par substitution sont apparus à cette période et ont été très largement utilisés depuis lors [1]. Au fil des siècles, les techniques cryptographiques se perfectionnent lentement et se complexifient.

L'utilisation de la cryptographie est jusqu'au XIXe siècle a été poussée par plusieurs motivations, mais les plus marquées ont été dans le domaine diplomatique et le domaine militaire, d'où l'apparition des dynasties de cryptographes comme les Rossignol en France, et les Da Porta en Italie. Au long de cette période la cryptographie est vue comme un art plutôt qu'une science [2] [3].

Au cours des années 1840 l'apparition du télégraphe a renforcé le besoin de sécuriser les messages, d'où la nécessité de commencer la réflexion sur les méthodes cryptographiques et la génération de la science de la cryptographie. Après 43 ans de cet essor, un livre a été publié par Auguste *Kerckhoffs* en 1883 intitulé « La cryptographie militaire » dans lequel, l'auteur a énoncé plusieurs propriétés pour concevoir un bon système de chiffrement. Par la suite nous citons les trois premières propriétés qui font preuve d'une étonnante modernité et qui sont toujours admises [4] :

- ✓ La première propriété, est que le système doit forcément être matériellement, sinon mathématiquement, indéchiffrable ;
- ✓ La deuxième, il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi ;
- ✓ La troisième dit que la clé doit pouvoir en être communiquée et retenue sans le secours de notes écrites, et être changée ou modifiée au gré des correspondants ;

La propriété "une" signifie que même s'il y a une façon théorique pour briser un système cryptographique, on peut le considérer comme un système sûr s'il n'existe pas de méthode qui

peut le casser en un temps raisonnable. Autrement dit, si la méthode pour casser un système de chiffrement nécessite de faire tourner le meilleur ordinateur pendant 1000 ans, on peut dans ce cas le considérer comme un système utilisable et sûr. Actuellement, la majorité des algorithmes reposent sur des problèmes difficiles. C'est-à-dire qu'ils ont des solutions mais impossible de les trouver rapidement ou même dans un temps raisonnable.

Pour la deuxième propriété, l'architecture d'un système cryptographique doit être connue, même pour un attaquant, et la seule chose qui doit être secrète c'est la clé, ce qui permet de garantir la sécurité du système en prouvant sa résistance contre les types d'attaques possibles.

Aussi la troisième propriété annonce que la clé du système doit être flexible et facilement changeable, c'est-à-dire pouvoir utiliser une clé différente à chaque échange, cela présente une difficulté de plus qui s'ajoute à la difficulté de concevoir un algorithme connu qui va résister aux attaques.

Cette difficulté a poussé les gens d'être très prudents lors de l'échange des clés. C'est d'ailleurs le problème majeur ayant lieu durant la seconde guerre mondiale. En fait, la difficulté de mettre à jour certaines clés dans les systèmes cryptographiques allemands a facilité la tâche des cryptanalystes Anglais. Sachant que les machines utilisées par les Allemands dans cette guerre sont les machines de chiffrement ***Enigma*** qui possèdent un jeu de rotors nécessitant une mise à jour quotidienne pour toutes les unités de l'armé Allemand et cela d'une manière synchronisée, ce qui a été difficile à réaliser. Par la suite les Anglais ont réussi à construire CLOSSUS, le premier ordinateur qui a vu le jour afin de casser les algorithmes de chiffrement Allemand et aussi c'était le jour de la rencontre entre la cryptographie et l'informatique [5].

***La cryptographie : d'un art à une science***

En 1949, et comme résultat de la guerre, Shannon a pu fonder la théorie de l'information qui a été le point de démarrage de la cryptologie moderne. C'est un moment très important dans l'histoire de la cryptologie. [6] [7]

Depuis, Le developpement des systèmes cryptographiques se déroulaient en trois phases, dans la premiere phase l'algorithme de chiffrement et la clé de chiffrement restaient tous les deux secréts. Dans la deuxième phase la clé reste secréte mais l'algorithme devient public, nous citons comme exemple ici le système de chiffrement DES. La troisième phase concerne les systèmes cryptographiques à clé public tel que le système RSA, dont l'algorithme et la clé de chiffrement sont publics et seule la clé privée (ou de déchiffrement) qui est secrete. [8] [9]

Le chiffrement symétrique DES est normalisé au court des années 70, ces années sont marquées aussi par la naissance de la cryptographie asymétrique par DIFFIE et HELLMAN [10], et le système RSA par R. L. Rivest, A. Shamir, et L. M. Adleman [11].

Généralement, un système cryptographique est acceptable, pour une application donnée, si trois conditions sont satisfaites :

- La première est la simplicité du processus de chiffrement et du déchiffrement.
- La deuxième est la complication d'utilisation des clés de chiffrement pour les utilisateurs non-initiés.
- Et la troisième, pas de liens entre le chiffrement et la confidentialité du chiffrement. En revanche, la sécurité d'échange de la clé secrète est toujours supérieure à celle d'un système.

### *Contributions*

Le but de nos contributions est de continuer les recherches sur la cryptographie évolutionniste et bénéficier de sa force déjà prouvée avec la naissance du système cryptographique évolutionniste (SEC) conçu par Professeur Fouzia OMARY [12] [13]. Nous rappelons que le SEC est un système de chiffrement symétrique basé sur les algorithmes évolutionnistes « Symetrical Evolutionist-based Ciphering », dont le but principal est de modifier les fréquences d'apparition des différents caractères du message clair ainsi que leurs positions, cela se fait à travers un Algorithme Évolutionniste mise en œuvre en 2005 [14].

Les Algorithmes Évolutionnistes (AE) s'inspirent de la théorie de Darwin sur l'évolution des espèces. Ces algorithmes travaillent sur une population initiale des individus, et leur force réside dans l'utilisation des fonctions objectives non régulières permettant l'exploration automatique des espaces de recherche très vaste, et non standard (par exemple : espaces de listes, de graphes, …). Ces algorithmes ont connu un grand succès dans la résolution de plusieurs problèmes réputés Difficiles et NP-Complet qui s'inscrivent dans le contexte des Problèmes d'Optimisation Combinatoire. Cependant, leur usage s'est limité exclusivement à la cryptanalyse. En effet, le crypto-système asymétrique de MERCLE basé sur le problème NP-Complet du Sac à Dos a succombé face à l'attaque par un processus basé sur les algorithmes génétiques par Spillman en 1993 et nous précisons que c'est pour la première fois que les algorithmes génétiques ont été utilisés en cryptologie. Aussi, et dans la même année les substitutions ont été attaquées par les AE. Vient ensuite le tour du RSA qui a subi aussi une tentative d'attaque par ce type d'algorithme.

En revanche, le SEC présente la première contribution exploitant les AE dans la cryptographie.

Cette thèse s'inscrit dans ce contexte, et cherche à proposer d'autres versions du système de chiffrement SEC en introduisant des nouvelles formules et outils mathématique afin de garantir sa performance et sa résistance à long terme.

Notre première contribution consiste à concevoir une nouvelle fonction d'évaluation pour le système, étant donné l'importance de cette partie dans un algorithme évolutionniste. En effet, la fonction d'évaluation constitue le cœur d'un système évolutionniste car elle permet de concrétiser toutes les propriétés qu'une solution doit satisfaire dans le système. De plus, elle permet de mener à bien la sélection des meilleurs individus et par conséquent avoir la solution optimale. Cela dit, la réussite d'un système utilisant les algorithmes évolutionnistes dépend essentiellement de la conception et réalisation de la fonction d'évaluation.

À travers une étude et une analyse approfondie sur la version initiale de la fonction d'évaluation du SEC, que nous avons eu la tâche de faire, nous avons constaté que la maximisation de la différence entre les fréquences d'apparition des caractères est une bonne condition pour valoriser nos solutions. Cependant, ceci ne permet pas d'éliminer la possibilité d'avoir une meilleure solution avec des caractères qui gardent les mêmes positions que dans le texte d'origine. Ainsi, nous avons mis en place une nouvelle fonction d'évaluation qui garantit les mêmes propriétés que l'initiale et qui satisfait, cette fois ci, la contrainte d'éliminer tous les cas dans lesquels un caractère occupe les mêmes positions dans le texte clair et dans le texte chiffré.

La deuxième contribution s'articule autour d'un nouvel axe de recherche qui consiste à édifier le système SEC sur un problème difficile, il s'agit ici du problème des Partitions. En effet, l'ensemble des listes des positions des caractères constituent une partition de l'ensemble des positions des caractères du texte clair. De ce fait, construire une partition dont les éléments sont des sous-ensembles ayant le même cardinal, aboutit incontestablement à un texte chiffré dont les fréquences d'apparition sont aléatoires. Et par conséquent, le système de chiffrement bâti sur une telle partition barrera la route contre l'attaque la plus redoutable, à savoir l'analyse des fréquences.

L'objectif de ce travail, est de concevoir de nouveau une autre fonction d'évaluation permettant de choisir la meilleure Partition à mettre en place pour remplir la condition de l'optimisation des modifications des fréquences d'apparition.

En fin, notre dernière contribution s'agit de mettre en place un nouveau système de chiffrement qui regroupe les deux types de chiffrement, symétrique et asymétrique. Cette hybridation a comme objectif, la proposition d'une nouvelle alternative de chiffrement qui assure le chiffrement des données et la sécurité de la communication des clé utilisées.

***Présentation du document***

Notre document comporte huit parties : une partie introductive au contexte du problème, trois parties définissant et étudiant les domaines où porte notre contribution, et quatre parties décrivent notre contribution et une dernière partie pour la conclusion.

L'organisation de ce document est faite de la façon suivante afin de faciliter le suivi et la compréhension du travail réalisé.

Dans l'introduction, nous décrivons le contexte général où se situe notre contribution, avec un petit flash sur celle-ci.

Le Chapitre 1 intitulé « *De la naissance d'une science aux différents systèmes cryptographique* » est consacré pour l'état de l'art, dont nous avons parlé de la cryptographie classique, moderne ainsi que quelques systèmes de chiffrement symétriques.

Le Chapitre 2 intitulé « *La sécurité des systèmes symétriques* », présente une vue globale sur la sécurité des systèmes symétriques et leur cryptanalyse.

Le Chapitre 3 intitulé « *Problème d'optimisation combinatoire* », dans lequel nous discutons les problèmes d'optimisations combinatoires et leurs résolutions, tout en rappelant quelques célèbres problèmes, ainsi que le problème de partition utilisé dans notre contribution.

Le Chapitre 4 intitulé *« Le système de chiffrement évolutionniste SEC »* dans lequel nous avons donné une description détaillée sur le SEC ainsi que leurs différentes étapes chiffrement, déchiffrement et tout le détail des opérations de ce dernier.

Le Chapitre 5 intitulé « *Conception de la nouvelle Fonction d'évaluation de SEC* », dans cette partie nous avons présenté notre nouvelle fonction d'évaluation et par conséquence, nous discutons notre nouvelle version du système de chiffrement évolutionniste SEC.

Le Chapitre 6 intitulé « *Nouvelle Fonction d'évaluation pour le système de chiffrement SEC intégrant le problème des Partitions* », dans ce chapitre nous présentons un nouveau

système basé sur le SEC et qui utilise le problème de partition qui un problème difficile sur la partie de chiffrement, ce qui a augmenté la résistance contre les types d'attaques possible.

Le Chapitre 7 intitulé « N*ouveau système de chiffrement hybride et étude de sa sécurité* » dans lequel nous présentons notre nouveau système de chiffrement hybride, qui est base sur le système de chiffrement symétrique ASEC et le système asymétrique RSA.

Le dernier chapitre a été consacré pour la conclusion et la perspective.

# CHAPITRE 1

# DE LA NAISSANCE D'UNE SCIENCE AU DIFFÉRENTS SYSTÈMES CRYPTOGRAPHIQUES

## Sommaire

## 1. Introduction

Depuis toujours, les codes ont existé. Ils présentaient un moyen pour retranscrire des idées, et formaliser un langage, ensuite ils sont utilisés pour cacher les informations et garder leur confidentialité, ce qui a donné naissance à la cryptographie, la science ou l'art de dissimuler ou cacher les messages, les textes, les informations etc. [15]

Aujourd'hui vu l'importance de l'information, il devient une nécessité de protéger cette information et de développer la science de la cryptographie qui assure cette protection. La concurrence n'est plus une concurrence de productivité mais une concurrence d'accès à l'information.

La cryptologie est une science qui inclut deux axes : la cryptographie et la cryptanalyse [16].

Dans le domaine de la cryptographie, on distingue deux grandes familles d'algorithmes : les algorithmes symétriques dites à clé secrète, et les algorithmes asymétriques ou à clé public, la différence entre les deux, est que la première nécessite le partage d'un secret par les deux protagonistes (qui vont partager l'info), et pour la deuxième (les algorithmes à clé publique), le secret reste connu d'un seul des deux acteurs. Cette distinction pourrait laisser penser que les systèmes symétriques seraient devenus dépassés dès le milieu des années soixante-dix, avec l'apparition de la cryptographie à clé publique. Et pourtant ils sont largement répandus car ils sont les seuls qui atteignent les débits de chiffrement requis par la plupart des systèmes informatiques.

La cryptographie symétrique est donc une discipline active ouverte à la recherche, sollicitée par un besoin industriel très pressant. [17]

## 2. La cryptographie classique

Dans la cryptographie classique une seule clé est utilisée pour chiffrer et déchiffrer les messages, seuls l'émetteur et le récepteur du message ont cette clé, et par conséquence la clé, la méthode de chiffrement et de déchiffrement sont connus par l'émetteur et le récepteur. Le principal inconvénient de ce type de système est le partage de cette clé unique entre les intervenants. [18].

La plupart des méthodes de chiffrement classiques reposent sur deux principes essentiels : la substitution et la transposition [19]. La transposition consiste à réparer les données de telle façon qu'elles soient inintelligibles. Tandis que la substitution consiste à remplacer des lettres par d'autres ou par des symboles.

Plusieurs méthodes de chiffrement se basent sur la substitution, la clé est simplement une permutation des caractères, pour les 26 caractères alphabétiques, le nombre des clés est 26 ! C'est-à-dire, plus que $4x10^{26}$ possibilités, donc impossible de rechercher la clé par une méthode exhaustive. (Mentionner qu'il existe d'autres méthodes pour la cryptanalyse de ce type de chiffrement). [20]

On distingue généralement entre plusieurs types de crypto-systèmes par substitution :

- ✓ **La substitution mono alphabétique** consiste à remplacer chaque lettre du message claire par une autre lettre de l'alphabet.
- ✓ **La substitution poly alphabétique** consiste à utiliser une suite de chiffres mono alphabétique réutilisée périodiquement.
- ✓ **La substitution homophonique** permet de faire correspondre à chaque lettre du message en clair un ensemble possible d'autres caractères.
- ✓ **La substitution de polygames** consiste à substituer un groupe de caractères (polygame) dans le message par un autre groupe de caractères.

Plusieurs systèmes cryptographiques ont étés mis en point durant des siècles jusqu'à nos jours.

### 1. Le code de César (101-44 avant JC)

Le code de Jules César est à la base de la cryptographie conventionnelle. L'algorithme consiste à décaler les lettres de l'alphabet par un certain nombre de position. Ce nombre des caractères décalés est donc la clé du système.

Par exemple, si vous codez le mot « CRYPTO » est la valeur de la clé de César est égale à 4, l'alphabet est donc décalé de manière à commencer à la lettre E.

Ainsi, l'alphabet : **ABCDEFGHIJKLMNOPQRSTUVWXYZ**

Si nous décalons le début des lettres par 4 lettres nous obtenons :

**EFGHIJKLMNOPQRSTUVWXYZABCD**

Où E = A, F = B, G=C etc.

Et par la suite le texte chiffré de notre exemple « CRYPTO » sera « GVCTXS ».

Le code de César n'admet que 26 façons différentes pour chiffrer un message, donc il est facile à le casser il suffit de tester toutes les possibilités.

## 2. Le chiffrement de Vigenère

Le chiffre de Vigenère est un système de chiffrement poly-alphabétique, c'est un chiffrement par substitution avec l'utilisation de 26 alphabets au lieu d'un seul décalés pour chiffrer le même message. Delà on peut dire que le chiffre de Vigenère est une amélioration du chiffre de César. Le chiffrement de Vigenère utilise la notion de la clé, une clé généralement est un mot ou une phrase utilisée pour chiffrer le message clair via une substitution effectuée entre chaque caractère du texte et une lettre de la clé. [21] [22]

**Exemple :**

| Texte clair | B | O | U | G | R | I | N | E | M | O | H | M | M | E | D |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Clé | C | R | Y | P | T | O | C | R | Y | P | T | O | C | R | Y |
| Décalage | 2 | 17 | 24 | 15 | 19 | 14 | 2 | 17 | 24 | 15 | 19 | 14 | 2 | 17 | 24 |
| Chiffré | D | F | S | V | K | W | P | V | K | D | A | A | O | V | B |

**Table 1: Exemple de chiffrement de Vigenère**

Donc le résultat du chiffrement de texte clair : « BOUGRINE MOHAMMED » tout en utilisant la clé « CRYPTO » est le chiffré « DFSVKWPVKDAAOVB » ce qu'on observe c'est que la même lettre peut être remplacée par différentes lettres par exemple le « M » dans notre exemple et remplacé par « K, A, O » ce qui rend la tâche d'un attaquant plus difficile.

Il est clair que ce chiffrement est plus sûr que le chiffrement de César, et pourtant, il a été facilement cassé par Friedrich Kasiski qui a publié en 1 863 une méthode pour déterminer la taille de la clé. Par conséquence, lorsqu'on utilise une clé avec une taille petite que celle des messages on peut déduire les caractères de la clé par un calcul de la fréquence d'apparition des lettres. Pour éviter ce problème, la solution consiste à choisir une clé dont la taille est proche de celle du texte à chiffrer, afin d'éliminer une étude statistique des lettres. On appelle ce type des systèmes de chiffrement : **Système à Clé Jetable**.

## 3. Le Carré de Polybe (env. 200-125 av. J.-C.)

Polybe écrivain grec, à l'origine du premier procédé de chiffrement par substitution. Le Carré de Polybe est un système de chiffrement mono-alphabétique basé sur une matrice carrée de 25 cases, et qui contient les lettres de l'alphabet sauf la lettre W pour la langue française et la lettre J pour l'Anglais. Carré Polybe = ($L_{ij}$) avec L la lettre de l'alphabet et i,j l'emplacement de cette lettre dans le tableau suivant : [23]

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | A | B | C | D | E |
| 2 | F | G | H | I | J |
| 3 | K | L | M | N | O |
| 4 | P | Q | R | S | T |
| 5 | U | V | X | Y | Z |

**Table 2: La grille de Polybe**

Donc chaque lettre $L_{ij}$ est codée par sa position dans le tableau ij, exemple pour coder la lettre A=11, O=35, Y=54 …

Comme exemple le nom « BOUGRINE » sera chiffré par «1235512243243415 ».

### 4. Le chiffrement de Delastelle

Le chiffre de Delastelle, décrit par le Français Félix-Marie Delastelle (1840-1902), en 1895, sous le nom de "cryptographie nouvelle", c'est un système de chiffrement par substitution et par transposition. La méthode de chiffrement consiste à commencer par un découpage du message en blocs de ***n*** lettres puis utiliser le carré de Polybe. Par exemple si on souhaite coder le message suivant : « Bienvenu » on procède comme suit :

On prend le n=5 ;

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| B | I | E | N | V | | E | N | U | | |
| 1 | 2 | 1 | 3 | 5 | | 1 | 3 | 5 | 0 | 0 |
| 2 | 4 | 5 | 4 | 2 | | 5 | 4 | 1 | 0 | 0 |

**Table 3: Chiffrement de Delastelle**

Ensuite, on effectue une transposition qui consiste à regrouper les chiffres deux par deux, de la droite vers la gauche et du haut vers le bas et on obtient :

12 13 52 45 42 13 50 05 41 00

La dernière étape dans ce procès consiste à utiliser une autre fois le carré de Polybe pour transformer les chiffres en lettres. (Remarque : pour les chiffres 50 05 00 on les remplace par des chiffres ou par des caractères spéciaux, 50→5, 05→@, 00→%)
Finalement on trouve le message chiffré suivant : **BCVTQC5@P%**

### 5. Le chiffrement affine

Le chiffrement affine est un chiffrement par substitution qui se base sur une fonction F(x) de la forme suivante :

$$F(x) = ax+b \ \textbf{mod}\ 26 \quad \text{ou } a, b \in Z_{26}$$

Cette fonction est appelée fonction affine d'où on a tiré le nom de ce procédé. Pour $\boldsymbol{a = 1}$ on retrouve le chiffrement par décalage.

Pour que le déchiffrement soit possible il faut que cette fonction admette une et une seule solution c'est-à-dire il faut qu'elle soit injective.

Il suffit de montrer que l'équation $\mathbf{ax = y\text{-}b \bmod 26}$ admet une et une seule solution est cela n'est vrai que si et seulement si le **pgcd (a, 26)=1.**

Pour le déchiffrement si $\boldsymbol{y= ax+b \bmod 26}$ admet une seule solution, et si on fait appel à l'inverse modulaire, c.-à-d. l'inverse de **a** est $\mathbf{a^{-1}}$**.**

Alors la fonction de déchiffrement sera : $\boldsymbol{x= a^{-1}.y\text{-}b \bmod 26}$.

### 6. Le chiffrement de Hill

Le chiffrement de Hill, inventé en 1929 par Lester S. Hill, se rapproche beaucoup du chiffrement affin. Il consiste à déterminer les caractères du texte chiffré par combinaison linéaire de **m** caractères du texte en clair [19].
On décide de chiffrer le message DCODE : 3, 2, 14, 3, 4 chiffré avec $\begin{pmatrix} 3 & 4 \\ 5 & 9 \end{pmatrix}$

$\mathbf{3\times3+4\times2 =17 =R}$ et $\mathbf{5\times3+9\times2 = 33 \equiv 7 \bmod 26}$, et ainsi de suite.
Si le nombre de lettre est impair, on rajoute une lettre quelconque à la fin du message clair par exemple ici la lettre X. Ici on obtiendra obtenir RHCTAT comme message codé.

### 7. Le chiffre des templiers (ou de PigPen)

Certain ont préféré de remplacer chaque lettre par un symbole au lieu de la remplacer par une autre lettre pour diffuser des messages codés.

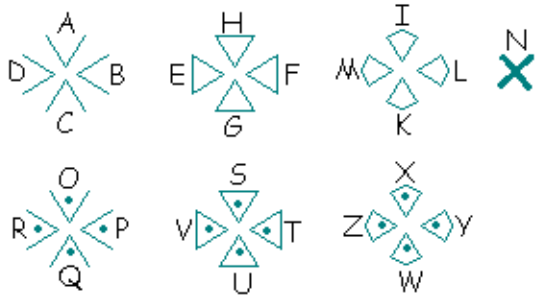
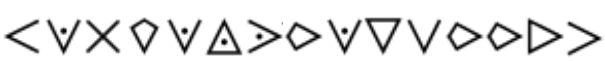



**Figure 1: Chiffrement de PigPen**

Par exemple le message codé du message suivant « bonjour Mohammed » est :



## 8. Les faiblesses de la méthode de substitution

Dans le chiffrement par substitution, la clé est simplement la permutation des 26 lettres alphabétiques. Le nombre de clés est 26! Une recherche exhaustive de la clé est donc impossible. Cependant, le chiffrement par substitution possède une grosse faiblesse structurelle [24]. Une étude simple de différentes langues va nous montrer que chaque lettre a une fréquence d'apparition dans une langue, par exemple dans la langue française la fréquence d'apparition de la lettre E est plus grande que celles des autres lettres, par contre la lettre W et la moins apparu que toutes les autres. Or, que ces lettres sont toujours remplacées par les mêmes lettres.

### Les fréquences d'apparition des lettres dans la langue Française

Le tableau ci-après représente les lettres et leurs fréquences d'apparition dans la langue française :

| Par ordre alphabétique | | | | | |
|---|---|---|---|---|---|
| **A** | **8.4 %** | **K** | **0.1 %** | **T** | **7.1 %** |
| **B** | **1.1 %** | **L** | **6.0 %** | **U** | **5.7 %** |
| **C** | **3.0 %** | **M** | **3.0 %** | **V** | **1.3 %** |
| **D** | **4.2 %** | **N** | **7.1 %** | **Y** | **0.3 %** |
| **E** | **17.3 %** | **O** | **5.3 %** | **Z** | **0.1 %** |
| **F** | **1.1 %** | **P** | **3.0 %** | **W** | **0.1 %** |
| **G** | **1.3 %** | **Q** | **1.0 %** | **X** | **0.4 %** |
| **H** | **0.9 %** | **R** | **6.6 %** | | |
| **I** | **7.3 %** | **S** | **8.1 %** | | |
| **J** | **0.3 %** | | | | |

**Table 4: Fréquence d'apparition des lettres dans la langue française**

**Fréquence d'apparition des lettres dans la langue Anglaise**

Pour information les lettres et les fréquences les plus courantes en langue anglaise sont :

| Par ordre alphabétique | | | |
|---|---|---|---|
| **A** | **8.2 %** | **N** | **6.7 %** |
| **B** | **1.5 %** | **O** | **7.5 %** |
| **C** | **2.8 %** | **P** | **1.9 %** |
| **D** | **4.3 %** | **Q** | **0.1 %** |
| **E** | **12.7 %** | **R** | **6.0 %** |
| **F** | **2.2 %** | **S** | **6.3 %** |
| **G** | **2.0 %** | **T** | **9.1 %** |
| **H** | **6.1 %** | **U** | **2.8 %** |
| **I** | **7.0 %** | **V** | **1.0 %** |
| **J** | **0.2 %** | **W** | **2.4 %** |
| **K** | **0.8 %** | **X** | **0.2 %** |
| **L** | **4.0 %** | **Y** | **2.0 %** |
| **M** | **2.4 %** | **Z** | **0.1 %** |

**Table 5 : Fréquence d'apparition des lettres dans la langue Anglaise**

Les deux tableaux montrent qu'il y a des groupes de lettres qu'ils ont plus ou moins la même probabilité d'apparition. Pour la langue française on voit que le pourcentage de la lettre E est loin des autres ce qui donne une grande chance que la lettre la plus fréquente dans un texte chiffré est en fait un E. Ensuite les lettres A, I, N, …

Toute en étudiant les groupes de deux lettres comme DE, TE, SE, ET, …

## 9. Les systèmes à clés jetables

Les systèmes à clés jetables sont apparus pour éviter les attaques statiques, d'où l'idée d'utiliser une substitution qui rend le texte chiffré non analysable. Cette idée des systèmes à clés jetables consiste à produire à chaque fois une clé composée d'une suite binaire aléatoire afin de chiffrer le message en appliquant le « ou exclusive » (XOR) entre le message et la clé. [25]

Ce chiffre est incassable vu que chaque clé n'est utilisée qu'une seule fois, malgré ça deux inconvénients sont présents, la taille de la clé et le problème de synchronisation. [26]

## 3. Cryptographie moderne

La cryptologie moderne étudie les méthodes qui permettent de garantir les services d'intégrité, d'authentification et de confidentialité dans les systèmes d'information et de communication. Elle protège aujourd'hui également l'ensemble des procédés informatiques devant résister à des attaquants [27]. [28]

La cryptographie moderne est composée de deux grandes parties, la cryptographie symétrique et la cryptographie asymétrique. [29]

### 1. La cryptographie symétrique

La cryptographie symétrique, aussi appelée la cryptographie à clé secrète ou encore la cryptographie conventionnelle, est la plus ancienne forme de chiffrement, utilisée par les Égyptiens vers 2000 ans avant J-C.

Les algorithmes de chiffrement à clé secrète sont ceux pour lesquels l'émetteur et le destinataire partagent une même clé secrète, c'est-à-dire que, les clés de chiffrement et de déchiffrement sont identiques. L'emploi d'un algorithme à clé secrète lors d'une communication nécessite donc l'échange d'un secret entre les deux protagonistes à travers un canal sécurisé. [30] [31]

On distingue dans les algorithmes symétriques entre deux catégories : le chiffrement par bloc, et le chiffrement continu. Dans le premier, on décompose le texte clair en blocs de même taille, chaque bloc est chiffré individuellement mais avec la même manière. Pour le second, chaque caractère du message en clair est chiffré un par un. [32]

Les systèmes de chiffrement symétrique sont nombreux, mais peu entre eux résident longtemps, parmi les plus mûrs on cite par exemple : DES, AES, IDEA, … [33]

#### 1.1. Chiffrement par bloc

Le chiffrement par blocs est une fonction $\boldsymbol{E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n}$, cette notation signifie que la fonction ***E*** prend deux entrées, une chaine de ***k*** bits, et une autre chaine de ***n*** bits, et qui donne comme résultat, une chaine de ***n*** bits. La première entrée est la clé, la seconde est le texte clair, et la sortie représente le texte chiffré. La longueur de la clé '***k'*** et la longueur du bloc '**n**'sont les paramètres associés au chiffrement par bloc. Ces paramètres varient d'un algorithme de chiffrement à un autre, bien entendu fait partie de la conception de l'algorithme lui-même. [34]

Pour chaque clé $k \epsilon \{0,1\}^k$, soit la fonction $E_k : \{0,1\}^n \rightarrow \{0,1\}^n$ définie par $E_k(M) = E(k, M)$. Pour n'importe quel algorithme de chiffrement par bloc, et une clé $k$, il est nécessaire que la fonction $E_k$ soit une permutation sur $\{0,1\}^n$. Cela signifie que c'est une bijection (c.à.d. une fonction bijective) de $\{0,1\}^n$ vers $\{0,1\}^n$. (Pour chaque $C \epsilon \{0,1\}^n$; il existe un et un seul $M \epsilon \{0,1\}^n$ tel que $E_K$ (M) = C. En conséquence $E_K$ admet un inverse, et on le note $E_k^{-1}$.

Cette fonction définit sur $\{0,1\}^n \rightarrow \{0,1\}^n$, comme suit:

$$E_k^{-1}(E_k(M)) = M \text{ et } E_k(E_k^{-1}(C)) = C \quad \text{pour tous M et C de } \{0,1\}^n,$$

donc la fonction $E_k^{-1} : \{0,1\}^n \rightarrow \{0,1\}^n$ définit par $E_k^{-1}(k, C) = E_k^{-1}(C)$ est appelé inverse de $E$.

Le chiffrement par bloc $E$ est un algorithme public bien spécifié. Le chiffrement $E$ et son inverse $E^{-1}$ doivent être facilement calculable, c'est-à-dire si on a le message M et la clé k on peut facilement calculer $E(k, M)$, et si on a la clé k et le chiffré C, on calcule facilement $E^{-1}(k, C)$. Dans le cas d'une utilisation normale, une clé aléatoire k est choisie et reste secrète entre deux utilisateurs, la fonction $E_k$ est ensuite utilisée par les deux parties pour traiter les données d'une certaine façon avant de l'envoyer à l'autre utilisateur. En général, nous supposerons que l'adversaire peut obtenir des exemples d'entrées-sorties pour $E_k$, c'est-à-dire des couples de la forme **(M, C)** où $C = E_k(M)$. Mais, le plus important est que l'adversaire ne sera pas capable d'avoir la clé secrète $k$. ***La sécurité repose sur le secret de la clé***. Donc, la première chose est de penser au but de l'adversaire qui est la récupération de la clé secrète $k$ tout en utilisant les entrées-sorties de la fonction $E_k$. Le chiffrement par bloc doit être bien conçu pour rendre cette tâche très difficile à calculer. (Plus tard, nous allons voir que la sécurité contre la récupération des clés est une condition nécessaire mais pas suffisante pour la sécurité d'un chiffrement par bloc.)

### 1.1.1. Chiffrement DES

Le système ***DES*** est conçu par IBM au début des années soixante-dix (Lucifer), après il a été repris et modifié par la NSA (National Security Agency). Ce chiffrement a été retenu en 1977 par l'U.S. *National Bureau of Standards* pour protection des données informatiques de toutes organisations fédérales. [35] [36].

Le D.E.S (Data Encryption system) est un système de chiffrement par blocs de 64 bits, dont 8 bits pour le contrôle de parité (un bit pour chaque octet). Ce système est un cas spécial d'algorithme de chiffrement appelé « schéma de Feistel ». Donc, pour pouvoir décrire ce système il faut tout d'abord se familiariser avec les définitions suivantes :

### a) Définitions

**Définition 1 : Le chiffrement par produit** combine un certain nombre de transformations ayant la même structure, d'une manière à ce que le chiffré obtenu soit plus sûr que les composants individuels.

**Définition 2: Le réseau de substitution-permutation (SP)** est un algorithme de chiffrement de produit, composé d'un certain nombre d'étapes appliquant chacune une substitution et une permutation.

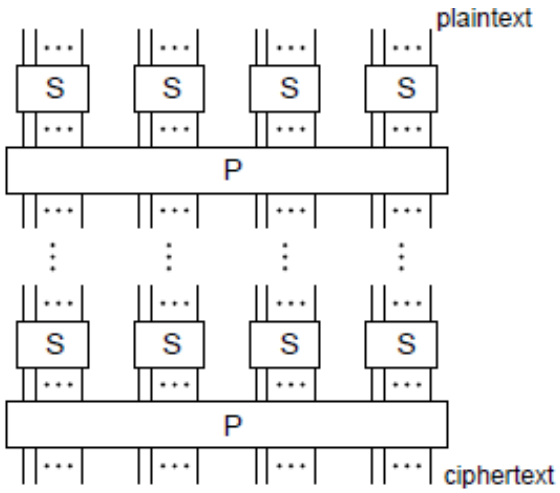



**Figure 2: Réseau de substitution-permutation (SP)**

**Définition : Le chiffrement par blocs itéré** est un chiffrement par blocs impliquant la répétition séquentielle d'une fonction interne appelée fonction tours (round). Les paramètres de ce chiffrement incluent le nombre de tours 'r', la taille en bits du bloc 'n', et 'k' la taille en bits de la clé K à partir de laquelle les entrées r (sous-clés Ki clés rondes) sont dérivés. Pour l'invisibilité (permettant le décryptage unique), pour chaque valeur de Ki de la fonction ronde est une bijection sur l'entrée ronde.

**Définition : Le chiffrement de Feistel** est un chiffrement par bloc avec itération de r tours qui donne en sortie un bloc chiffré ($R_r$ , $L_r$) de taille 2n à partir d'un bloc clair en entrée de 2n bits ($L_0$ , $R_0$), à travers un processus de r tours avec r ≥1. [37] [38]

Pour chaque tour i avec 1≤ i ≤ r transforme $(L_{i-1} ; R_{i-1})$ en $(L_i , R_i)$ de la manière suivante :

$$\begin{cases} L_i = R_{i-1} \\ R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \end{cases}$$

Où chaque sous-clé $K_i$ est dérivée de la clé de chiffrement $K$. Typiquement, dans un chiffrement de Feistel, $r \geq 3$ est souvent le même. La structure de Feistel demande que le texte chiffré en sortie soit dans l'ordre $(R_r , L_r)$ plutôt que $(L_r , R_r)$, les blocs sont échangés de

leur ordre habituel après le dernier tour. Le déchiffrement est ainsi réalisé en utilisant le même processus de r-tour, mais avec des sous-clés utilisées dans l'ordre inverse, de $K_r$ à $K_1$ [39].

### b) Description de DES

***DES*** est un algorithme de chiffrement symétrique basé sur le schéma de Feistel qui traite des blocs de texte en clair de n = 64 bits pour produire des blocs de texte chiffré de 64 bits.

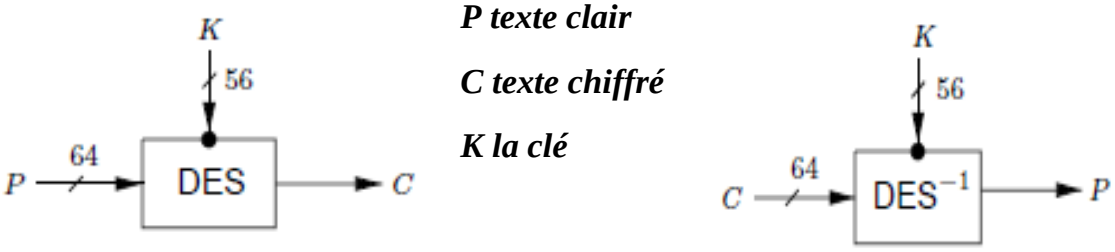



**Figure 3: DES entrée-sortie**

Le chiffrement se déroule en 16 étapes ou rondes. À partir de la clé en entrée ***K***, seize sous-clés de 48 bits $K_i$ sont générées, une pour chaque tour. Dans chaque tour, 8 sont fixés, et puis on choisit soigneusement les schémas des 6 à 4 substitutions de bits (S-boites) qu'on nomme ***Si***, collectivement notées ***S***. Le texte clair de 64 bits est divisé en deux moitiés de 32 bits $L_0$ et $R_0$. Chaque tour est fonctionnellement équivalent à l'autre, prenant en entrée les 32 bits $L_{i-1}$ et $R_{i-1}$ du tour précédent et produit en sortie 32 bits représentant $L_i$ et $R_i$ pour $1 \leq i \leq 16$, on a donc :

$$L_i = R_{i-1};$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \text{ ; où } f(R_{i-1}, K_{ij} = P(S(E(R_{i-1}) \oplus K_i)$$

Ici **E** est une application d'expansion déterminée à l'aide de la permutation $R_{i-1}$ passant de 32 à 48 bits (tous les bits sont utilisés au moins une fois).

P est une autre permutation déterminée sur 32 bits. Une permutation initiale des bits IP précède en premier tour et après la dernière ronde, les moitiés gauche et droite sont échangées. Enfin, la chaine résultante est peut-être permutée par l'inverse de la permutation

IP. Le déchiffrement comporte la même clé de l'algorithme, mais avec des sous-clés appliquées aux tours internes dans le sens inverse.

| IP | | | | | | | |
|---|---|---|---|---|---|---|---|
| 58 | 50 | 42 | 34 | 26 | 18 | 10 | 2 |
| 60 | 52 | 44 | 36 | 28 | 20 | 12 | 4 |
| 62 | 54 | 46 | 38 | 30 | 22 | 14 | 6 |
| 64 | 56 | 48 | 40 | 32 | 24 | 16 | 8 |
| 57 | 49 | 41 | 33 | 25 | 17 | 9 | 1 |
| 59 | 51 | 43 | 35 | 27 | 19 | 11 | 3 |
| 61 | 53 | 45 | 37 | 29 | 21 | 13 | 5 |
| 63 | 55 | 47 | 39 | 31 | 23 | 15 | 7 |

| $IP^{-1}$ | | | | | | | |
|---|---|---|---|---|---|---|---|
| 40 | 8 | 48 | 16 | 56 | 24 | 64 | 32 |
| 39 | 7 | 47 | 15 | 55 | 23 | 63 | 31 |
| 38 | 6 | 46 | 14 | 54 | 22 | 62 | 30 |
| 37 | 5 | 45 | 13 | 53 | 21 | 61 | 29 |
| 36 | 4 | 44 | 12 | 52 | 20 | 60 | 28 |
| 35 | 3 | 43 | 11 | 51 | 19 | 59 | 27 |
| 34 | 2 | 42 | 10 | 50 | 18 | 58 | 26 |
| 33 | 1 | 41 | 9 | 49 | 17 | 57 | 25 |

**Table 5: Permutation initiale et inverse du DES (IP and $IP^{-1}$)**

Une vue simple, est que la moitié droite de chaque tour (après l'expansion de l'entrée 32 bits à 8 caractères de 6 bits chacun) effectue une substitution qui dépend d'une clé sur chacun des 8 caractères, et qui utilise ensuite une transposition fixe pour redistribuer les bits des caractères résultants pour produire les 32 bits de sortie.

L'algorithme (c) montre comment calculer les clés Ki des rondes de DES, dont chacun contient 48 bits de K. Ces opérations utilisent des tables PC1 et PC2 du tableau 8, qu'on l'appelle choix permuté 1 et choix permuté 2. Pour commencer, 8 bits (K8, K16,…, K64) de K sont rejetés (par PC1). Les 56 bits restants sont permutés et affectés à deux variables de 28 bits C et D, et ensuite pendant 16 itérations, à la fois C et D sont mis en rotation soit 1 ou 2 bits, et 48 bits (Ki) sont choisis dans le résultat concaténé.

| *E* | | | | | |
|---|---|---|---|---|---|
| 32 | 1 | 2 | 3 | 4 | 5 |
| 4 | 5 | 6 | 7 | 8 | 9 |
| 8 | 9 | 10 | 11 | 12 | 13 |
| 12 | 13 | 14 | 15 | 16 | 17 |
| 16 | 17 | 18 | 19 | 20 | 21 |
| 20 | 21 | 22 | 23 | 24 | 25 |
| 24 | 25 | 26 | 27 | 28 | 29 |
| 28 | 29 | 30 | 31 | 32 | 1 |

| *P* | | | |
|---|---|---|---|
| 16 | 7 | 20 | 21 |
| 29 | 12 | 28 | 17 |
| 1 | 15 | 23 | 26 |
| 5 | 18 | 31 | 10 |
| 2 | 8 | 24 | 14 |
| 32 | 27 | 3 | 9 |
| 19 | 13 | 30 | 6 |
| 22 | 11 | 4 | 25 |

**Table 6: Fonctions par tour de DES: l'expansion E et la permutation P.**

Ci-après les S-boxes utilisés par l'algorithme DES

| S1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 |
| 1 | 0 | 15 | 7 | 4 | 14 | 2 | 13 | 1 | 10 | 6 | 12 | 11 | 9 | 5 | 3 | 8 |
| 2 | 4 | 1 | 14 | 8 | 13 | 6 | 2 | 11 | 15 | 12 | 9 | 7 | 3 | 10 | 5 | 0 |
| 3 | 15 | 12 | 8 | 2 | 4 | 9 | 1 | 7 | 5 | 11 | 3 | 14 | 10 | 0 | 6 | 13 |

| S2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 15 | 1 | 8 | 14 | 6 | 11 | 3 | 4 | 9 | 7 | 2 | 13 | 12 | 0 | 5 | 10 |
| 1 | 3 | 13 | 4 | 7 | 15 | 2 | 8 | 14 | 12 | 0 | 1 | 10 | 6 | 9 | 11 | 5 |
| 2 | 0 | 14 | 7 | 11 | 10 | 4 | 13 | 1 | 5 | 8 | 12 | 6 | 9 | 3 | 2 | 15 |
| 3 | 13 | 8 | 10 | 1 | 3 | 15 | 4 | 2 | 11 | 6 | 7 | 12 | 0 | 5 | 14 | 9 |

| S3 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 10 | 0 | 9 | 14 | 6 | 3 | 15 | 5 | 1 | 13 | 12 | 7 | 11 | 4 | 2 | 8 |
| 1 | 13 | 7 | 0 | 9 | 3 | 4 | 6 | 10 | 2 | 8 | 5 | 14 | 12 | 11 | 15 | 1 |
| 2 | 13 | 6 | 4 | 9 | 8 | 15 | 3 | 0 | 11 | 1 | 2 | 12 | 5 | 10 | 14 | 7 |
| 3 | 1 | 10 | 13 | 0 | 6 | 9 | 8 | 7 | 4 | 15 | 14 | 3 | 11 | 5 | 2 | 12 |

| S4 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2 | 12 | 4 | 1 | 7 | 10 | 11 | 6 | 8 | 5 | 3 | 15 | 13 | 0 | 14 | 9 |
| 1 | 14 | 11 | 2 | 12 | 4 | 7 | 13 | 1 | 5 | 0 | 15 | 10 | 3 | 9 | 8 | 6 |
| 2 | 4 | 2 | 1 | 11 | 10 | 13 | 7 | 8 | 15 | 9 | 12 | 5 | 6 | 3 | 0 | 14 |
| 3 | 11 | 8 | 12 | 7 | 1 | 14 | 2 | 13 | 6 | 15 | 0 | 9 | 10 | 4 | 5 | 3 |

| S5 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 7 | 13 | 14 | 3 | 0 | 6 | 9 | 10 | 1 | 2 | 8 | 5 | 11 | 12 | 4 | 15 |
| 1 | 13 | 8 | 11 | 5 | 6 | 15 | 0 | 3 | 4 | 7 | 2 | 12 | 1 | 10 | 14 | 9 |
| 2 | 10 | 6 | 9 | 0 | 12 | 11 | 7 | 13 | 15 | 1 | 3 | 14 | 5 | 2 | 8 | 4 |
| 3 | 3 | 15 | 0 | 6 | 10 | 1 | 13 | 8 | 9 | 4 | 5 | 11 | 12 | 7 | 2 | 14 |

| S6 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 12 | 1 | 10 | 15 | 9 | 2 | 6 | 8 | 0 | 13 | 3 | 4 | 14 | 7 | 5 | 11 |
| 1 | 10 | 15 | 4 | 2 | 7 | 12 | 9 | 5 | 6 | 1 | 13 | 14 | 0 | 11 | 3 | 8 |
| 2 | 9 | 14 | 15 | 5 | 2 | 8 | 12 | 3 | 7 | 0 | 4 | 10 | 1 | 13 | 11 | 6 |
| 3 | 4 | 3 | 2 | 12 | 9 | 5 | 15 | 10 | 11 | 14 | 1 | 7 | 6 | 0 | 8 | 13 |

| S7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 4 | 11 | 2 | 14 | 15 | 0 | 8 | 13 | 3 | 12 | 9 | 7 | 5 | 10 | 6 | 1 |
| 1 | 13 | 0 | 11 | 7 | 4 | 9 | 1 | 10 | 14 | 3 | 5 | 12 | 2 | 15 | 8 | 6 |
| 2 | 1 | 4 | 11 | 13 | 12 | 3 | 7 | 14 | 10 | 15 | 6 | 8 | 0 | 5 | 9 | 2 |
| 3 | 6 | 11 | 13 | 8 | 1 | 4 | 10 | 7 | 9 | 5 | 0 | 15 | 14 | 2 | 3 | 12 |

| S8 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 13 | 2 | 8 | 4 | 6 | 15 | 11 | 1 | 10 | 9 | 3 | 14 | 5 | 0 | 12 | 7 |
| 1 | 1 | 15 | 13 | 8 | 10 | 3 | 7 | 4 | 12 | 5 | 6 | 11 | 0 | 14 | 9 | 2 |
| 2 | 7 | 11 | 4 | 1 | 9 | 12 | 14 | 2 | 0 | 6 | 10 | 13 | 15 | 3 | 5 | 8 |
| 3 | 2 | 1 | 14 | 7 | 4 | 10 | 8 | 13 | 15 | 12 | 9 | 0 | 3 | 5 | 6 | 11 |

**Table 7: les S-boxes**

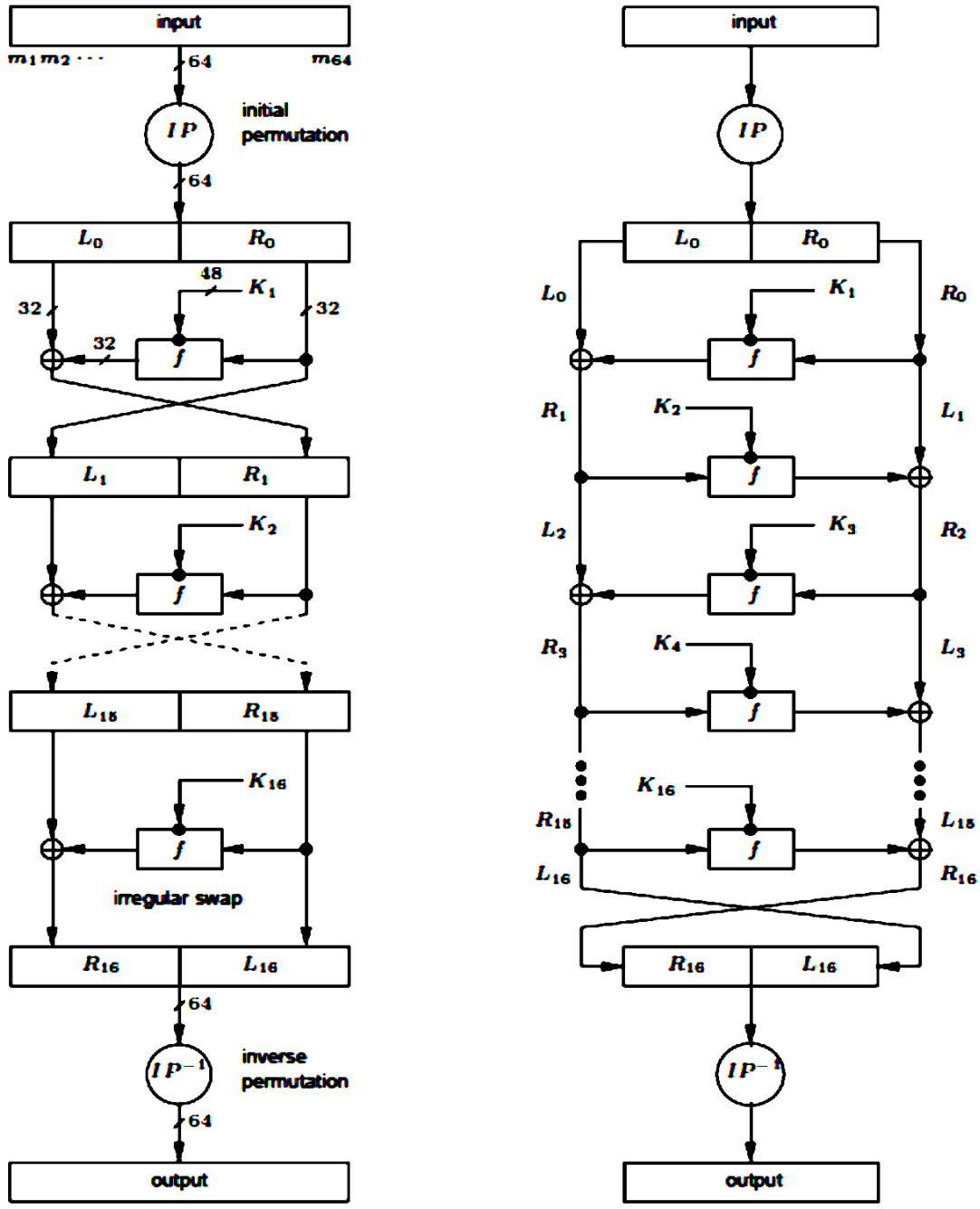



**Figure 4: Enchainement des calculs (chiffrement) dans le DES.**

Cette figure montre le processus de chiffrement de DES

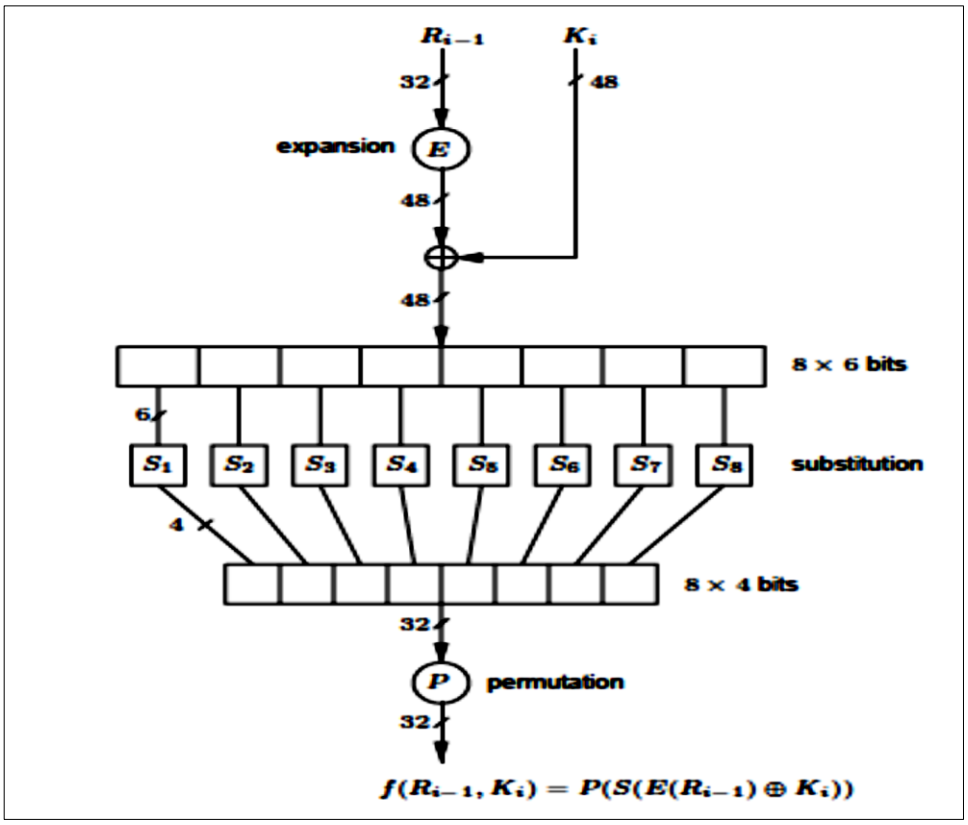



**Figure 5: Fonction interne f de DES.**

### c) Algorithme de production des clés de DES.

En entrée : Des clés initiales de 64 bits $k = k_1 \ldots k_{64}$ (avec 8 bits de parités sont inclues)

En sortie : seize clés de 48 bits $k_i$ ,$1 \leq i \leq 16$.

1. Définir $V_i$, ,$1 \leq i \leq 16$ de la façon suivante : $V$ i = 1 pour i ϵ {1, 2, 9,16} ; $V$ i =2 sinon.
2. T→ PC1 (K); représente T comme moitiés de 28 bits (C0, D0). (Utilisez le PC1 dans le tableau 8 pour sélectionner les bits de $K : C_0 = k_{57}k_{49} \ldots k_{36}$, $D_0 = k_{63}k_{55} \ldots k_4$).
3. Pour $i$ allant de 1 à 16, les *Ki sont* calculés de la manière suivante : Ci← ($C_{i-1}$ - $V$ i), $D_i$← ($D_{i-1}$ - $V_i$), *Ki* ← PC2 ($C_i$ ; $D_i$). (Utilisez le PC2 dans le tableau 8 pour sélectionner les 48 bits de la concaténation $b_1b_2 \ldots b_{56}$ de $C_i$ et $D_i$… $K_i = b_{14}b_{17} \ldots b_{32}$ ; '←' indique le décalage circulaire à gauche).

### d) Le déchiffrement de DES

Le déchiffrement DES utilise le même algorithme de chiffrement et les mêmes clés mais en inversant l'ordre d'application des clés, c'est-à-dire en commençant par $K_{16}$, $K_{15}$, …, $K_1$. Cela fonctionne de la manière suivante (voir la figure 4). L'effet de $IP^{-1}$ est annulée par la permutation IP dans le décryptage, partant de ($R_{16}$; $L_{16}$), appliquant le premier tour à

cette entrée. L'opération sur les rendements et demie, sera $R_{16} \oplus f(R_{15}; K_{16})$ au lieu de $\boldsymbol{L_0 \oplus f(R_0; K_1)}$, avec $\boldsymbol{L_{16} = R_{15}}$ **et** $\boldsymbol{R_{16} = L_{15} \oplus f(R_{15}; K_{16})}$, et qui est égale à $\boldsymbol{L_{15} \oplus f(R_{15}, K16) \oplus f(R_{15}, K16) = L_{15}}$. Donc le rendement du premier tour de décryptage est $(R_{15}, L_{16})$, c'est à dire, en inversant le tour 16. Notez que l'annulation de chaque tour est indépendante de la définition de ***f*** et de la valeur déterminée de $K_i$. La permutation des moitiés combinées avec le processus XOR est inversée par la seconde application. Les 15 tours restants sont également annulés un par un dans l'ordre inverse d'application, convenablement avec la production des clés inversées.

| PC1 | | | | | | |
|---|---|---|---|---|---|---|
| 57 | 49 | 41 | 33 | 25 | 17 | 9 |
| 1 | 58 | 50 | 42 | 34 | 26 | 18 |
| 10 | 2 | 59 | 51 | 43 | 35 | 27 |
| 19 | 11 | 3 | 60 | 52 | 44 | 36 |
| 63 | 55 | 47 | 39 | 31 | 23 | 15 |
| 7 | 62 | 54 | 46 | 38 | 30 | 22 |
| 14 | 6 | 61 | 53 | 45 | 37 | 29 |
| 21 | 13 | 5 | 28 | 20 | 12 | 4 |

| PC2 | | | | | |
|---|---|---|---|---|---|
| 14 | 17 | 11 | 24 | 1 | 5 |
| 3 | 28 | 15 | 6 | 21 | 10 |
| 23 | 19 | 12 | 4 | 26 | 8 |
| 16 | 7 | 27 | 20 | 13 | 2 |
| 41 | 52 | 31 | 37 | 47 | 55 |
| 30 | 40 | 51 | 45 | 33 | 48 |
| 44 | 49 | 39 | 56 | 34 | 53 |
| 46 | 42 | 50 | 36 | 29 | 32 |

**Table 8: Les permutations PC1 et PC2.**

### 1.1.2. Chiffrement IDEA

#### *a)* Le chiffrement IDEA

IDEA (International Data Encryption Algorithm) est un système de chiffrement par bloc de 64 bits, il repose sur une clé secrète de 128 bits, cette clé est utilisée pour générer 52 sous-clés de 16 bits. Le bloc de 64 bits est divisé en quatre blocs de 16 bits chacun qui subissent 8 rondes de chiffrement utilisant six sous-clés. La huitième ronde est suivie d'un changement final qui utilise quatre sous-clés. [40]

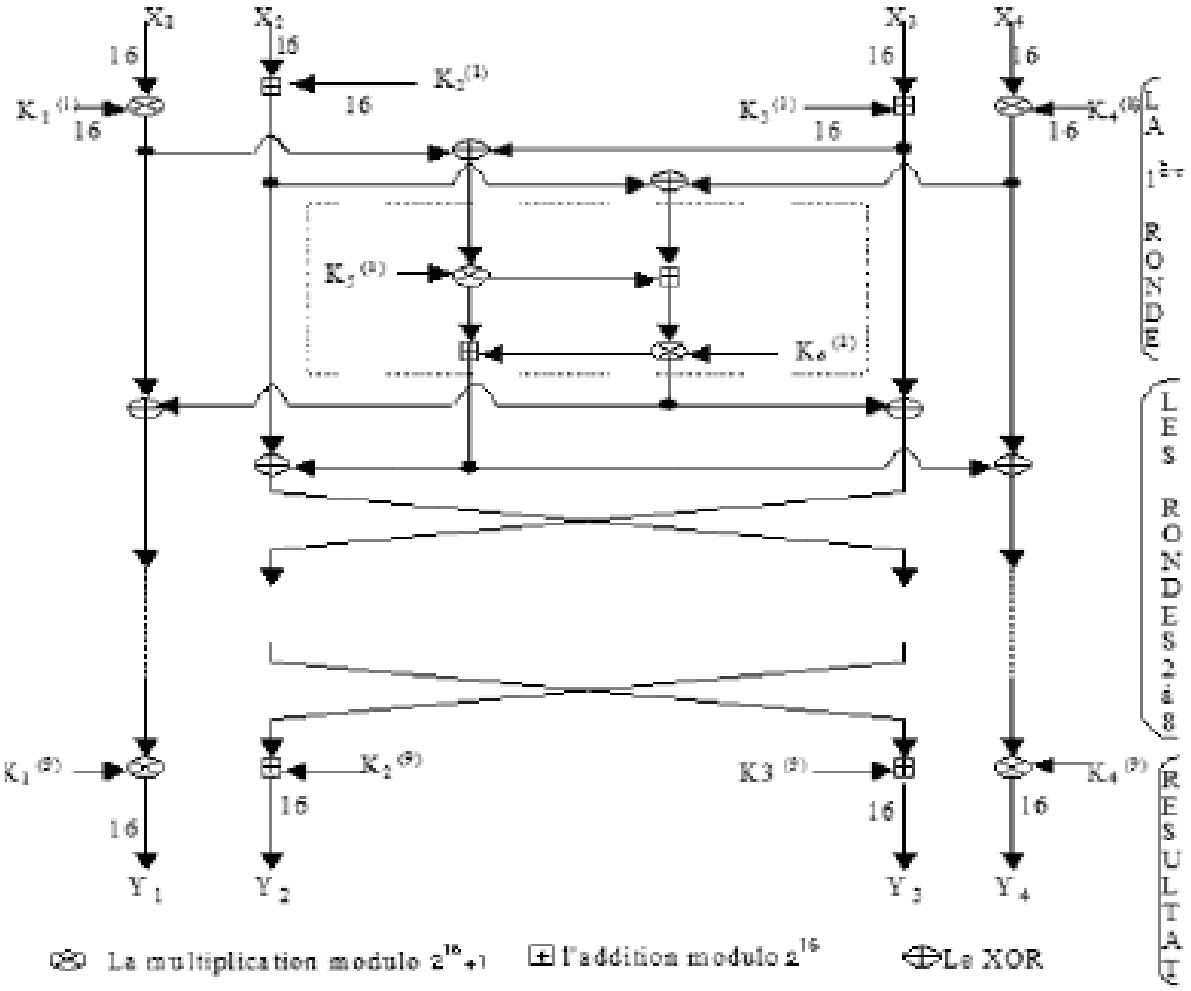



**Figure 6: Le chiffrement IDEA**

Lors de chaque ronde on applique une combinaison d'opérations de différents groupes algébriques, le OU exclusif (représenté par un $\oplus$), l'addition Modulo $2^{16}$ (représenté par un $\boxplus$) et la multiplication modulo $2^{16}+1$ (représenté par un $\otimes$). Toutes ces opérations manipulent des sous-blocs de 16 bits. Cet algorithme est ainsi efficace même sur des processeurs 16 bits.

*b)* **Génération des clés.**

Les 52 sous-clés sont générées à partir de la clé K de 128 bits de la façon suivante :

**1.** La clé de 128 bits est divisée en huit blocs de 16 bits. Ces huit blocs constituent les huit premières sous-clés utilisées dans le chiffrement.

**2.** La clé de 128 bits est ensuite cycliquement décalée de 25 positions vers la gauche et divisée en huit blocs de 16 bits. Ces huit blocs sont les huit sous-clés suivantes utilisées dans le chiffrement.

**3.** Le cycle est répété jusqu'à l'obtention des 52 sous-clés désirées.

*c)* **Déchiffrement**

Le déchiffrement s'effectue de façon analogue, les blocs de texte chiffré étant traités dans l'ordre inverse du chiffrement.

Les 52 sous-clés doivent également être recalculées comme inverses des sous-clés de chiffrement, par rapport à l'opération utilisée au cours de la ronde (soit l'addition soit la multiplication)

### 1.2. Chiffrement par flot

Est l'un des chiffrements les plus importants parmi les systèmes de chiffrement, l'idée principale consiste à chiffrer une suite de caractères à la fois, en utilisant une transformation qui varie au fur et à mesure selon le texte.

Les chiffrements par flot sont généralement plus rapides que les chiffrements par blocs coté hardware, et ne nécessitent pas de zone tampon.

Il y a un ensemble vaste de connaissances théoriques sur les chiffrements par flot, et de divers principes de conception pour ces chiffrements qui ont été proposés et largement analysés. Cependant, il y a relativement peu d'algorithmes de chiffrement par flot entièrement spécifiés dans la littérature ouverte. Cette situation regrettable peut s'expliquer en partie par le fait que la majorité des chiffrements par flot utilisés dans la pratique, ont tendance à être propriétaires et confidentielles. Par contre, de nombreuses propositions de chiffrement par bloc ont été publiés, dont certains ont été normalisés ou placés dans le domaine public. Pourtant, en raison de leurs avantages importants, les chiffrements par flot sont largement utilisés aujourd'hui, et on peut s'attendre à des propositions de plus en plus concrètes dans le futur.

***Exemple : One-time pad*** est un procédé de chiffrement, qui chiffre avec une manière continue par une addition binaire, dans lequel un flux de clés véritablement aléatoires est généré et ensuite combiné avec le message clair au cas de chiffrement et avec le message chiffré au cas de déchiffrement.

Soit $m_i$, $i$ = 1, 2, 3, … les caractères du texte clair. On utilisant un chiffrement de Verman : $c_i = m_i \oplus k_i$ , où $k_i$, $i$ = 1, 2, 3,…est le flot de chiffrement. Si les $k_i$ sont aléatoires et indépendants, on appelle ce chiffrement one-time pad, ce chiffrement est très simple et pourtant complètement inviolable si les préconditions suivantes sont remplies :

- ✓ La clé doit être aussi longue que le texte.
- ✓ La clé doit être aléatoire.
- ✓ La clé doit être utilisée une seul fois.

On distingue deux classes dans le chiffrement par flot :

- ✓ Le chiffrement par flot synchrone.
- ✓ Le chiffrement par flot auto-synchronisant.

### 1.2.1. Le chiffrement par flot synchrone

Un algorithme de chiffrement par flot synchrone est un algorithme qui combine le texte clair avec un flot de chiffrement qui est généré indépendamment du texte clair et du texte chiffré. On peut décrire le procédé de chiffrement par les équations :

$$\mu_{i+1} = f(\mu_i, k),$$
$$z_i = g(\mu_i, k),$$
$$c_i = h(z_i, m_i)$$

Où $\mu_0$ est l'état initial qui peut être déterminé à partir de la clé $k$, $f$ la fonction qui produit l'état suivant, $g$ la fonction produisant le flot de chiffrement $z_i$ et $h$ la fonction de sortie produisant le texte chiffré $c_i$ toute on combinant entre la séquence clé et le message clair $m_i$. Le procédé de déchiffrement et identique à celui de chiffrement, avec $m_i = h^{-1}(z_i, c_i)$ c'est-à-dire que la même clé est utilisée, de plus, il faut que l'émetteur et le récepteur soient dans le même état $\mu_i$ pour un bon déchiffrement. Ainsi, si lors de la transmission des caractères sont supprimés ou ajoutés, le déchiffrement sera échoué. Donc pour éviter ce problème, on peut réinitialiser, ou placer des marqueurs à des intervalles réguliers dans le texte chiffré.

***Exemple : Le chiffrement par flot additif*** est un chiffrement par flot synchrone avec une fonction h de sortie égale au « ou exclusif ($\oplus$) » :

$$\mu_{i+1} = f(\mu_i, k),$$
$$z_i = g(\mu_i, k),$$
$$c_i = z_i \oplus m_i$$

### 1.2.2. Le chiffrement par flot autosynchronisant

Nous avons déjà vu que pour les chiffrements à flot synchrones la perte de synchronisation entre les processus de chiffrement et de déchiffrement fausse tous les bits qui suivent et par conséquence, ils seront inutilisables.

Une solution proposée à ce problème est le chiffrement à flot autosynchronisant, pour lequel, le flot de chiffrement est produit à partir de la clé et d'un nombre fixe de caractères du message chiffré :

$$\mu_{i+1} = (c_{i-t}, c_{i-t+1}, \ldots, c_{i-1}),$$
$$z_i = g(\mu_i, k),$$
$$c_i = h(z_i, m_i),$$

Avec l'état initial est $\mu_0 = (c_{i-t}, c_{i-t+1}, \ldots, c_{i-1})$, $k$ est la clé, $g$ la fonction produisant le flot chiffrant $z_i$ et $h$ la fonction de sortie qui produit le message chiffré. L'avantage de ces algorithmes est qu'ils permettent au récepteur de synchroniser le générateur pseudo-aléatoire tous les m blocs, ce qui est désirable lorsque certains blocs sont détruits ou ajoutés au cours de la transmission. De plus ces algorithmes ont une particularité de corriger d'une façon automatique les erreurs liées à la perte de la synchronisation et cela est fait à l'aide d'un registre contenant les derniers bits du chiffré.

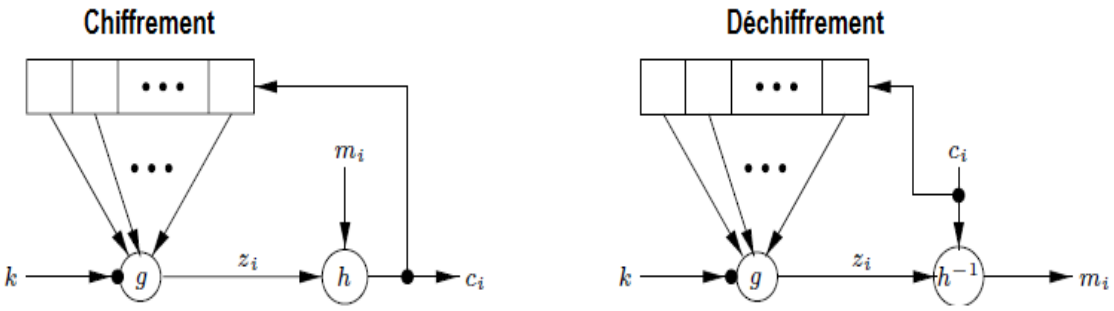



**Figure 7: Algorithme de chiffrement à flot autosynchronisant**

# CHAPITRE 2

# SÉCURITÉ DES SYSTÈMES SYMÉTRIQUES

## Sommaire

# 1. Introduction

En générale, la cryptographie symétrique est très utilisée dans le domaine de chiffrement, beaucoup plus que les algorithmes asymétriques. Elle se caractérise par une grande rapidité d'exécution que ça soit au niveau de son implémentation logicielle que du hardware qu'elle exige, ce qui accélère nettement les débits et autorise son utilisation massive. En contrepartie, la sécurité de ces systèmes repose essentiellement sur la sécurité de sa clé symétrique, sans pour autant négliger l'importance de la conception de l'algorithme de chiffrement lui-même. Dans ce chapitre nous étudions ce volet plus en détail et ce pour les deux familles des algorithmes de chiffrement symétriques à savoir les chiffrements par bloc et les chiffrements par flot.

# 2. La Cryptanalyse

## 2.1. Définitions

La ***confidentialité*** est le fait de s'assurer que l'information n'est accessible que pour les personnes autorisées.

Une ***attaque*** dans le contexte de la cryptographie est tout acte sur un système dont le but est de nuire aux propriétés C.A.I.N (confidentialité, intégrité, authentification, non répudiation), Dans les systèmes de chiffrement on appelle une attaque : ***Cryptanalyse.***

La ***cryptanalyse*** est donc, l'art de rendre clair un texte chiffré sans avoir connaissance de la clé utilisée. On considère que la force d'un système de cryptographie ne doit pas reposer sur la non connaissance d'un algorithme mais sur la force du principe utilisé. Dans la cryptographie, les méthodes de cryptanalyse sont très nombreuses, et dépendent en grande partie du type d'algorithme auquel on est confronté. [41] [42]

## 2.2. Classification d'attaque

La difficulté d'une cryptanalyse dépend essentiellement de la quantité d'informations à disposition de l'attaquant, plus les données manquent plus l'attaque est considérée difficile. On trouve alors les types d'attaques suivants :

1. **L'attaque à texte chiffré seulement** : L'attaquant a, à sa disposition un ou plusieurs textes chiffrés de la même manière.

2. **L'attaque à texte clair connu** : ce cas est cependant très rare, l'attaquant dispose de plusieurs textes chiffrés et les clairs correspondants.

3. **L'attaque à texte clair choisi** : L'attaquant possède la machine de chiffrement lui permettant d'avoir des textes chiffrés à partir des textes clairs bien choisis, cependant il lui manque la connaissance de la clé et parfois le mécanisme utilisé, même si ce dernier ne présente pas un problème car il peut facilement l'avoir. Ce cas se présente bien plus souvent qu'il ne le semble.

4. **L'attaque à texte chiffré choisi** : Dans ce cas de figure l'attaquant possède l'outil de déchiffrement lui permettant de partir d'un texte chiffré choisi pour avoir le texte clair correspondant sans connaitre la clé.

Néanmoins, si le système de chiffrement repose sur un algorithme parfait et sans faille on peut dire qu'il est sûr car la seule piste qui reste est *l'attaque à force "brute"* (ou attaque exhaustive). Ce type d'attaque dépend essentiellement de la taille de la clé ; plus la taille est grande, plus l'attaque devient impossible. En effet, étant donné que cette technique repose sur l'essaie de toutes les clés possibles pour le système de chiffrement en question, qui est un nombre exponentiel, l'attaque demande des centaines d'années (par exemple : 228 années pour une clé de DES de 56bits) pour tester toutes les clés possibles même avec un ordinateur très puissant. Cette limitation technique rend cette attaque désespérée sauf si la clé est petite. La recherche d'une faille dans le système reste la façon la plus efficace pour briser l'algorithme. [43]

### 2.3. Complexité des attaques

Dans les systèmes de chiffrement symétrique la mesure de la sécurité s'effectue principalement à travers deux moyens différents : une étude théorique des informations récupérées, en s'inspirant des règles de la théorie de l'information [44]. Sinon, l'autre moyen de mesure repose sur une étude de la complexité du système, face aux meilleures attaques réalisées sur les systèmes les plus connus. On en distingue naturellement plusieurs types :

- La complexité en données mesure la quantité de données nécessaire à la réalisation d'une attaque.
- La complexité en mémoire mesure la quantité de mémoire nécessaire au cours de l'attaque.

- La complexité en temps mesure le nombre d'unités de temps nécessaires pour mener à bien l'attaque. En général, l'unité de temps correspond à une opération de chiffrement ou à un cycle d'horloge.
- La complexité temps-mémoire-données : connue aussi par le nom temps-mémoire de Hellman, originellement conçu par Hellman et qui cherche à avoir un compromis entre ces diverses complexités.

## 3. Sécurité des systèmes symétriques à flot

Les algorithmes de chiffrement par flot se basent sur le procédé du chiffrement de Vernam en utilisant une suite chiffrante issue d'un GPA au lieu de la clé aléatoire. L'objectif derrière est d'empêcher l'adversaire à remettre en cause la confidentialité des messages si la sortie du GPA lui apparait comme aléatoire. En effet, l'attaquant doit être incapable de reconstituer tout ou partie de l'état interne du GPA, ni la clé utilisée pour initialiser le GPA, ni l'entré auxiliaire qui représente le vecteur d'initialisation public, IV.

Dans cette partie nous présentons les grands axes des cryptanalyses les plus connus dans ce domaine, sachant que la sécurité des systèmes symétriques à flot est évoluée dans le contexte d'une attaque à texte clair connu. [45]

### 3.1. Moyens de l'attaquant

Avant de commencer une cryptanalyse sur un système de chiffrement par flot, , on considère que les spécifications de l'algorithme sont bien connues chez l'attaquant. Toutefois il faut définir les moyens à mettre en place afin de permettre une analyse réussite permettant de retrouver la clé dans le cadre du chiffrement par flot. En générale, on se retrouve alors avec la classification suivante :

*Attaque à suite chiffrante connue* : ce moyen est utilisé dans le cas des GPA synchrones, il se base sur la construction de la suite chiffrante à partir du GPA en utilisant le mécanisme des attaques clairs connus, à clairs choisis, où même à chiffrés choisis.

*Attaque à IV connus* : on considère ce type d'attaque dans le cas des algorithmes utilisant l'entrée auxiliaire public IV permettant de produire plusieurs suites chiffrante à partir de la même clé secrète, ce mécanisme nécessite de mettre à la disposition de l'attaquant les différents IV utilisés dans le système.

*Attaques à IV choisis* : ce mécanisme repose sur le contrôle du comportement du GPA en fonction des IV misent en place.

Toutefois, une connaissance rigoureuse des systèmes visés est très demandée afin de pouvoir cibler le contexte le plus optimal pour mener à bien une attaque et aussi évaluer sa sécurité.

### 3.2. Objectifs de l'attaquant.

Dans la plupart des cas, une attaque correspond à la construction de la clé de chiffrement symétrique, ceci n'est pas toujours possible, cet objectif est aussi bien difficile qu'on ne peut pas l'atteindre rapidement et nécessite un travail complexe et très couteux pour l'accomplir. L'autre intuition conduit à reformuler un autre objectif moins couteux et plus aisé à remplir. Il s'agit de trouver tout ou partie de l'état interne en analysant les différentes sorties du GPA. Cet objectif est atteint à l'aide de l'attaque par distingueur des données aléatoires. Ce mécanisme permet de comparer les différentes données aléatoires générées par l'algorithme afin d'en tirer les sorties parfaitement aléatoires des sorties non aléatoires. De cette manière l'attaquant peut remplir l'objectif d'avoir une partie de l'état interne du GPA.

### 3.3. Attaques génériques

Les algorithmes de chiffrement par flot peuvent être attaqués d'une manière générique, pourquoi ? C'est à cause de la définition et la construction de l'algorithme lui-même. Donc on peut dire que c'est le dimensionnement de l'algorithme qui lui permet de se résister contre ce type d'attaques. Effectivement nous trouvons des attaques contre l'algorithme de chiffrement par flot que nous ne pouvons pas les pratiquer vu leur complexité qui dépend de trois variables ; la taille de clé, la taille de l'état interne, et la taille du IV.

#### 3.3.1. Attaques algébriques

C'est l'une parmi les attaques génériques en cryptographie symétrique proposée par Courtois et Meier en 2003 [46]. Lors d'une attaque à clair connu on peut établir un système d'équation reliant les chiffrés et les clairs, et les inconnus sont les bits de la clé. Si ce système est facile, on peut le résoudre par l'un des outils algébrique (linéarisation, Algèbre de Boole, Algèbre modulaire, ...). [47]

### 3.3.2. Attaque directe

Comme nous l'avons déjà vu***, la Recherche exhaustive sur la clé.*** C'est une attaque élémentaire contre tout algorithme de chiffrement utilisant des tailles de clés fixes. L'attaquant doit cependant disposer de la boite noire de l'algorithme sinon du test d'arrêt qui lui permet de détecter la clé utilisée, pour les GPA (Générateur Pseudo-Aléatoire) il est nécessaire que l'attaquant dispose d'une partie de suite chiffrante qui correspond à la taille de clé.

En contrepartie il faut augmenter le nombre des clés devant le nombre des calculs réalisables avec les moyens actuels et même les moyens envisagés au moyen terme. Par exemple si nous somme devant un calcul qui nécessite $\mathbf{2^{128}}$ opérations ce qui n'est pas réalisable au moyen terme alors il faut envisager au futur des clés de taille **128** pour prémunir contre une recherche exhaustive.

Pour éviter la recherche exhaustive sur un système de chiffrement à flot il faut augmenter la taille de la clé secrète pour avoir au moins 80 voire 128 bits ou plus. De plus, la taille de l'IV doit permettre d'éviter les collisions de telle sorte qu'on ne puisse pas tomber sur la même suite chiffrante IV utilisant le même IV. Ceci peut être vérifié par le paradoxe des anniversaires.

Pour bien comprendre ce principe nous supposons que dans une attaque à clair connu Oscar a récupéré $\boldsymbol{k}$ bits du message clair $\boldsymbol{m_1, m_2, \ldots, m_{k-1}}$ à un instant donné $\boldsymbol{t_0}$, ainsi que leur chiffrés $\boldsymbol{c_1, c_2, \ldots, c_{k-1}}$, donc il connait k bits de la suite chiffrante $\boldsymbol{z_1, z_2, \ldots, z_{k-1}}$ et par la suite il peut faire une recherche exhaustive sur l'état $Et_0$ tout on vérifiant la correspondance des bits de la suite chiffrante. Une fois Oscar trouve le bon état interne, il peut par la suite générer la suite chiffrante $(\boldsymbol{Z_i})$ avec $\boldsymbol{i > t_0+k}$, ce qui lui permet de déchiffrer tous les $(\boldsymbol{c_i})$ avec $\boldsymbol{i > t_0+k}$.

***Collision sur l'IV.*** Dans un chiffrement par flot une attaque par collusion sur les IV est le fait de trouver une collision entre deux IV pour la même clé. Lorsque ces IV sont générés de manière aléatoire, ils doivent être dimensionnés de telle sorte qu'une collision sur ces IV n'apparaisse qu'avec une probabilité négligeable voir Paradoxe des anniversaires *(Proposition 1).*

**Proposition 1**. Paradoxe des anniversaires : on considère un ensemble fini **X** de N éléments. La probabilité d'avoir une collision en tirant aléatoirement *k* éléments de façon

uniforme

est :

$$P(k) = 1 - \left(\frac{N-1}{N}\right)^{\frac{k(k-1)}{2}}$$
$$= 1 - \exp\left(\frac{k(k-1)}{2} \ln\left(1 - \frac{1}{N}\right)\right)$$
$$\geq 1 - \exp\left(-\frac{k(k-1)}{2N}\right)$$

Ainsi, on a $P(\sqrt{N} + 1) > 1 - e^{1/2} > 0.39$, et la probabilité augmente très rapidement avec *k* :

$P(1.2\sqrt{N} + 1) > 0.5$, $P(2\sqrt{N} + 1) > 0.86$, $P(4\sqrt{N} + 1) > 0.999$.

### 3.4. Compromis temps-mémoire-données

Le principe de cette attaque a été conçu par Hellman en 1980 [48]contre les chiffrements par bloc, après en 1995 contre les chiffrements par flot Babbage [49] et Golic [50]. L'objectif de ces attaques est d'inverser une fonction à sens unique. Si f une fonction de E vers F pour chercher x ϵ E / ***f(x) = y*** avec y ϵ F nous utilisons deux techniques :

- ✓ La recherche exhaustive
- ✓ L'attaque par dictionnaire qui a pour objet le calcul de toutes les images de ***x*** et de stoker les couples ***(x, f(x))*** dans une mémoire triés suivant la valeur de ***f(x)*** et par la suite à chaque fois que nous récupérons un ***y*** nous cherchons dans notre mémoire le ***x*** qui correspond à ***f(x) = y*** , c'est une phase de pré calcul qui nécessite le calcul et le stockage de tous les couples ***(x, f(x))***. Le problème majeur de cette technique est la taille de mémoire utilisée pour stoker les couples.

On peut appliquer l'attaque compromis temps-mémoire-donnée contre les chiffrements par flot de deux manières différentes :

1. La première lorsque l'attaquant veut retrouver l'état interne du GPA avec le paramétrage suivant : la fonction à sens unique est la fonction qui associe les n bits de l'état initial au ***n*** bits de la suite chiffrante générée par le GPA. Et on produit les ***k-n*** séquences de taille n de la suite à partir des k bits consécutifs connus de la suite chiffrante.

**Un résultat primordial de cette attaque** est que l'état interne d'un GPA doit toujours être au minimum deux fois plus grand que la taille de la clé secrète, sinon l'attaque par compromis temps-mémoire fournit une cryptanalyse plus efficace que la recherche exhaustive de la clé.

2. Dans un chiffrement par flot qui utilise des valeurs initiales de IV d'une taille n, pour la synchronisation du système, on prend comme fonction à sens unique la fonction qui associe à tout couple (clé secrète, IV) les (k + n) bits premiers de la suite chiffrante, avec k est la taille de la clé [51] [52].

### 3.5. Attaques par corrélation

Les attaques par corrélation sont des classes d'attaques appliquées sur les chiffrements par flot. Ces attaques ont été introduites en 1985 par SEIGENTHALER [53] . Le principe général de ce type d'attaque est de décomposer un système en des petites composantes, pour avoir des sous-systèmes crypto-graphiquement faibles. Cette attaque a été conçue contre les générateurs de LFSRs. Mais elle reste valide contre tout type de générateur pseudo-aléatoires dont l'état interne est décomposé en parties indépendantes.

***Description de l'algorithme*** : Supposons que nous avons un GPA et que nous pouvons décomposer l'état interne à l'instant t en deux, $\mathbf{x_t}$ et $\mathbf{y_t}$ de taille respectives $\boldsymbol{k}$ et $\boldsymbol{n-k}$ mises à jour indépendamment par les fonctions $\boldsymbol{\mu_0}$ et $\boldsymbol{\mu_1}$.

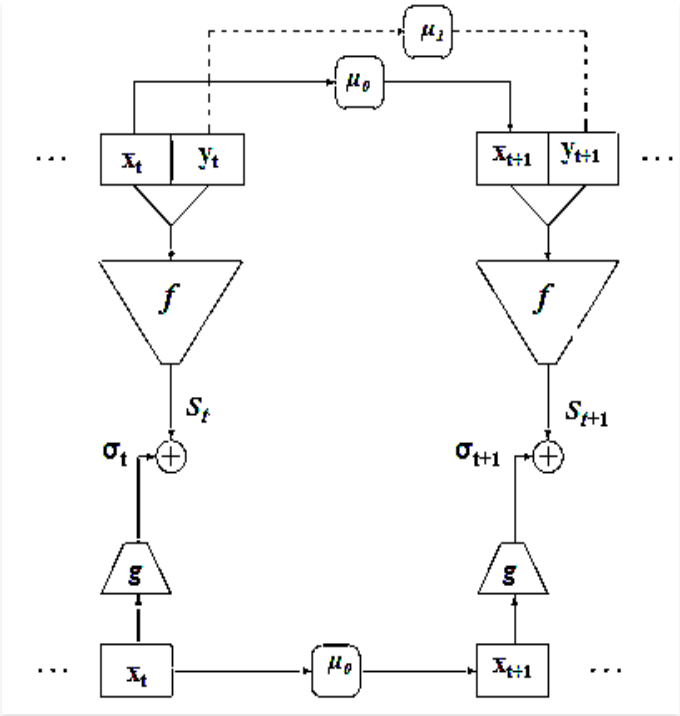



**Figure 8: Schéma de l'attaque par corrélation.**

Si l'attaquant cherche à retrouver la première partie $x_0$, donc on peut décomposer l'entrée de la fonction f en deux partie x, y et par la suite on peut appliquer l'attaque mais à condition qu'on trouve une fonction ***g*** de ***k*** variables qui coïncide avec la sortie de f avec la probabilité :

$$P_g = Pr_{x,y}[f(x, y) = g(x)] > \frac{1}{2}$$

La suite $\sigma(x0)$ produite par le générateur filtré de l'état initial $\mathbf{x_0}$ et de la fonction g est alors corrélé avec la suite chiffrante ***s*** de fait que pour tout $t \geq 0$ :

$$Pr[s_t = \sigma_t] = P_g > \frac{1}{2}$$

Dans ce cas l'attaque par corrélation est une simple recherche exhaustive sur la partie $\mathbf{x_0}$ en appliquant l'algorithme suivant :

*Entrée.*

Les N premiers bits de la sortie du GPA, $(s_t)_t < N$.

*Sortie.*

$x_0$, vecteur de $k$ bits correspondant à une partie de l'état initial.

*Algorithme*

Pour chaque vecteur $x_0$ de $k$ bits

- Calculer les N bits de la suite $\sigma(x0)$ définie par

$$\sigma_t = g \circ \mu_0^t(x_0).$$

- Calculer la corrélation sur N bits entre les suites s et $\sigma(x_0)$ :

$$c(\mathbf{s}, \sigma(x_0)) = \sum_{t=0}^{N-1} (-1)^{\sigma_t + s_t}.$$

Retourner la valeur de $x_0$ qui maximise $c(\mathbf{s}, \sigma(x_0))$.

**Figure 9: Algorithme Attaques par corrélation**

## 4. Sécurité des chiffrements symétriques par blocs

### 4.1. Cryptanalyse Différentielle

La notion d'attaque différentielle a été introduite en 1990 par Eli Biham et Adi Shmir. Ils ont détaillé et amélioré une technique de cryptanalyse du DES appelée

cryptanalyse différentielle. C'est une attaque à message clair choisi applicable aux algorithmes à clé secrète itératifs ou on fabrique un distingueur.

Le principe général de cette attaque consiste à considérer des couples de clairs M et M' présentant une différence ΔM fixée et à étudier la propagation de cette différence initiale à chaque ronde pendant le chiffrement.

**Différentielle et attaque par distingueur**.

Une différentielle est l'information donnée par les deux valeurs δ et δ* telle que :

$$M \oplus M' = \delta \text{ et } E_k(M) \oplus E_k(M') = \delta^*.$$

pour une clef k et une paire des textes clairs (M, M').

La probabilité de la différentielle est la probabilité p que la différentielle soit vérifiée sur l'ensemble des clés et sur l'ensemble des paires (M, M') telles que $M \oplus M' = \delta$.

Ceci est noté $\delta \rightarrow_p \delta^*$.

On note aussi $P_k$ la probabilité lorsque la clé est fixée et $P_{M,M'}$ lorsque la paire des textes clairs est fixée.

Le principe de l'attaque par distingueur est le suivant :

- L'attaquant chiffre des paires (M, M') de différence δ et observe le résultat (C, C'). Appelons p* la probabilité qu'une paire aléatoire vérifie $C \oplus C' = \delta^*$.
- L'attaquant sait que la probabilité que $C \oplus C' = \delta^*$ vaut $P_k$ si le système est bien $E_k$, et que cette probabilité vaut $p^* = 2-b$ pour une permutation aléatoire.
- Si $P_k \neq p^*$, alors le chiffrement d'un nombre N suffisamment élevé de paires aléatoires indépendantes et de différence δ permet de distinguer avec bonne probabilité $E_k$ d'une permutation aléatoire.

Supposons que $P_k > p^*$ (l'inégalité inverse se traite de façon similaire) et introduisons un seuil α tel que : si on observe au moins α paires telles que $C \oplus C' = \delta^*$, alors on confirme que la boite noire implante $E_k$, sinon on affirme que c'est une permutation aléatoire. Une valeur précise de la probabilité d'échec de ce distingueur peut être obtenue avec les bornes de CHERNOFF.

Si la probabilité $P_k$ est indépendante de k alors le distingueur nous permet obtenir une distinction avec une bonne probabilité le système de chiffrement d'une permutation

aléatoire [54] . Si la probabilité $P_k$ n'est pas indépendante de k, la même technique permet de détecter des clés faibles.

### 4.2. Cryptanalyse linéaire

Introduite par A.Corfdir et H.Gilbert pour la cryptanalyse de l'algorithme FEAL [55] et appliquée par M.Matsui au DES [56]. C'est une attaque à clairs connus sur les chiffrements par bloc itératifs. Matsui a d'abord appliqué la technique du chiffrement FEAL (Matsui et Yamagishi, 1992) [57]. Par la suite, il a publié une attaque sur le Data Encryption Standard (DES), pour aboutir finalement à la première cryptanalyse expérimentale de ce dernier dans la communauté open (Matsui, 1993; 1994). [58]

La cryptanalyse linéaire est une forme générale de la cryptanalyse basée sur la recherche d'approximations affines du chiffrement en le simplifiant. Et elle se base principalement sur les expressions linéaires probabilistes.

Il y'a deux techniques utilisées dans la cryptanalyse linéaire. La première consiste à construire des équations linéaires reliant (clair, chiffré) et les bits des sous-clés qui ont une forte probabilité (celles que l'on exhibe lors d'une attaque algébrique). La seconde est d'utiliser ces équations linéaires en conjonction avec les paires (clair, chiffré) connus pour dériver les bits de la clé [59].

Le principe de la cryptanalyse linéaire est de trouver l'expression linéaire efficace pour un chiffré donné :

$$P[i1, i2, \ldots, ia] \oplus C[j1, j2, \ldots, jb] = K[k1, k2, \ldots, kc] \quad (1)$$

Où i1, i2, …., ia , j1, j2, …., jb et k1, k2,…., kc désigne l'emplacement fixe des bits et l'équation (1) a une probabilité $\mathbf{p \neq ½}$ pour des paires (clair P, chiffré C) choisis de façon aléatoire. Donc la grandeur |p-1/2| représente l'efficacité de l'équation ci-dessus.

# CHAPITRE 3

# PROBLÈMES D'OPTIMISATION COMBINATOIRE

## Sommaire

## 1. Introduction

L'optimisation combinatoire est un domaine mathématique récent qui devient une spécialité très importante en recherche opérationnelle, en mathématique et en informatique théorique. Son importance et sa croissance viennent de la possibilité de formuler un nombre très important des problèmes concrets. Et d'autre part, la transformation des applications pratiques sous forme d'un problème d'optimisation combinatoire [60].On peut dire que la plupart de ces transformations sont faciles à définir mais très difficile à résoudre. On distingue entre deux familles des problèmes d'optimisation combinatoire, ceux qu'on connait des algorithmes pour les résoudre et d'autres qui ne possèdent pas de solution algorithmique valable jusqu'à ce jour connus sous le nom de problèmes *NP-difficiles*.

Dans ce chapitre nous allons définir quelques concepts généraux de l'optimisation combinatoire, aussi nous allons citer quelques exemples de ces problèmes et les solutions misent en place afin d'avoir une vision complète sur l'utilité de ce domaine dans le travail réalisé au cours de cette thèse.

## 2. Optimisation Combinatoire

Les problèmes d'optimisation se divisent en deux catégories différentes : ceux avec des variables continues, et ceux avec des variables discrètes appelées combinatoires. Dans la première catégorie nous cherchons généralement un ensemble des nombres réels ou un ensemble de fonctions réelles. Tandis que pour les problèmes combinatoires nous cherchons un objet qui appartient à un ensemble fini ou infini dénombrable sui peut être considéré généralement comme un entier, un ensemble de permutation, où même un graphe. Ce chapitre traite seulement les problèmes d'optimisation combinatoire.

***C'est quoi un problème d'optimisation combinatoire (POC) ?*** Un POC est défini par un ensemble *d'instances*. Chaque instance est associée à un ensemble discret de solution *S*, et en défini *X* un sous-ensemble de *S*, *ensemble des solutions réalisables* et une fonction *f* dite *fonction objective* qui attribue à chaque solution $s \epsilon X$ un nombre réel ou entier qui est *f(s)*. Trouvé une solution à cette instance de problème n'est rien que trouver une solution optimisée $s^* \epsilon X$ de la fonction *f*. s* est appelé solution optimale ou optimum global. D'où la définition suivante :

Noté bien qu'on puisse définir des problèmes de maximisation en remplaçant simplement ≤ par ≥.

***Définition :*** Un problème d'optimisation est un ensemble *I* d'instances d'un problème d'optimisation.

Après ces deux définitions il ne faut pas confondre un problème d'optimisation avec une instance de problème. Dans une instance nous avons des données en entrée ainsi que des informations pour obtenir une solution. Mais un problème d'optimisation est la collection de ces instances. Par exemple une instance de problème de voyageur de commerce est de donner une matrice de distance, mais lorsqu'on parle en général du problème de voyageur de commerce c'est la collection de toutes les instances associées à toutes les matrices de distance.

*Problème de décision* : est un problème où la réponse est soi positive ou négative c'est-à-dire oui ou non, prenons comme exemple le problème de l'existence d'un circuit Hamiltonien de coût inférieur à un paramètre *k* dans un graphe pondéré. La justification de la réponse négative ou positive est appelée *certificat*.

## 1. Classification des problèmes

On peut distinguer dans les problèmes de décision entre trois classes P, NP, et NP-complet. Intuitivement on peut dire que P est la classe des problèmes pour lesquels il existe des algorithmes polynômiaux pour trouver une solution ou pour prouver qu'elle n'existe pas est facile. La classe NP est la classe des problèmes pour lesquels on ne connait pas un algorithme polynômial.la troisièmes classes est la classe des problèmes les plus difficiles dans la classe NP.

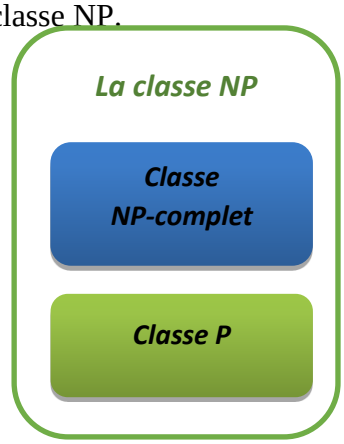



***Figure* 11: Conjecture couramment admise concernant les relations entre P, NP et NPC**

Il est difficile de résoudre un problème, mais il est facile de prouver qu'il appartient à une parmi ces classes. Par exemple si on veut prouver qu'un problème $P_1$ appartient à la classe *P*, il suffit d'exhiber un algorithme polynômial qui le résoudre.

Pour prouver qu'il appartient à NPC, il faut en premier temps établir qu'il appartient à NP. Généralement c'est immédiat. Ensuite deux démarches sont possibles : ou bien une preuve directe, c'est ce qui a été fait par exemple pour le problème SAT (problème de satisfiabilité d'une expression booléenne), ou bien une preuve indirecte qui utilise le fait que l'appartenance de certains problèmes à NPC a déjà été établie. On peut par exemple montrer que $P_1$ admet comme sous-problème un problème $P_2$ qui appartient à NPC. On peut aussi mettre en évidence une transformation polynomiale d'un problème $P_2$ qui appartient à NPC vers $P_1$. On utilise alors le fait que la combinaison de deux transformations polynomiales est aussi une transformation polynomiale. À noter l'existence de problèmes de NP dont l'appartenance, ni à P, ni à NPC, n'a pu être établie. [61]

## 2. Exemple des problèmes classiques d'optimisation combinatoire

### 2.1. Problème du sac-à-dos

Le problème de sac-à-dos est l'un des problèmes NP-complet qui consiste à maximiser un critère de qualité sous une contrainte linéaire de capacité de ressource. Le problème qui de propose est de remplir le sac-à-dos avec des objets choisis de façon à avoir un sac le plus « utile » possible, tout en respectant son volume.

Plus formellement, on peut le décrire de la façon suivante. Soit un ensemble de n éléments et une ressource disponible en quantité limitée, b. Pour j = 1 à n, on note $p_j$ le profit associé à la sélection de l'élément j et on note $a_j$ la quantité de ressource que nécessite l'élément j, s'il est sélectionné. Les coefficients pj et $a_j$ prennent des valeurs positives pour tout j = 1 à n. Le problème du sac-à-dos consiste à choisir un sous-ensemble des n éléments qui maximise le profit total obtenu, en respectant la quantité de ressource disponible [62].

Pour obtenir le profit total on associer à chaque j une variable $x_j$ binaire qui vaut 1 si l'élément j est choisi 0 sinon, et donc on peut écrire la formule du profit

comme suit :$\sum_{j=1}^{n} pj.xj$ et la quantité totale de ressource utilisée sera la somme suivante :$\sum_{j=1}^{n} aj.xj$. Le problème de sac-à-dos se modélise comme suit :

$$\text{Max} \sum_{j=1}^{n} pj.xj$$

$$\text{Sous contrainte} \quad \sum_{j=1}^{n} aj.xj \leq b$$

$$x_j \in \{0,1\} \ \forall j = 1..n$$

La contrainte de ressource est appelée « contrainte de sac-à-dos ». Dans le cas où l'on a plusieurs contraintes de ressource par exemple un volume maximal et un poids maximal on parle de problème de sac-à-dos « multidimensionnel ».

Plusieurs travaux ont été développés pour proposer des solutions au problème du sac-à-dos. Parmi ces méthodes, les méthodes exactes, généralement basées sur une approche par séparation et évaluation sont limitées à des instances de petites tailles, ce qui justifie le recours aux méthodes heuristiques.

### 2.2.Problème d'affectation

Le problème d'affectation consiste à distribuer à chaque agent une tâche. Mathématiquement parlant c'est établir un lien entre deux ensembles distincts. Tout en minimisant le coût et en respectant le fait que ces liens doivent être uniques.

Si on a un ensemble $m$ des tâches et un ensemble $n$ des agents, avec $m \leq n$. $\forall$ $(i,j)$ avec $i = 1..m$ et $j = 1..n$. l'affectation de $i$ à $j$ entraine un cout de réalisation noté $c_{i,j}$ ($c_{i,j} \geq 0$).chaque tâche doit être réalisé seulement une fois et chaque agent peut réaliser au plus une tâche. Donc notre problème consiste à affecter les tâches aux agents. Avec un coût total de réalisation minimal et un respect de certaines contraintes (contraintes de réalisation des tâches et la disponibilité des agents). [62].

A chaque couples $(i,j)$, ont associé une variable d'affectation $x_{i,j}$ qui vaut 1 si la tâche $i$ est affecté à l'agent $j$ et 0 sinon. On peut modéliser le problème d'affectation sous la forme :

$$\text{Min} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{i,j}.x_{i,j}$$

$$\text{s.c} \quad \sum_{j=1}^{n} x_{i,j} = 1 \qquad \forall j = 1..n$$

$$\sum_{i=1}^{m} x_{i,j} \leq 1 \qquad \forall i = 1..m$$

$$x_{i,j} \in \{0,1\} \qquad \forall i = 1..m, \forall j = 1..n$$

Avec :

- $\sum_{j=1}^{n} xi,j$ le nombre d'agents réalisant la tâche i pour tout i=1..m.
- $\sum_{i=1}^{m} xi,j$ le nombre des tâches i réalisées par l'agent j pour tout j =1..n.
- $\sum_{i=1}^{m} \sum_{j=1}^{n} c_{i,j}.x_{i,j}$ est le coût total de réalisation des tâches.

Ces contraintes est appelé en générale « les contraintes d'affectation ».

### 2.3.Problème de voyageurs de commerce

Le problème de voyageur de commerce PVC (Traveling Salesman Problem- TSP), est un problème mathématique qui consiste à visiter un nombre de ville séparées entre eux par des distances données, le PVC est l'un parmi les problèmes les plus important des problèmes combinatoires car il est facile à décrire et difficile à résoudre.

Plus formellement, un exemple de PVC est donné par un graphe complet G sur un ensemble de nœuds V = {1,2, ... m}, pour un entier m, et par une fonction de coût assignant une $c_{i,j}$ coût de l'arc (i, j) , pour tout i, j dans V. autrement dit soit un Graphe complet $G=(V, A, c)$ avec $V$ un ensemble de sommets, $A$ un ensemble d'arêtes et c, une fonction de coût sur les arcs. Le problème est de trouver le plus court dans le graphe.

***Définition :*** On appelle chemin hamiltonien (chaine hamiltonienne) un chemin (une chaine) passant une fois, et une fois seulement, par chacun des sommets de G.

Un chemin hamiltonien (une chaine hamiltonienne) est donc un chemin (une chaîne) élémentaire de longueur n-1.

Un circuit hamiltonien (un cycle hamiltonien) est un circuit (un cycle) qui passe une fois, et une seule fois, par chacun des sommets de G.

On dit qu'un graphe G est hamiltonien s'il contient un cycle hamiltonien (cas non orienté) ou un circuit hamiltonien (cas orienté). [63]

Prenant comme exemple le graphe indiqué sur la figure ci-après le chemin idéal c'est celui avec un coût total minimal.

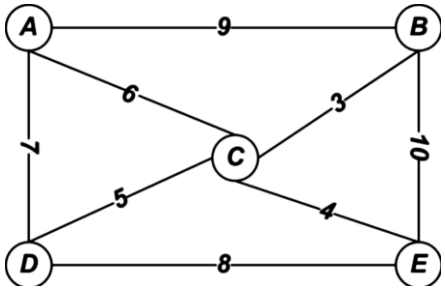



**Figure 12 : La tournée décri par les sommets (A, B, C, E, D, A), de coût 31 est un cycle optimal.**

En général on peut deviser le PVC en deux types différents, le PVC symétrique et asymétrique. Pour la forme symétrique on l'appelée PVCS (STSP en anglais) il n'y a qu'un seul chemin entre deux villes voisines, à savoir, la distance entre les villes A et B est égale à la distance entre les villes B et A. Par contre dans la forme asymétrique appelé le PVCA (ATSP en anglais) il n'y a pas une telle symétrie, il est possible d'avoir deux coûts ou deux distances différentes entre les deux villes A et B. Par conséquent, le nombre de visites dans le PVCS et le PVCA sur n sommets (villes) est respectivement (n-1)! , et (n-1)!/ 2.

### 2.4.Problème de partition

Dans ce paragraphe, nous considérons le dernier exemple de nos problèmes NP-complets de base, le problème de PARTITION. Il est particulièrement utile pour prouver les résultats NP-complétude pour des problèmes impliquant des paramètres numériques, telles que la longueur, le poids, les coûts, les capacités, etc.

***Théorème***. Le problème de partition est NP-complet.

***Preuve :*** Il est facile de voir que le problème de partition ϵ NP, à partir d'un algorithme non déterministe il suffit de trouver un sous-ensemble A' de A et de vérifier en temps polynomial que la somme des tailles des éléments de A' est la même que les éléments de A-A'.

***Démonstration*** : On va réduire le problème de la *SOMME DE SOUS-ENSEMBLE* à *PARTITION*. Soit (E, t) une instance de SOMME DE SOUS-ENSEMBLE.

On pose $S=\sum_{x\in E} x$.

Quitte à changer t en S−t (changer l'ensemble obtenu en son complémentaire), on peut supposer que $2t \leq S$.

L'idée naturelle consisterait à ajouter l'élément u = S−2t à E ; le résultat de partition serait alors deux sous-ensembles (A' et son complémentaire) de somme S−t ; l'un des deux contient l'élément S−2t, donc en enlevant ce dernier, on trouve un ensemble de termes de E de somme t. Malheureusement, cette technique échoue si S−2t est déjà dans E.

Au lieu de cela, on prend le nombre X = 2S et $X_0$ = S + 2t, et on applique PARTITION à E' = E ∪{ X, X'}. Il existe une partition de E' si et seulement s'il existe un sous-ensemble de E de somme t. En effet, s'il existe une partition de E', il existe deux sous-ensembles complémentaires de somme 2S + t. Chacun des deux sous-ensembles doit contenir soit X, soit X', car sinon sa somme ne peut excéder 2S + t ; donc un des deux ensembles contient X et non X', et on obtient en enlevant X un sous-ensemble F de E de taille t. Réciproquement, étant donné un tel F, (F∪{X}; E−F∪{X'}) constitue une partition de E'. On a donc bien réduit SOMME DE SOUS-ENSEMBLE à PARTITION.

Reste à justifier que la réduction est bien polynomiale. L'essentiel de la réduction est le calcul de A et A', qui est bien polynomial en la taille des entrées (l'addition de k nombres de n bits se fait en temps $\mathcal{O}$(k log n)).[GJ79]

## 3. Résolution pratique des problèmes combinatoires

Les problèmes d'optimisation combinatoires étant généralement difficiles et de grandes tailles, un grand nombre de méthodes directes et indirectes ont été proposés pour leur résolution. Mais nous avons toujours une difficulté, c'est de trouver un algorithme à la fois correct, rapide et complet. Et par conséquent, l'objectif est d'avoir un comportement acceptable en pratique, autrement dit, avoir la capacité de trouver une solution de qualité dans un temps raisonnable pour certaines instances de problème.

### 1. Approche complète

L'idée générale de cette approche c'est de chercher à explorer l'ensemble des configurations de façon exhaustive et systématique, on cherchant à réduire la combinatoire en utilisant un ensemble de techniques visant à éliminer le plus tôt possible des sous-ensembles de configurations en prouvant qu'ils ne contiennent pas de solution. Ces approches sont complètes dans le sens où elles sont toujours capables de trouver la solution optimale ou, le cas échéant, de prouver l'absence de solution.

***Structuration en arbre***. Cette méthode consiste à découper l'ensemble de configuration en sous-ensembles. En construisant un arbre de recherche, avec la racine qui est construite par toutes les configurations et les nœuds sont les sous-ensembles de configurations, et on ajoute une précision que les sous-ensembles deviennent de plus en plus petits à chaque fois qu'on descend dans l'arbre. Généralement on se base sur « la séparation et l'évaluation » (Branch and bound) pour construire cet arbre, en partant de la racine et on répète les étapes suivantes :

- ✓ Selon une heuristique de sélection donnée, on choisit la dernière feuille créée pour une exploration, ou choisit la feuille ayant une meilleure exploration ;
- ✓ Partitionner l'ensemble des configurations associé à une feuille sélectionnée en plusieurs sous-ensembles
- ✓ Évaluer le sous-ensemble de configurations associé à un nœud, afin de déterminer s'il peut contenir ou pas une solution.

L'utilité de la fonction d'évaluation est de justifier la fiabilité de l'approche en pratique : cette fonction d'évaluation doit pouvoir déterminer efficacement qu'un ensemble de configurations ne peut pas contenir de solutions.

Dans le cas des problèmes combinatoires, la fonction d'évaluation sert à détecter la bonne solution tout en comparant les estimations des coûts qu'elle retourne, des différentes solutions. Si après avoir une bonne solution, une estimation est moins bonne que la meilleure celle de la solution trouvée jusqu'ici, alors on peut couper le nœud correspondant.

Pas bien compris le paragraphe ci-dessus.

Pour les problèmes de décision, la fonction d'évaluation doit être capable de détecter le fait qu'un ensemble ne contient pas de configuration solution. Cette

évaluation est souvent réalisée en « filtrant » l'ensemble des configurations, i.e., en éliminant des parties dont on infère qu'elles ne peuvent pas contenir de solution. Ainsi, une stratégie classique pour résoudre des problèmes de satisfaction de contraintes consiste à exploiter (« propager ») les contraintes du problème pour éliminer du domaine des variables les valeurs violant les contraintes, filtrant ainsi l'ensemble des configurations candidates. Le résultat de cette étape de filtrage vérifie généralement une certaine consistance partielle, e.g. la consistance de nœud, d'arc ou de chemin [Tsa93].

## 2. Approche incomplète

Il est vrai que via les approches complètes nous sommes arrivés à résoudre en pratique pas mal des problèmes combinatoires. Mais nous ne cachons pas qu'elle reste incapable de résoudre certains d'autres.

Une alternative consiste à feuilleter l'espace de solution d'une façon opportuniste, tout on utilisant des heuristiques pour but d'extraire les zone de l'espace des solutions qui semble être prometteur, et d'ici que vient la nomination par l'approche incomplète, car elles n'explorent qu'une partie de l'espace des solutions.

Essentiellement on distingue entre deux grandes familles de l'approche heuristique :

***Les approches perturbatrices***, qui consiste à construire des nouvelles combinaisons tout on modifie d'autres combinaisons existantes.

***Les approches constructives***, qui consiste à générer des nouvelles combinaisons d'une maniéré incrémentale en utilisant pour cela un modèle stochastique.

Par la suite de cette paragraphe sur les approche incomplète on va considérer le couple ***(E, f )*** qui définit le problème à résoudre avec ***E*** l'ensemble des combinaisons et ***f : E→IR*** est une fonction objectif qui associé à chaque combinaison de E une valeur réelle. Résoudre le problème est équivalent à trouver ***e*∈ E*** qui optimise la fonction *f*.

### 2.1. Les approches perturbatrices

#### 2.1.1 Les algorithmes génétiques

Les algorithmes génétiques (AG) sont les plus connues des approches perturbatrices, les AG ont été initiées par John Holland dans les années 1970 [64]. Les

AG sont des techniques qui s'inspirent de la génétique et de la théorie de l'évolution de la nature en se basant sur les mécanismes suivants :

***La sélection naturelle*** permet aux individus les plus forts et les plus adaptés de survivre pendant une longue durée.

***Le croisement*** est l'opérateur génétique qui produit des individus qui héritent les propriétés des deux parents croisés, c.-à-d. l'étape de reproduction des individus (population).

***La mutation*** est l'opérateur qui consiste à modifier d'une façon aléatoire la valeur d'une ou plusieurs variables d'un individu. La mutation est considérée dans les AG comme un opérateur secondaire après le croisement.

Afin de définir une méta-heuristique, les AG répètent ces opérations génétiques. L'idée est de refaire les opérations jusqu'à trouver la meilleure combinaison c.-à-d. la plus adapté, et cela à travers l'évolution, et l'évaluation de cette combinaison par la fonction objective.

***Schéma général des AG***

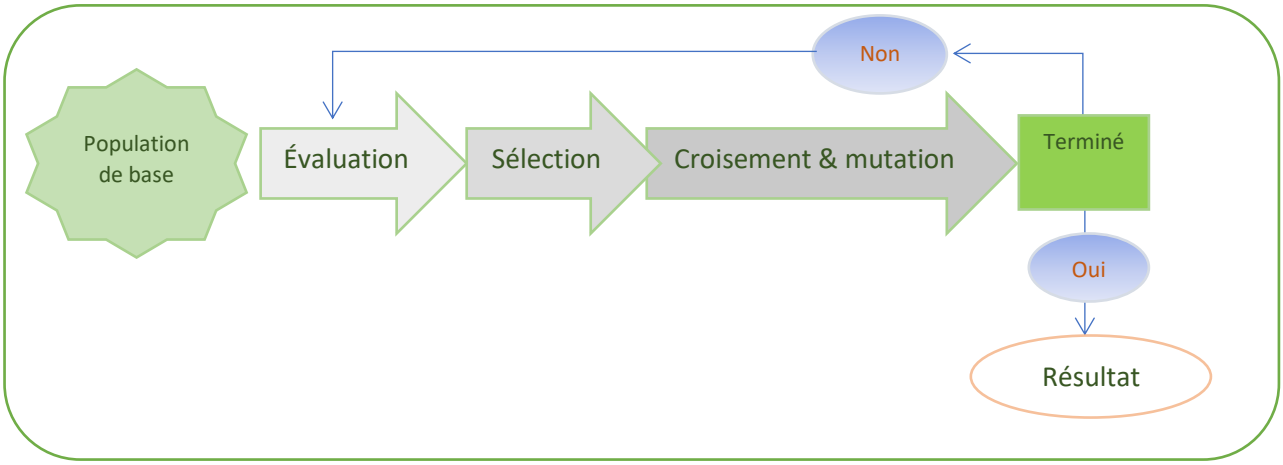



**Figure 10: Schéma d'un algorithme génétique**

### 2.1.2 La recherche locale

**La recherche locale**, cette technique intensifie la recherche en exploitant l'espace des combinaisons de proche en proche, en partant d'une combinaison initiale et en sélectionnant à chaque itération une combinaison voisine de la combinaison courante, obtenue en lui appliquant une transformation élémentaire [33]. L'algorithme 2 décrit ce principe général, dont les principales étapes sont détaillées ci-après.

***Algorithme* : Recherche locale**

Génération une combinaison initiale e $\epsilon E$
**Tant que** les critères d'arrêt non atteints **faire**
    Choisir **e'$\epsilon$ V(e)**
    **e ← e'**
**Retourner** la meilleure combinaison construite

Différentes variantes se présentes et qui dépendent de la politique de sélection de la prochaine configuration parmi l'ensemble des voisinages. Par exemple on peut choisir à chaque itération le meilleur voisin, c'est-à-dire celui qui permet de donner une bonne amélioration de la fonction objective. Un autre exemple, on choisit le premier voisin repéré qui améliore la fonction objective. Cette approche peut tomber sur des voisins qui peuvent être moins bons, et pour éviter ces optima locaux, on fait appeler des méta-heuristiques comme :

- ***le recuit simulé*** *(Simulated Annealing*) [65], cette méthode est une généralisation de la méthode Mont-Carlo, l'objectif de cette méthode est de trouvé une solution optimale toute on élimine les optima locaux les moins bons. Cette méthode a été conçue par trois chercheurs de la société IBM S. Kirkpatrick, C.D. Gelatt et M.P. Vecchi en 1983. Le recuit simulé s'inspire du processus de recuit physique. Ce processus utilisé en métallurgie pour améliorer la qualité d'un solide cherche un état d'énergie minimale qui correspond à une structure stable du solide.

- ***La recherche de tabou***, cette méthode proposée par Fred Glover en 1986 [66] et indépendamment par Hansen [67]. Cette méthode utilise un ensemble des règles pour guider la recherche d'une manière intelligente dans l'espace des solutions. Contrairement au recuit simulé qui génère aléatoirement une seule solution voisine **e'$\epsilon$ V(e**) à chaque itération, tabou examine un échantillonnage de solution de **V(e)** et retire la meilleure solution **e'**. Le problème est qu'on peut tomber sur une solution ***e'*** même si la solution initiale e est meilleure que ***e'***, ce qui peut générer des cycles comme ***e→ e'→ e→ e'…*** Pour éviter ce cycle, on mémorise les ***k*** dernières configurations visitées dans une mémoire

temporaire qu'on l'appelle *liste taboue,* cette liste permet d'éviter tous les cycles de longueur inférieur ou égale à *k.*

- ***La recherche à voisinage variable*** [68], le principe de cette méthode et de changer systématique de voisinage au sein d'une recherche locale. En plus on introduit une phase de perturbation pour s'éloigner des mauvaises solutions.

### 2.2.Les approches constructives

Le principe des approches constructives est de construire une ou plusieurs configurations d'une manière incrémentale, i.e., en commence par une configuration « vide », et on ajoute des composants a fur et à mesure à la configuration jusqu'à obtenir une configuration finale, le choix de ces composants se fait en une fonction d'une heuristique qui estime localement la qualité du composant. Il existe différentes stratégies pour le choix des composants, on présente par la suite deux stratégies, *les stratégies gloutonnes* (greedy), *les algorithmes à base de fourmis*.

***Les algorithmes gloutons***, le principe de ces algorithmes est de partir d'une solution vide et d'ajouter des composants d'une manière incrémentale jusqu'à sans remettre en cause le choix de ces composants, ces algorithmes permettent de construire des configurations très rapidement, par contre la qualité de la configuration dépend de l'heuristique.

***Les algorithmes à base de fourmis***, dont le principe général est de construire des configurations de façon incrémentale, mais le choix du prochain composant dépend d'une règle de transition probabiliste. Ce principe a donné naissance à la méta-heuristique d'optimisation par colonies de fourmis.

# CHAPITRE 4

# LE SYSTÈME DE CHIFFREMENT ÉVOLUTIONNISTE SEC

## Sommaire

## 1. Introduction

Symetrical Evolutionary Ciphering SEC, est un nouveau système de chiffrement symétrique reposant sur les algorithmes évolutionnistes. Ce système permet d'assurer un meilleur chiffrement en cherchant à maximiser une fonction d'évaluation qui concrétise chaque transformation candidate d'être le chiffré du texte clair et permet ainsi de sélectionner la meilleure solution possible à la fin de l'algorithme.

L'emploi de ce système est effectué sur la base d'un encodage bien défini du texte clair, afin de garder toutes les propriétés nécessaires pour la réalisation de la meilleure transformation.

L'idée derrière l'application des algorithmes évolutionnistes sur un problème de chiffrement, n'était pas un fait de hasard mais c'était le fruit d'une étude approfondie sur les différentes approches possibles et dont les principaux points de motivation étaient :

1- Profiter des processus aléatoires qu'offre cet algorithme et qui représente un atout très important dans le domaine de la cryptographie.

2- Meilleur algorithme pour la résolution des problèmes d'optimisation combinatoire auquel nous avons pu ramener le problème de chiffrement

3- L'utilisation d'une fonction objective permettant d'évaluer les différentes transformations possibles pour un texte donné

La première version de ce système présente un vrai succès de cette approche à travers laquelle on a pu dépasser les obstacles les plus durs pour la construction d'un tel algorithme et qui sont :

- Le passage de l'espace génotype à l'espace phénotype sans risque de perte d'information.

- La définition d'une fonction mathématique permettant d'évaluer chaque individu de la population générée

- La définition des méthodes à mettre en place au niveau de chaque étape de l'algorithme afin de pouvoir créer les différentes solutions possibles pour notre problème.

Plusieurs autres versions dérivées de cette approche ont été réalisées afin d'améliorer le résultat final et ceci en se focalisant sur l'encodage du texte en entrée comme ce qu'on va voir dans la deuxième partie de ce chapitre, et aussi sur la combinaison de ce système avec de nouveaux processus tel qu'il est détaillé dans le chapitre 6. Et enfin l'hybridation de ce système avec un système de chiffrement asymétrique.

Cependant; notre conviction sur la robustesse de l'algorithme en soit, nous a ramené à pousser nos études sur le système d'origine encore plus, et chercher à augmenter sa résistance contre les différentes attaques possibles dans notre cas. Une description détaillée de ce travail sera discutée en détail dans le dernier chapitre.

## 2. Description du système SEC

### 1. Formalisation du problème

Soit M le message clair à chiffrer.

***Formalisation du problème***

M est constitué d'une séquence de caractères, des chiffres ou même des symboles qui seront désignés dans notre système par $C_1, C_2, \ldots, C_n$.

La représentation de M sera donc définie en fonction de ces différents éléments ainsi que les listes de leurs positions dans le message d'origine, soient $L_1, L_2, \ldots, L_n$.

La reformulation mathématique de cette représentation peut être traduite par le vecteur suivant :

$$M = \begin{Bmatrix} (C_1, L_1) \\ (C_2, L_2) \\ . \\ . \\ . \\ (C_n , L_n) \end{Bmatrix}$$

Je note ainsi que les listes $L_1, L_2, \ldots, L_n$ représente une partition de l'ensemble $\{1, 2, \ldots, n\}$ avec n est la taille du message M.

Ce qui veut dire : $Li \cap Lj = \emptyset$ si $i \neq j$, $\forall i,j \in \{1, 2, \ldots, n\}$.

Passant maintenant au codage de l'individu (Chromosome) qui permet de présenter les différentes solutions utilisées dans le système.

En fait ; une fois nous définissons le vecteur initial de notre texte clair qui représente notre chromosome d'origine « Original-CH », le codage de notre problème devient très facile puisque chaque gène sera équivalent à une liste des positions d'un caractère de ce texte. Autrement dit, un individu est équivalent à un vecteur ($C_i$, $L_{pi}$) dont $L_{pi}$ est le $i^{ème}$ gène contenant les nouvelles positions du caractère $C_i$

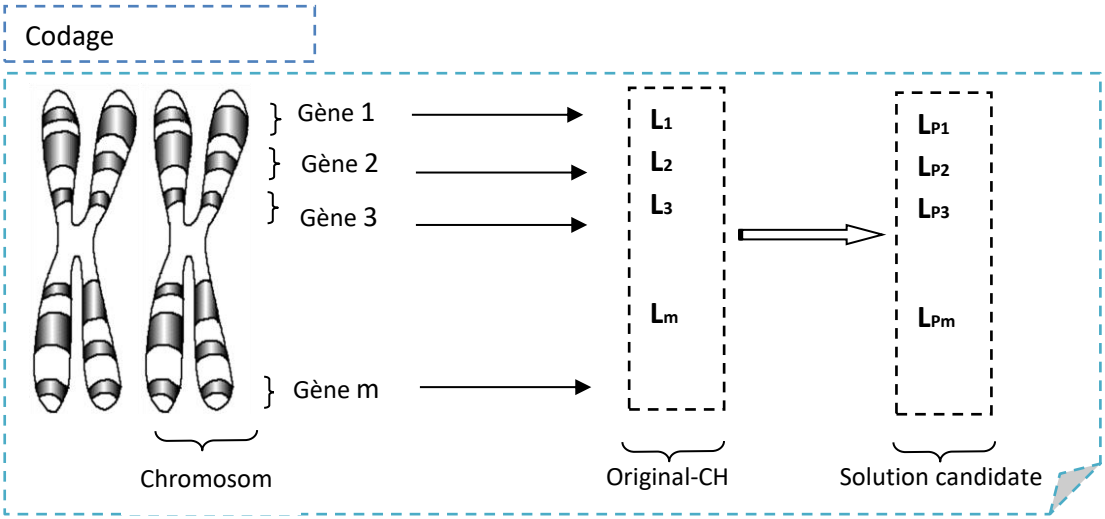



**Figure 11: Codage du problème dans SEC.**

Les chromosomes de nos populations désigneront alors les différentes permutations des listes du chromosome d'origine, étant donné que chaque permutation entre les différentes listes des positions des caractères du texte clair est un texte chiffré et une solution envisageable pour notre système.

## 2. Algorithme

### Étape 1 : Population Initiale

La première population de notre algorithme est construite manuellement en fixant un entier q qui présentera la taille de chaque population.

Nous devons donc générer q différentes solutions potentielles $X_1$, $X_2$, … Xq en effectuant q permutations des listes Li (i entre 1 et n) afin d'avoir la population initiale $P_0$.

### Étape 2 : Évaluation des individus

Dans cette étape; nous essayons de valoriser les différentes solutions de la population courante afin de pouvoir choisir la meilleure solution et sur laquelle on a pu effectuer une meilleure modification entre les fréquences d'apparitions des caractères.

L'objectif principale derrière cette modification est de :

⇨ Maximiser la différence entre la fréquence d'apparition initiale et finale des caractères pour chaque solution.

Ceci peut être traduit en mathématique par la fonction suivante :

$$F(Xj) = \sum_{i=1}^{n} |Card(Lji) - Card(Li)|$$

**Étape 3 : Sélection des meilleurs individus**

La technique proposée afin de sélectionner les meilleurs chromosomes de la population courante est la méthode de la roulette. Comme ce qui a été dit dans le chapitre 4, ce processus permet d'attribuer à chaque individu un secteur de la roulette dont la surface de l'angle est proportionnelle à sa performance et en se focalisant sur un nombre aléatoire de distribution uniforme entre 0 et 1. Ces règles probabilistes sont très utiles dans notre cas d'utilisation dans la mesure où il le rend plus complexe et non déterministe.

**Étape 4 : Croisement**

L'enrichissement de notre population par les nouveaux éléments doit prendre en considération les caractéristiques initiaux de nos chromosomes, d'où le choix de l'opérateur de croisement MPX qui représente la meilleure méthode de sélection dans le cas de notre système, permettant ainsi de garantir la disjonction entre les listes de chaque chromosome.

Cet opérateur est généralement appliqué sur 60% à 100% des individus sélectionnés.

**Étape 5 : Mutation par transposition**

L'application de cet opérateur consiste à faire inverser la position de deux gènes choisis aléatoirement pour l'un des chromosomes issus du croisement. Le taux d'application de cet opérateur génétique reste faible entre 0,1% à 5%.

**Étape 6 : Vérification de la condition d'arrêt.**

Une fois avoir la nouvelle génération issue du croisement et de la mutation, on passe à l'évaluation des nouveaux individus afin de vérifier la condition d'arrêt prédéfinie par le système.

On continue nos itérations jusqu'à avoir la meilleure solution possible.

### 3. Condition d'arrêt

En théorie, la convergence de la fonction d'évaluation est assurée car la valeur de cette fonction ne peut pas dépasser la taille du message claire à une constante multiplicative pas très faible.

Ce constat, peut être aussi prouvé mathématiquement en essayant de définir la borne inférieure et supérieure de la fonction d'évaluation :

Démonstration :

$$\sum_{i=1}^{n} |Card(Lji) - Card(Li)| < \sum_{i=1}^{n} Card(Lji) + Card(Li)$$
$$\leq \sum_{i=1}^{n} Card(Lji) + \sum_{i=1}^{n} Card(Li)$$
$$\leq 2 * m$$

Sur le plan expérimental, la condition d'arrêt est bien vérifiée pour une valeur très proche de la valeur maximale souhaitée pour la fonction d'évaluation.

### 4. La clé du système

Le système génère la clé permettant de déchiffrer l'information chiffré. Elle s'agit du chromosome final utilisé dans le processus de chiffrement. Ceci dit, pour chaque texte chiffré nous aurons une clé différente ce qu'on appelle clé de session.

La robustesse du système réside dans la complexité de la recherche de la clé qui est exponentielle et aussi dans le fait que la clé n'est utilisée qu'une seule fois et son attaque ne sera pas prise en compte pour les futurs messages

### 5. Processus de Déchiffrement

La récupération de l'information initiale à partir du chiffré et de la clé de session est définie dans les deux étapes suivantes représentant le processus de déchiffrement :

**Étape 1 :**

Cette première étape consiste à retrouver les listes et les caractères du texte clair mais dans un ordre perturbé.

Ceci est fait en appliquant le même principe de codage réalisé dans le processus de chiffrement mais cette fois-ci sur le texte chiffré.

Considérons alors que les caractères trouvés sont : $C'_1, C'_2, C'_3, \ldots, C'_n$

Et leurs listes des positions dans le texte chiffré sont : $L'_1, L'_2, L'_3, \ldots, L'_n$

**Étape 2**

Le travail qui reste à faire dans cette étape sera de retrouver l'ordre initial de chaque caractère et sa liste des positions dans le texte clair. Pour cela; nous faisons appel à la clé symétrique du système désignée par la séquence de la permutation utilisée dans le chiffrement : $P_1, P_2, \ldots, P_n$ telle que $Pi \in \{1,2, \ldots, n\}$ Ainsi, l'ordre principal est donné par la formule suivante : $C'_i = C_{Pi}$

Ce qui veut dire :

Le caractère $C'_{P1}$ correspondra à la liste $L'_1$ dans le texte clair.

Le caractère $C'_{P2}$ correspondra à la liste $L'_2$ dans le texte clair.

…

Le caractère $C'_{Pm}$ correspond à la liste $L'_m$ dans le texte clair.

Ainsi, nous obtenons le message clair M.

**Exemple :**

Afin d'avoir une idée sur le modèle du chiffré et la clé génétique générée par le système, nous proposons le texte clair suivant :

Le projet d'organiser une Coupe du monde commence dès la création de la Fédération internationale de football association (FIFA) en 1904. En 1906, la première édition initiée par le dirigeant néerlandais Carl Hirschmann est programmée en Suisse et quatre poules de quatre équipes en guise de premier tour sont mises en place. Mais lors de la clôture des confirmations d'inscriptions pour les seize sélections invitées, le 31 août 1905, aucune fédération ne confirme sa participation et le projet est enterré. Avec la mise en place d'un tournoi olympique de football à partir de 1908, Hirschmann veut procéder à la reconnaissance de ce tournoi olympique comme championnat du monde de football amateur. L'idée est validée lors du congrès de la FIFA en 1914, mais la Première Guerre mondiale bloque cette initiative

**Figure 12 : 1 er message en clair**

L'application de SEC est généralement effectuée après avoir appliqué un brouillage du texte clair en se basant sur des méthodes de chiffrement simple et qui sont liées à la cryptographie classique. Le chiffré donné ci-dessous, est le résultat de chiffrement SEC en se basant sur un brouillage utilisant un chiffrement affine défini par **F(x)= ax+b** ou **a** = 1 , **b** = 255 et x est le code ASCII du caractère du message initial.

**Figure 13 : 1er message chiffré**

## 3. SEC Binaire

### 1. Motivation

Après avoir posé la première structure du système SEC, les prochains efforts mises en place afin d'accomplir le travail et renforcer la robustesse du système contre tout type d'attaques.

Le SEC Binaire est une deuxième version de ce système dont l'objectif principal est de rendre le brouillage des caractères initiaux implicite dans l'étape du codage, étant donné qu'il repose sur des techniques de la cryptographie classique. Du point de vue technique, on cherche à partir de cette nouvelle réflexion à représenter l'information de manière aussi efficace que possible pour garantir sa confidentialité.

Comme son nom l'indique, le SEC Binaire repose sur un codage binaire du texte clair et un entier k>1 et différents de 8*b (b entier > 0).

Ensuite, on effectue une division de ce texte en blocs binaire de K bits, on dit aussi k-blocs, qui seront les éléments de base de notre texte et qui remplaceront les ci de l'ancienne version du système. Le travail qui reste à faire avant d'appliquer l'algorithme, est de définir les listes des occurrences de chaque k-blocs dans le texte d'origine et réaliser le vecteur initial des prochaines populations pour passer ensuite aux autres étapes de l'algorithme.

**2.** Exemple

Supposons que le Message à chiffrer est constitué des lignes qui suivent:

Le message en binaire correspondant à ces lignes est :

```
0100010001100101011101010111100000100000011101000111100101110000011001010111
0110010000001100100011001010010000001110100011001010110001101101000011011100 1
1010010111000101110101011001010111001100100000011100110110010100100000011001 0
0011010010111001101110100011010010110111001100111011101010110010101101110011 1
0100001000000110010101101110011001110110110001101111011000100110000101101110 0
1110100001000000111010001101111011101010111010001100101011100110010000001101 1
0001100101011100110010000001101101000101110111010001101000011011110110010001 1
0010101110011001000000110010001100101001000000110001101101000011010010110011 0
0110011001110010011001010110110101100101011001010110111001110100001000000110 1
1010110111101100100011001010111001001101110011001010111001100100000011000110 1
1011110110111001101110011101010110010101110011001000000010000000
```

**Figure 14 : Message clair en binaire**

En décomposant le message à des blocs de 10 bits on obtient 63 blocs différents.

Ci-dessous ces 63 blocs avec leurs listes de positions :

| | | | |
|---|---|---|---|
| 0100010001: {1} | 0111000101: {17} | 0110001001: {37} | 1001100110: {62} |

| | | | |
|---|---|---|---|
| 1001010111: {2} | 1101010110: {18} | 1000010110: {38} | 0110011100: {63} |
| 0101011110: {3} | 0101011100: {19,55,75} | 1011110111: {42} | 1001100101: {64} |
| 0000100000: {4,32,40} | 1100100000: {20,56,76} | 0101011101: {43} | 0110110101: {65} |
| 0111010001: {5,41} | 0111001101: {21,25} | 0001100101: {44,72} | 0101011011: {67} |
| 1110010111: {6} | 1001010010: {22,58} | 0111001100: {45} | 1001110100: {68} |
| 0000011001: {7,23} | 0001101001: {24} | 1000000110: {46} | 1011010110: {70} |
| 0101110011: {8,48} | 1101000110: {26} | 1100011001: {47} | 1111011001: {71} |
| 0010000001: {9,49,69} | 1001011011: {27} | 1011010001: {50} | 0111001001: {73} |
| 1001000110: {10,54} | 1001100111: {28} | 0111011101: {51} | 0110001101: {77} |
| 0101001000: {11} | 0111010101: {29} | 0001101000: {52} | 1011110110: {78} |
| 0001110100: {12} | 1001010110: {30,66} | 0110111101: {53} | 1110011011: {79} |
| 0110010101: {13,33,81} | 1110011101: {31,39} | 0110010001: {57} | 1001110101: {80} |
| 1000110110: {14} | 1011100110: {34,74} | 0000011000: {59} | 1100110010: {82} |
| 1000011011: {15} | 0111011011: {35} | 1101101000: {60} | 0000001000: {83,84} |
| 1001101001: {16} | 0001101111: {36} | 0110100101: {61} | |

Après avoir effectué 48 itérations on obtient le chiffré en binaire suivant:

```
0000001000110010000001110011010110111101011001000100000110001101101000011010011
0110011001100110011100100110010100100000011001000110010100100000011101000110011
0101010111001100001000000111010001111001011100000110011001010111111011010000101
0111100000011001010001000110010100100001101001110100011010010110111001100111011
1011110110010001100111010101100101011011100111011011100110011101101110011001111
0110111101011001000100011011111011000100110000101101011110111010101110100011001
0101101001011001100110011100110010000001101100011001101101000101100111000111011
0001111001011101110111011001010111000110100010001101101000011011100111010010010
1010010110110101010101101110010110111001110101111001101110011001101011110110011
1100001011000010110110101011001110101010111010001111001011101010111001111011001
1011010110011000110110010001100111001001110011001101001100110010
```

**Figure 15 : le chiffré du message en binaire**

La clé de chiffrement est :

23 21 22 17 18 19 20 16 11 12 13 14 15 46 47 49 55 56 57 1 2 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 3 4 5 6 7 8 9 10 50 51 48 59 58 61 60 54 53 52 62 0

La conversion du message chiffré en hexadécimal est :

02320735BD64418DA1A59999C99481919481D1955CC207479706657DA15E0
651194869D1A5B99DBD919D595B9DB99DB99DBD6446F62616BDD5D195A
5999CC81B19B459C74797776571A23686E749A5B556E5B9D79B99AF671616
D59D574797573D9B598D919C9CCB32.

### 3. Conclusion

Comme vous pouvez remarquer dans l'exemple présenté ci-dessus, l'application du SEC sur un texte binaire change complètement les caractères du texte en clair ainsi que leurs fréquences. Par conséquent, la cryptanalyse par étude des fréquences des caractères devient impossible.

# CHAPITRE 5

# CONCEPTION DE LA NOUVELLE FONCTION D'ÉVALUATION DE SEC

## Sommaire

## 1. Introduction

La détermination de la fonction d'évaluation est une phase très critique dans le processus de l'évolution d'une solution. En fait, dans le cadre de l'optimisation combinatoire et si on revient à la définition d'une solution optimale on va surement être ramené à définir les propriétés générales que doit inclure cette solution et qui lui offre le maximum de survie possible.

Dans notre cas les propriétés qu'on cherche à évoluer dans cette solution concernent essentiellement :

- **La fréquence d'apparition des caractères.**
- **La différence entre la solution initiale et la solution finale.**

Après avoir étudié et analysé profondément le système de chiffrement SEC, nous avons constaté qu'il est possible que certains caractères gardent les mêmes listes de positions dans le texte chiffré, quoique cette possibilité est d'une probabilité très négligeable.

Ainsi, dans cette contribution nous avons veillé à ce que la fonction d'évaluation conçue dans le SEC doit satisfaire la contrainte suivante : Aucun caractère de texte clair ne doit garder sa liste des occurrences dans le texte chiffré.

Notre nouvelle fonction a pour objectif de maximiser la différence entre les fréquences d'apparition des caractères et éliminer tous les cas qui entrainent la possibilité d'avoir une solution finale où les listes initiales sont affectées à leurs caractères d'origine.

## 2. Description

### 1. Formulation Mathématique

Soit M le message clair à chiffrer et soit $X_0$ le chromosome original (Original-CH) qui représente M ➔ Donc $X_0 = L_1\ L_2\ L_3 \ldots L_n$

Soit $Xj = L_{j1}\ L_{j2} \ldots L_{jn}$ , le nouveau chromosome à évaluer. La nouvelle formule de la fonction d'évaluation nous met face aux deux situations différentes :

⇨ 1- Ou bien le gène $L_{ji}$ coïncide avec $L_i$

⇨ 2- Ou bien le gène $L_{ji}$ est différent de $L_i$

À partir de cette nouvelle formule, nous essayons d'écarter au maximum la première situation et minimiser la valeur d'évaluation de la solution contenant ce cas de figure afin qu'elle n'aura aucune chance de survie dans les prochaines générations.

Ainsi, la nouvelle fonction objective proposée est exprimée sous la forme suivante :

$$F(X_j) = \sum_{i=1}^{n} f(Xji)$$

Avec :

$$f(Xji) = \begin{cases} -Card(Lji), & Lji = Li \\ |Card(Lji) - Card(Li)|, & Lji \neq Li \end{cases}$$

Par conséquent, la permutation qui représente l'individu Xji sera évaluée en fonction de la position de chaque liste Lji dans cette permutation. Deux valeurs sont possibles, à savoir :

- Une valeur positive qui représente toujours la différence entre le cardinal de la nouvelle et l'ancienne liste du caractère Cj, et qu'on cherche à maximiser à travers cet algorithme.
- Une valeur négative qui représente le cardinal de la liste Lji quand celle-ci coïncide avec la liste d'origine du caractère Cj. Cette valeur aura, en revanche, un impact négatif sur la somme de la fonction d'évaluation pour l'individu en question. Ce qui permettra d'augmenter sa chance d'être exclue des solutions de la prochaine population.

## 2. Application de la nouvelle fonction d'évaluation sur le SEC

Sur le plan théorique, l'application de la nouvelle formule de la fonction d'évaluation doit avoir un impact positif sur les trois volets suivants :

- La qualité de la solution obtenue.
- La convergence de l'application.
- Le temps d'exécution du système.

Les différentes expériences réalisées s'inscrivent dans ce contexte et consistent à effectuer des comparaisons entre les résultats obtenus avec l'ancienne fonction d'évaluation du système SEC et la nouvelle fonction intégrée dans ce système.

Ci-dessous des exemples des résultats tirés à partir de ces expériences.

### 3. Expériences :

Nous avons mené nos expériences sur différents textes afin de pouvoir en sortir le cas qu'on cherche à étudier : Il s'agit bien du cas où la solution optimale contient des caractères dont les occurrences coïncident avec celles du texte clair.

Pour ce faire, nous avons décidé de démarrer avec la même population initiale générée en premier temps d'une manière aléatoire.

Les résultats obtenus ne sont pas toujours très différents par rapport à la nouvelle solution car, en pratique, la probabilité de voir apparaitre l'un des caractères du message clair dans la même position initiale est faible même avec l'ancienne fonction d'évaluation. En fait, étant donné que dans le SEC [69], on cherche à maximiser la fonction d'évaluation, la solution dans laquelle un caractère garde la même liste de positions est dans la majorité des fois écartée.

Le texte suivant représente une expérience réussite parmi vingt textes choisis au hasard et qui permet de mettre en évidence notre cas d'étude :

Le texte modèle est le suivant :

```
La reconnaissance de l'écriture manuscrite n'est pas un sujet
nouveau. Il remonte àplus d'une trentaine d'années. La
reconnaissance de l'écriture manuscrite s'avère unproblème
extrêmement complexe qui n'a pas de solution satisfaisante à
ce jour. Ainsirapidement, les chercheurs ont restreint leurs
études à des problèmes particuliers enliaison avec des
applications bien définies. Parmi celles-ci on cite les
applications de lareconnaissance de l'écriture manuscrite: le
tri automatique du courrier postal, letraitement automatique
de dossiers administratifs, des formulaires d'enquêtes, ou
encore l'enregistrement des chèques bancaires.Dans le domaine
de la reconnaissance de l'écriture, les primitives peuvent
être décritescomme un moyen permettant de distinguer un objet
(mot, lettre, chiffre) d'une classed'un autre objet (mot,
lettre, chiffre) d'une autre classe. Dès lors, il est
nécessaire dedéfinir des primitives significatives lors du
développement d'un système dereconnaissance. Les primitives
sont généralement définies par expérience ou parintuition.
Plusieurs primitives peuvent être extraites. La
représentation des primitivesutilisée est une représentation
vectorielle. La taille du vecteur peut être large si ungrand
nombre de primitives est extrait. Il a été observé que
certaines primitivesextraites sont non pertinentes ou
redondantes au système de reconnaissance. Lasélection de
primitives pertinentes est par contre un problème complexe et
fait l'objet denombreuses recherches.
```

**Figure 16 : Texte Clair**

Le chiffré en utilisant l'ancienne fonction d'évaluation :

Ci-dessous nous présentons le chiffré obtenu en utilisant SEC


**Figure 17 : le chiffré obtenu en utilisant SEC**

Le chiffré en utilisant la nouvelle fonction d'évaluation :

Ci-dessous nous présentons le chiffré obtenu en intégrant la nouvelle formule de la fonction d'évaluation dans SEC :

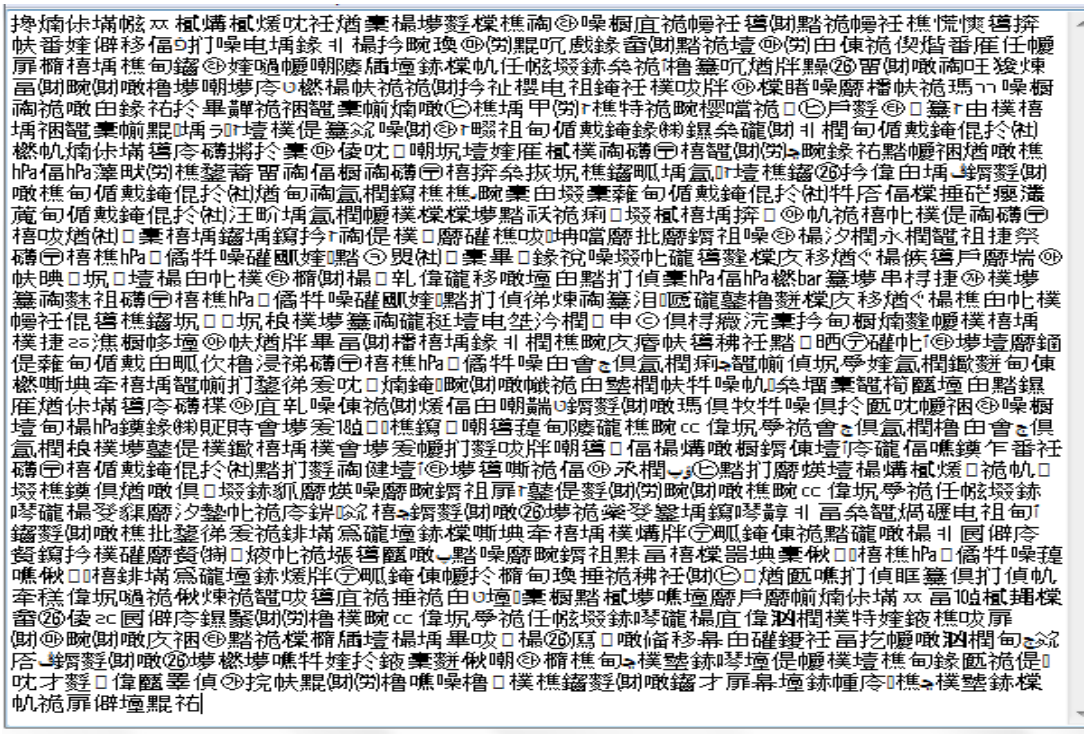


**Figure 18 : Le chiffré en utilisant la nouvelle fonction d'évaluation**

## 4. Résultats

***Qualité de la solution optimale:***

Afin de s'assurer de la qualité de la solution obtenue par notre système, nous avons défini le pourcentage de coïncidence des caractères du texte chiffré par rapport au texte clair. Ce pourcentage est calculé pour chaque caractère en fonction de sa fréquence d'apparition dans le texte chiffré et celle dans le texte initial, ce qui est traduit par la formule suivante :

Coï(Ci) = (fréquence d'apparition (Ci) dans le chiffré / fréquence d'apparition (Ci) dans texte clair) * 100

Le graphe ci-après, illustre ce pourcentage de la coïncidence de chaque caractère dans le texte clair et les deux chiffrés obtenus, tantôt avec la nouvelle fonction d'évaluation et tantôt avec l'ancienne formule de cette fonction.

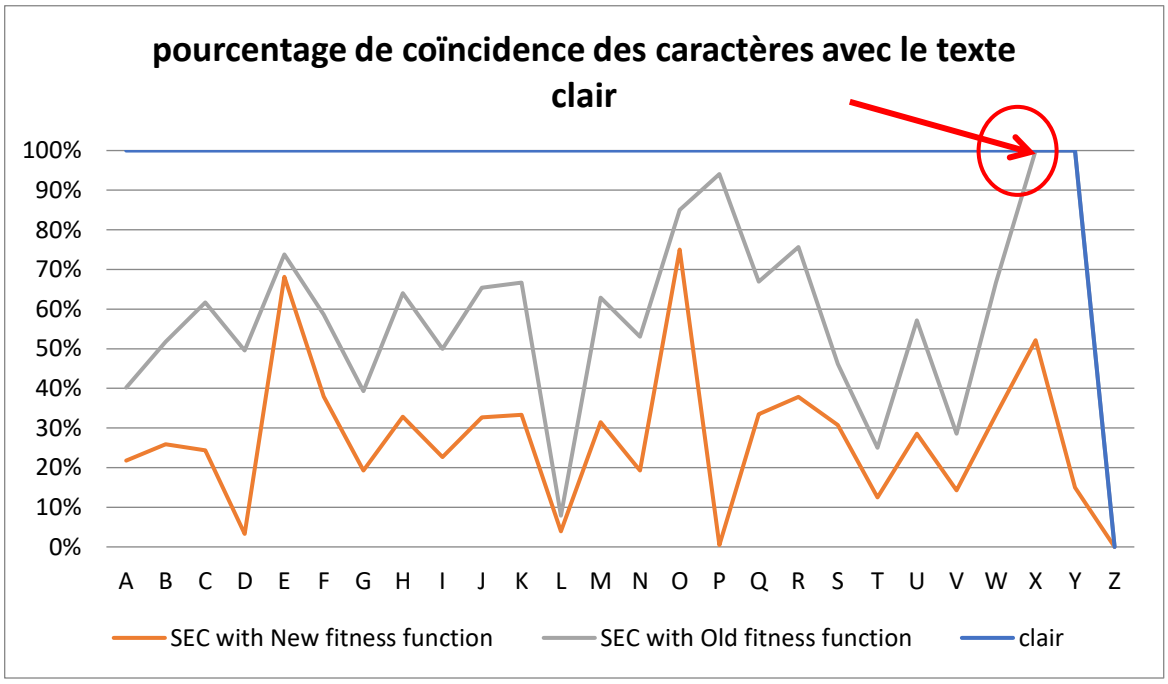



**Figure 19 : Pourcentage de coïncidence des caractères avec le texte clair**

## 5. Interprétation

Comme nous le remarquons sur le graph, le pourcentage de la coïncidence entre les positions des caractères du texte clair et celles du texte chiffré avec le système SEC utilisant la nouvelle fonction d'évaluation, est dans la plupart du temps inférieur à 40%. Cependant, dans le premier chiffré, cette valeur est plus grande et elle reste dans la plupart du temps entre 40% et 60% et parfois dépasse largement les 60%. De plus, avec la nouvelle formule,

aucun caractère ne peut avoir la même liste des positions dans le texte initial. Chose que nous avons pu avoir pour ce texte, en utilisant l'ancienne fonction d'évaluation.

## 6. Étude de la convergence

***Résultats Expérimentaux:***

La convergence de notre système de chiffrement est toujours assurée que ça soit avec l'ancienne ou la nouvelle fonction de l'évaluation. Cependant, notre objectif à travers cette expérience est de comparer :

1- Le nombre des itérations au bout duquel cette convergence a été atteinte.

2- La valeur maximale de la solution retenue par le système.

3- Et aussi, l'écart entre les populations d'une itération à l'autre

Les graphes ci-dessous montrent les résultats ressortant pour ces trois points.

Le nombre des itérations et la valeur de la convergence pour les deux fonctions d'évaluation sont donnés dans le graphe 1, tandis que les résultats de l'écart entre les populations sont donnés au niveau du graphe 2.

**Mesure de la Performance de la nouvelle fonction d'évaluation :**

Les résultats de ces expériences sont présentés dans le graphe suivant :

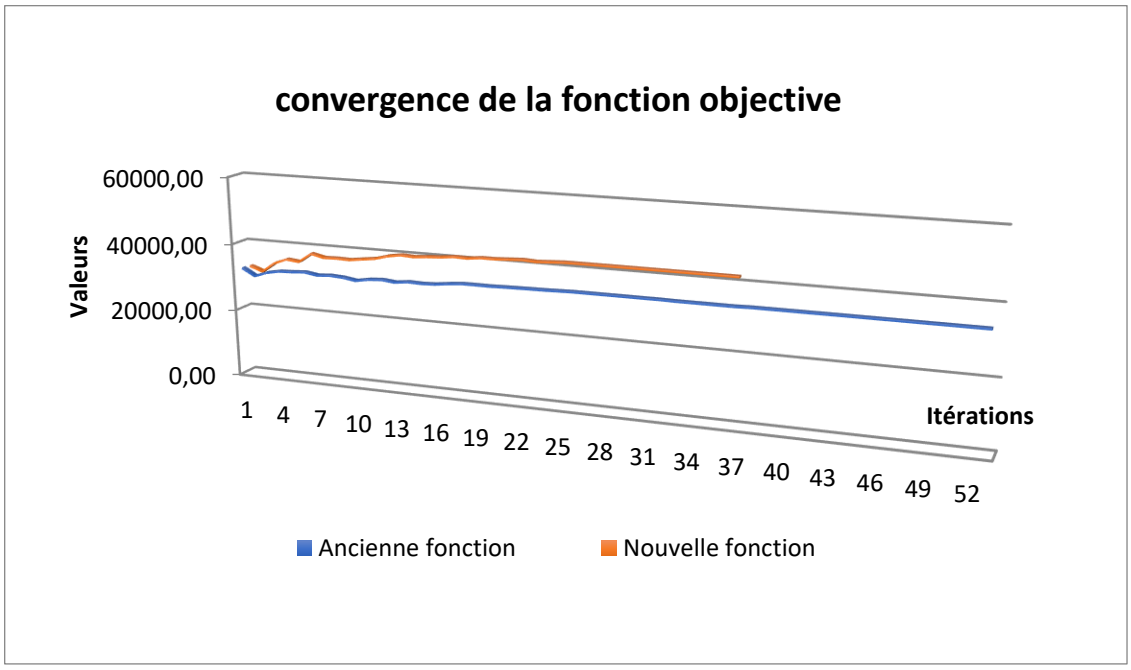



**Figure 20 : convergence de la fonction objective**

***Interprétation du graphe 1***

La lecture du graphe1 permet de faire une comparaison entre les différentes valeurs de la fonction d'évaluation pour chaque itération, ce qui permet de déduire la convergence du système SEC avec la nouvelle et l'ancienne fonction objective.

Pour la première version du système (courbe en bleu), la convergence est atteinte au bout de la 52ème itération, tandis que la convergence dans le nouveau système (courbe rouge) est atteinte plus tôt et ceci au bout de la 37ème itération. La différence est bien claire et l'exécution du système devient plus rapide et plus performante qu'avant.

**Mesure de l'écart entre les populations tout au long l'exécution de chiffrement :**

L'écart type mise en place dans cette expérience, nous a permis de synthétiser les résultats numériques de la fonction d'évaluation d'une population à une autre.

Ce calcul est réalisé à la fin de chaque itération afin de définir la différence entre les individus de la population en cours et ceux de la nouvelle population. Par conséquent, plus l'écart est faible plus la population est homogène et converge vers la même solution optimale. À l'inverse, s'il est plus important, les individus sont dispersés et moins resserrés.

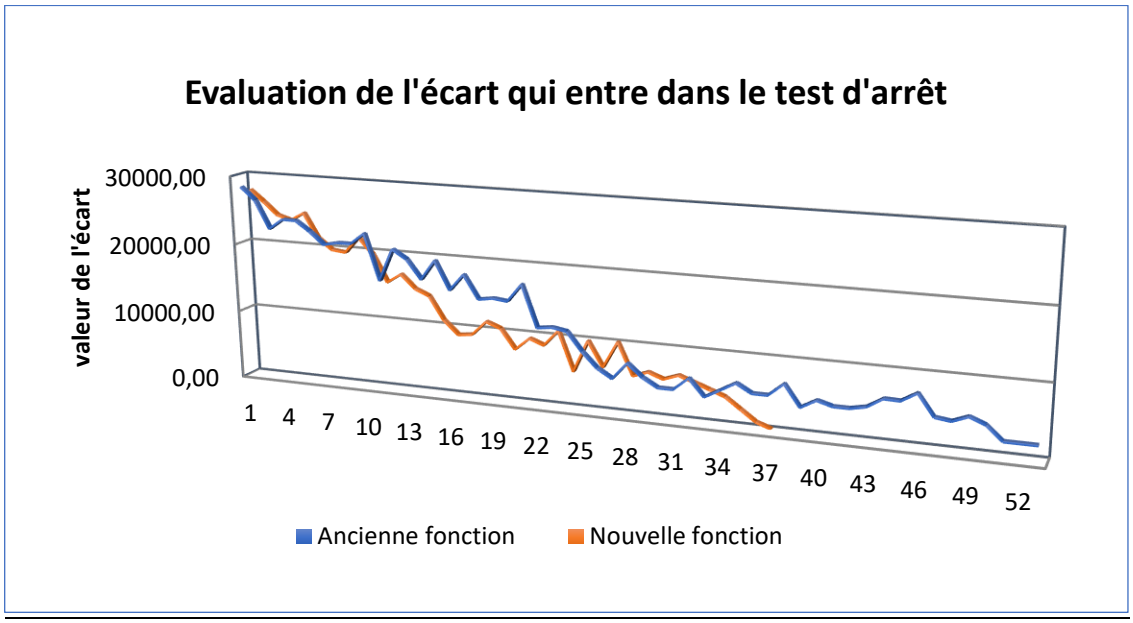



**Figure 21 : Évaluation de l'écart qui entre dans le test d'arrêt**

***Interprétation du graphe 2***

Par rapport aux résultats donnés dans le graphe 2, nous constatons que les deux courbes de l'écart sont en décroissance d'une itération à une autre. Ce qui signifie que les individus

de la population se regroupent autour de la solution optimale d'une itération à une autre. Cependant, ce qui fait la différence entre ces deux courbes, c'est plutôt la valeur de l'écart sur laquelle l'exécution du système se termine. On remarque que cette valeur atteint 0 à la fin de l'exécution du SEC avec la nouvelle fonction d'évaluation ce qui signifie que toute la population tend vers la même solution et par conséquent on arrive à une convergence Totale du système. L'utilisation de l'ancienne fonction, par contre, nous ramène à une convergence de la population au bout d'une valeur de l'écart qui vaut 967 (pour cet exemple). Une valeur faible par rapport à l'ordre de grandeur des valeurs des premières populations, néanmoins, il reste loin d'une convergence totale qui représente l'un des objectifs de cet algorithme.

**Temps d'exécution :**

Au cours des expériences réalisées auparavant, nous avons mise en place un compteur de temps permettant de calculer le temps passé pour terminer l'exécution d'un chiffrement que ça soit au niveau de l'ancien système où pour cette nouvelle version.

Dans le tableau suivant, nous avons mis les valeurs retenues pour un échantillon de textes clairs ayant des tailles différentes. Ces valeurs sont renseignées et présentées en graphe par ordre croissant de celles-ci

| | | Temps d'exécution (ms) | | | |
|---|---|---|---|---|---|
| Taille du texte clair | **Système de Chiffrement** | **Expérience 1** | **Expérience 2** | **Expérience 3** | **Expérience 4** |
| 1256 caractères | SEC | 47 | 44 | 43 | 42 |
| | SEC avec la nouvelle fonction d'évaluation | 26 | 27 | 27 | 29 |
| 2702 caractères | SEC | 53 | 56 | 50 | 52 |
| | SEC avec l'ancienne fonction d'évaluation | 31 | 35 | 28 | 30 |
| 5890 caractères | SEC | 57 | 56 | 58 | 57 |
| | SEC avec la nouvelle fonction d'évaluation | 39 | 40 | 42 | 37 |
| 10039 caractères | SEC | 65 | 69 | 64 | 66 |
| | SEC avec l'ancienne fonction d'évaluation | 42 | 43 | 46 | 45 |

**Table 9: Tableau du temps d'exécution du chiffrement avec et sans la nouvelle fonction d'évaluation**

***Représentation graphique :***

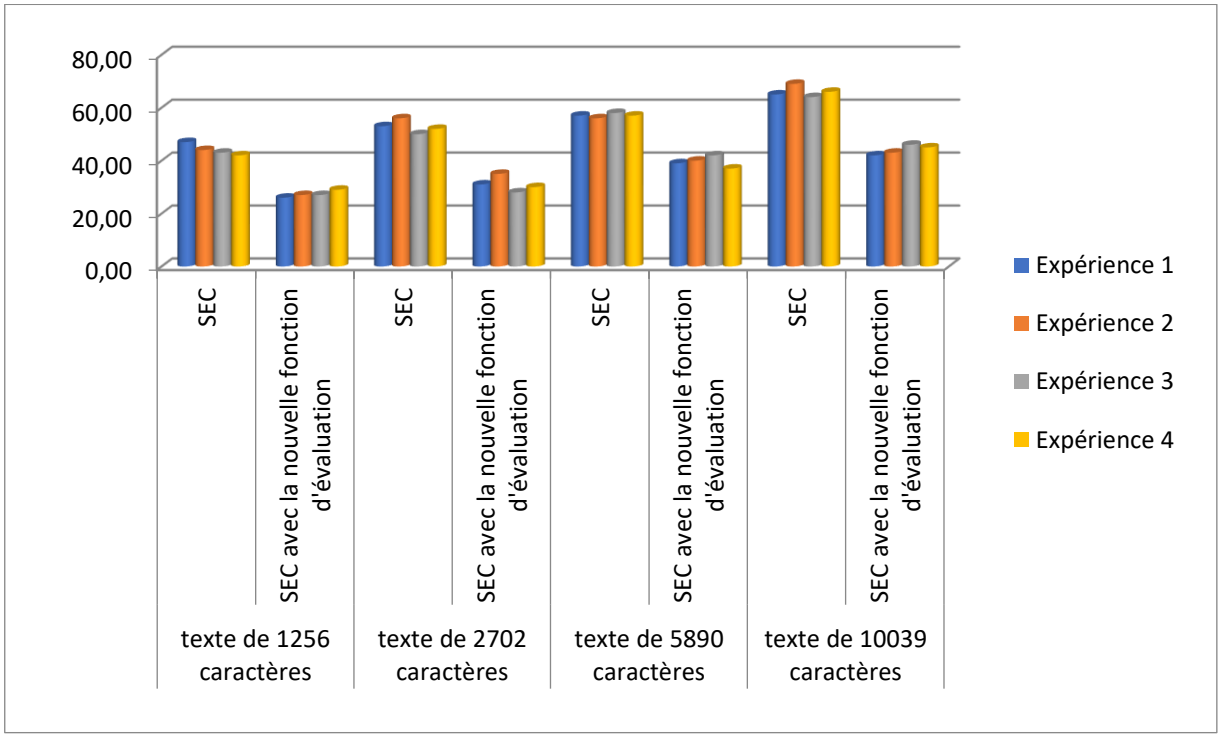



**Figure 22 : Comparaison du temps d'exécution du chiffrement dans le SEC avec et sans la nouvelle fonction d'évaluation**

***Interprétation :***

La représentation graphique de la comparaison entre le temps découlé pour exécuter un chiffrement du système SEC en sa version initiale et celle intégrant notre contribution, a mis en évidence la différence énorme de ce paramètre entre les deux exécutions et l'amélioration efficace qu'a apportée cette nouvelle fonction d'évaluation sur le système.

Donc, peu importe la taille du message ni la performance du processeur dans chaque expérience, le temps d'exécution d'un processus de chiffrement utilisant le nouveau système fera sujet d'une réelle optimisation dans le cadre de la performance du système SEC, et ce, en à peu près la moitié du temps.

## Conclusion

Suite aux différentes expériences réalisées sur des textes choisis d'une manière arbitraire, l'intégration de la nouvelle fonction d'évaluation dans le système de chiffrement SEC, est d'une grande utilité dans la mesure où elle permet de garantir une meilleure solution vis-à-vis aux critères de chiffrement qu'on voulait avoir à travers ce système, et qui porte

principalement sur la maximisation de la différence entre les positions des caractères dans le message initial, et le chiffré.

D'autres résultats très intéressants ont été ressortis suite à ces expériences, et qui montrent l'amélioration de la performance du système dû à la mise en place de cette nouvelle fonction d'évaluation. Par conséquent, le système de chiffrement SEC devient plus rapide en termes de temps d'exécution et consomme moins d'espace mémoire vu que le nombre des itérations a été diminué. Outre, la convergence de ce nouveau système est obtenue avec une meilleure valeur de l'écart, ce qui explique que la solution obtenue peut être considérée comme une solution optimale, sinon globale pour le système. Et par conséquent, nous pouvons confirmer que la nouvelle fonction permettait de satisfaire la contrainte majeure exigée par ce type d'algorithme et réaliser un meilleur chiffrement.

# CHAPITRE 6

# NOUVELLE FONCTION D'ÉVALUATION POUR SEC INTÉGRANT LE PROBLÈME DES PARTITIONS

## Sommaire

## 1. Introduction

Nos travaux de recherche continuent à récolter ses fruits afin d'aboutir à un système complet, complexe à résoudre et aussi simple à manipuler.

Notre nouveau travail présente une nouvelle version du système de chiffrement symétrique SEC que nous avons nommé ASEC (Advanced Symetrical Evolutionary Ciphering).

Dans cette version, nous avons pu reformuler la conception du système SEC en introduisant le problème des « partitions ». Ceci dit, une nouvelle fonction d'évaluation a été réalisée et aussi de nouveaux opérateurs génétiques ont été conçus.

Notre objectif à partir de ce travail est non seulement effectuer la meilleure permutation entre les positions des caractères de ce message, mais plutôt de rendre impossible son analyse statistique par étude des fréquences d'apparition qui représente la seule étude possible afin de casser le chiffré dans le cas où on travaille avec la version de SEC sans brouillage ni caractères spéciaux et ponctuation.

Notre contribution dans ce cadre était très intéressante dans la mesure où elle nous a ramené à changer complètement le contenu des listes des positions des caractères au sein du même système, et ce en effectuant une nouvelle partition aléatoire sur les listes initiales. Cependant, le reste des itérations continues bien comme avant sauf que cette fois-ci on se base sur cette nouvelle partition. En outre, les mécanismes misent en place dans le cadre de ce travail sont effectués sans perte d'information permettant ainsi de garder la réversibilité du système essentielle pour le déchiffrement.

## 2. Description

### 1. Codage

Comme dans le système SEC, le message à chiffrer est présenté par un ensemble des listes disjointes. Chaque liste correspond aux différentes positions de l'un des caractères de ce message.

Soit M le message à chiffrer, constitué de n différents caractères : $C_1, C_2, C_3, \dots, C_n$.

Et soient $L_1$, $L_2$, $L_3$, …, $L_n$ leurs listes des positions dans le texte clair.

Notons que :

- ⇨ Li ∩ Lj est vide, i,j ϵ [1,n].
- ⇨ Li est un élément de la partition initiale de l'ensemble {1, 2, …, m} qui représente l'ensemble des positions du texte clair avec m est la longueur de ce texte.
- ⇨ Nous rajoutons aussi une nouvelle propriété implémentée par défaut dans le système SEC lors de la construction de nos listes et qui est très intéressante à respecter dans cette nouvelle version. Elle s'agit de l'ordre du contenu des listes que nous veillons à le garder toujours croissant.

Le message M prendra toujours la présentation vectorielle suivante:

$$M = \begin{Bmatrix} (C1, L1) \\ (C2, L2) \\ \\ \\ (Cn , Ln) \end{Bmatrix}$$

**2. Population initiale**

Comme dans SEC, la construction de la population initiale consiste à générer d'une manière aléatoire q différentes permutations des listes Li, telle que q est la taille de la population fixée au préalable.

Chaque permutation présentera une solution potentielle pour notre système.

Notons par : {Xj / (1≤ j ≤ q)} l'ensemble des individus de cette population

Et $X_0$ le chromosome d'origine qui représente le texte clair.

**3. Évaluation des individus :**

Dans cette étape, l'évaluation des individus a pris une autre forme afin de suivre les contraintes de notre nouvel objectif. Il s'agit bien d'avoir un chiffré complètement aléatoire et indépendant du texte d'origine. En effet, dans cette nouvelle conception nous ne sommes plus face à une permutation entre les listes des différents caractères, mais nous

souhaitons modifier même le contenu de ces listes. D'où l'intérêt d'avoir une nouvelle fonction d'évaluation permettant d'intégrer ce nouvel aspect et évaluer son influence sur la qualité de la solution trouvée qui, normalement, doit être beaucoup plus optimale que celle trouvée au cours de l'ancienne version.

La nouvelle méthode d'évaluation est définie dans 3 étapes :

### i. Réalisation des partitions :

Dans cette étape, nous essayons de générer à partir de chaque individu une nouvelle partition des positions des caractères en se basant sur l'ordre des listes de cet individu.

En fait, si on considère l'ensemble fini {1, 2, …, m} qui représente les positions des caractères dans le texte clair, alors $\{L_1, L_2, L_3, \ldots, L_n\}$ est la partition initiale construite à partir de ce texte et $L_i$ sont ses différents éléments. Construire une nouvelle partition de l'ensemble de départ revient à modifier le contenu des Li. Cette partition doit cependant, respecter les caractéristiques que nous avions au préalable dans la partition Initiale, et qui sont :

- ⇨ Listes disjointes
- ⇨ Listes triées au préalable, dès leur création et sans application d'un processus de trie pour ne pas perde la propriété de réversibilité.
- ⇨ Nombre des listes de cette partition reste le même que celui de la partition initiale et correspond au nombre des caractères du message claire.

Pour ce faire, on se base sur la chaine des nombres construite à partir de la permutation donnée par chaque solution potentielle. Ensuite, on réalise le découpage de telle sorte à ce que chaque nouvelle liste commence par un petit nombre et se termine par un grand nombre et ainsi de suite jusqu'à la reconstruction de toutes les 'n' nouvelles listes.

### ii. Évaluation des individus en fonction de leurs partitions.

Une fois ayant obtenu pour chaque individu de la population courante une nouvelle partition, nous passons à leur évaluation.

Étant donné que nous cherchons une partition qui ne permet pas l'analyse statistique de notre chiffré, la meilleure solution sera donc d'avoir un chiffré dont tous les caractères ont presque la même probabilité d'apparition.

Cela ne peut être réalisé que si toutes les listes ont presque la même taille, c'est-à-dire contiennent toutes un nombre très proche de la moyenne des positions possibles pour une liste donnée.

Mathématiquement parlons :

Nous considérons que μ est la moyenne des positions que peut contenir une liste dans un texte de n différents caractères et de taille m.

$$\mu = [m/n]$$

et soit $E_j = \{e_{j1}, e_{j2}, \dots, e_{jn}\}$ la nouvelle partition créée pour l'individu Xj.

Donc, on cherche que :

$$| \text{Card}(e_{ji}) - \mu | = \varepsilon \quad \text{avec } \varepsilon \to 0$$

Ce qui revient à minimiser la somme de la fonction d'évaluation suivante :

$$F(Xj) = \sum_{i=1}^{n} | \text{Card}(eji) - \mu |$$

### iii. Stockage de la meilleure solution :

Nous prévoyons dans cette étape d'enregistrer l'individu et la partition ayant donnée cette meilleure valeur d'évaluation afin de la comparer par la suite avec les meilleures solutions des générations qui suivent.

Mémoire d'évaluation

| Permutation | Partition | Valeur de fitness |
|---|---|---|

**Figure 23 : Structure de la mémoire d'évaluation**

## 4. Sélection des meilleurs individus

Dans cette étape, nous restons sur la même méthode de sélection utilisée dans le système SEC sauf que dans notre cas nous auront recourt à appliquer une fonction d'adaptation sur les valeurs données par la fonction d'évaluation afin de les adapter

avec le fonctionnement défini par la sélection de la roulette et qui consiste à récupérer celui dont la valeur est la plus grande.

La fonction d'adaptation à prendre en compte dans la sélection de la roulette est :

$$f(Xj) = \frac{1}{F(Xj)}$$

**5. Croisement MPX**

Nous appliquons toujours le croisement MPX afin de construire la nouvelle génération car c'est le meilleur croisement qui respecte la contrainte de la disjonction des nouvelles listes.

**6. Mutation**

Dans l'ancienne version de SEC, l'application de cet opérateur consistait à échanger les positions de deux gènes choisis d'une manière aléatoire pour l'un des chromosomes issus du croisement. Ce qui revient à créer une nouvelle permutation. Cependant, dans cette version, cette étape représente l'axe principal autour duquel s'est basée la nouvelle conception de notre système.

Dans cette étape, nous appliquons la mutation avec partition (Mutation With Partition MWP) qui fait partie des nouveaux mécanismes misent en place dans ce travail. Cette mutation consiste à appliquer la partition enregistrée dans la mémoire d'évaluation et qui représente la meilleure solution jusqu'à l'itération courante.

L'application de la nouvelle partition sur le système passe par les étapes suivantes :

**i. <u>Restructuration des nouvelles listes</u>**

Cette étape a pour objectif d'affecter pour chaque nouvelle liste un ordre en fonction de l'ordre des listes dans l'individu élu.

Considérons par exemple que Xj est l'individu ayant la meilleure partition.

Xj représente la permutation de base pour listes initiales:

$Xj = Lj_1 \rightarrow Lj_2 \rightarrow Lj_3 \ldots \rightarrow Lj_n$

Donc la représentation vectorielle du texte correspondante à cet individu est la suivante :

$$M' = \{(C_1, L_{j1}), (C_2, L_{j2}), \ldots, (C_n, L_{jn})\}$$

En remplaçant les anciennes listes par la nouvelle partition, nous aurions la présentation suivante :

$$M' = \{(C_1, e_{j1}), (C_2, e_{j2}), \ldots, (C_n , e_{jn})\}$$

Cependant, lors de déchiffrement nous auront un ordre différent de celui qu'on vient de présenter car la liste qui contient la première position ne correspond pas forcement au caractère $C_1$ et la même chose pour les autres correspondances.

Ceci conduit à effectuer un chiffrement en fonction de cette distribution afin de retrouver l'ordre exacte de ces vecteurs. La nouvelle distribution à partir de ce chiffré sera donc présentée comme suite :

$$M' = \{(C'_1, e'_1), (C'_2, e'_2), \ldots, (C'_n , e'_n)\}$$

L'information intéressante à stocker, est l'ordre des listes $e_{ji}$ dans la nouvelle distribution. Par exemple si $e_{j1}$ correspond à la liste $e'_3$ donc on stocke 3, ensuite si $e_{j2}$ correspond à liste $e'_5$ on stocke 5 dans la $2^{ème}$ position et ainsi de suite jusqu'à avoir la permutation correspondante à tous les $e_{ji}$ en fonction de son ordre dans la nouvelle distribution.

Cette permutation sera rajoutée à la clé du système que nous avons nommé « clé de Mutation »

#### ii. Propagation de la nouvelle distribution des listes sur la génération courante

La deuxième partie de la mutation peut être considérée comme une projection de la génération courante sur la nouvelle distribution. Cela veut dire que si $X_k$ est un des individus de cette génération, $X_k$ sera donc présenté par la même permutation mais avec les listes $e'_{ki}$ au lieu de $L_{ki}$

### 7. Évaluation et vérification de la condition d'arrêt.

Une fois avoir la nouvelle génération issue du croisement et de la mutation, on passe à l'évaluation des nouveaux individus afin de vérifier la condition d'arrêt prédéfinie par le système.

On continue nos itérations jusqu'à avoir la meilleure solution possible.

### 8. Condition D'arrêt

Comme pour l'ancienne fonction d'évaluation, la nouvelle fonction est aussi bornée et admet une borne inférieure qui est 0. Par conséquent la convergence du système est aussi assurée

### 9. Clé du système

La clé du système est composée de 4 éléments :

1- L'individu élu pour réaliser la partition.

2- Les cardinaux des listes de l'individu élu.

3- La clé de mutation qui correspond à la permutation représentant la correspondance entre l'ancienne distribution des listes et la nouvelle distribution

4- Le Final_CH qui représente la clé génétique.

### 10. Déchiffrement

Le processus de déchiffrement est défini sur deux étapes :

**Étape 1 :**

Consiste à utiliser la clé génétique afin d'avoir les listes de la nouvelle partition appliquée par le système ainsi que l'ordre de ces listes. Ceci doit être effectué en se basant sur la même procédure de déchiffrement donnée pour le système SEC.

Considérons alors que les caractères trouvés sont : $C''_1, C''_2, C''_3, \dots, C''_n$

Et leurs listes des positions dans le texte chiffré sont : $e''_1, e''_2, e''_3, \dots, e''_n$

Nous essayons alors de retrouver l'ordre initial de chaque caractère et sa liste des positions suite à la nouvelle partition. Pour cela; nous faisons appel à la clé génétique du système désignée par la séquence de la permutation utilisée dans le chiffrement : P1, P2, …, Pn telle que $Pi \in \{1,2, \dots, n\}$ Ainsi, l'ordre principale est donnée par la formule suivante : $C''_i = C'_{Pi}$

Ce qui veut dire :

Le caractère $C''_{P1}$ correspondra à la liste $e''_1$ dans la nouvelle partition.

Le caractère $C''_{P2}$ correspondra à la liste $e''_2$ dans la nouvelle partition.

…

Le caractère $C''_{Pm}$ correspond à la liste $e''_m$ dans la nouvelle partition.

Ainsi, nous obtenons la distribution ($C'_i$., $e'_i$ ).

**Étape2 :**

En premier lieu, nous appliquons la clé de mutation sur les listes $e'_i$ qu'on vient de trouver afin de retomber sur l'ordre des listes que nous avons eu avant le passage à la nouvelle partition et que nous avons désigné par $e_{ji}$.

Après, nous les déposant l'une à côté de l'autre suivant la permutation donnée par la clé de Mutation.

Ensuite, à partir des cardinaux des listes initiales stocker dans la clé nous retrouvons la distribution de la partition initiale L'i

Par la suite, et afin d'avoir les vecteurs initiaux représentant M, nous appliquons encore une autre fois le déchiffrement de SEC en utilisant l'individu utilisé lors de la création de la nouvelle partition et qui fait partie aussi de la clé.

Ainsi nous retrouvons le message clair.

## 11. Expériences & Résultats

Les expériences menées sur ce système ont été effectuées sur plusieurs textes de différentes tailles afin de pouvoir engendrer tous les cas possibles et ressortir des résultats plus efficaces et réalistes. Les expériences ont considéré aussi la langue des textes, à savoir : le Français et l'Anglais. En premier temps, nous nous sommes intéressés à déterminer la configuration la plus adéquate pour notre nouveau système en se focalisant sur les points suivants :

- ✓ La taille de la population
- ✓ Le taux d'application des opérateurs génétiques
- ✓ La taille de la clé

Ensuite, nous avons passé à l'étude analytique du chiffré afin de calculer la résistance du système face au différentes attaques possibles dans ce cadre.

Ci-dessous deux exemples qui illustrent l'application de notre système, une fois en utilisant le texte en anglais et l'autre fois pour un texte français :

## 12. Exemple

### a) Modèle message clair en Anglais

The history of cryptography began thousands of years ago. Until recent decades, it has been the story of what might be called classic cryptography — that is, of methods of encryption that use pen and paper, or perhaps simple mechanical aids. In the early 20th century, the invention of complex mechanical and electromechanical machines, such as the Enigma rotor machine, provided more sophisticated and efficient means of encryption; and the subsequent introduction of electronics and computing has allowed elaborate schemes of still greater complexity, most of which are entirely unsuited to pen and paper .The development of cryptography has been paralleled by the development of cryptanalysis — the "breaking" of codes and ciphers. The discovery and application, early on, of frequency analysis to the reading of encrypted communications has, on occasion, altered the course of history. Thus the Zimmermann Telegram triggered the United States' entry into World War I; and Allied reading of Nazi Germany's ciphers shortened World War II, in some evaluations by as much as two years. Until the 1970s, secure cryptography was largely the preserve of governments. Two events have since brought it squarely into the public domain: the creation of a public encryption standard (DES), and the invention of public-key cryptography.

**Figure 24 : Message clair en anglais**

### b) La Clé Symétrique

**Elected child:** 0 26 51 35 17 45 29 3 14 5 48 23 22 31 37 42 2 41 1 39 24 16 47 7 33 30 10 50 13 27 36 4 44 19 21 38 12 15 9 28 46 49 20 11 40 25 6 32 8 18 34 43

**Cardinals of the elected child:** 0 7 14 22 33 40 48 46 38 32 21 13 6 1 8 16 26 34 41 50 45 37 31 20 12 5 2 10 17 28 35 23 42 51 44 49 47 9 15 24 27 36 30 39 19 11 4 3 25 18 29 43

**Mutation_Ch:** 6 1 1 14 62 2 1 22 23 2 4 40 13 1 2 79 3 1 11 129 2 1 29 203 2 1 8 70 2 1 87 20 1 1 36 40 1 2 66 6 1 4 55 2 1 42 85 3 1 80 54 2

**Final_Ch:** 0 27 49 22 5 32 44 17 10 37 39 12 15 42 34 7 20 47 29 2 25 51 24 3 30 46 19 8 35 41 14 13 40 36 9 18 45 31 4 23 50 26 1 28 48 21 6 33 43 16 11 38

**Figure 25 : La clé symétrique pour le texte anglais**

### c) Message chiffré


**Figure 26 : le chiffré du Message Anglais**

### d) Modèle du message clair en Français

*Les mots de passe constituent une technique d'authentification unidirectionnelle très répandue : un utilisateur fournit son identité et un mot de passe pour accéder à une ressource. Un mot de passe constitue donc un secret partagé entre l'utilisateur et le système auprès duquel il s'authentifie : prouver qu'il connaît ce secret donne l'assurance que son identité est correcte. La faiblesse principale de cette technique provient justement de ce que les mots de passe peuvent facilement être dévoilés ou découverts. Les systèmes d'authentification par mot de passe sont sujet à plusieurs types d'attaques, en particulier la recherche exhaustive de mots de passe à partir de leur texte chiffré et le rejeu de mots de passe.Le stockage des mots de passe sous forme de fichiers chiffrés constitue une précaution élémentaire mais autorise cependant les attaques exhaustives. Il est par exemple possible d'essayer exhaustivement tous les mots de passe possibles et pour chacun, de comparer son chiffrement à la valeur du chiffrement du mot de passe de l'utilisateur stockée dans le fichier. Le salage de mots de passe [MT 79], par exemple utilisé dans Unix, améliore la sécurité de ce schéma : il consiste à ajouter à chaque mot de passe, lorsqu'il est introduit initialement, une chaîne de t bits aléatoires, appelée "sel", avant d'appliquer une fonction de hachage à sens unique. Le mot de passe haché et le "sel" sont enregistrés dans le fichier de mots de passe. Le salage d'un mot de passe n'augmente pas la difficulté de recherche par une attaque exhaustive sur un seul mot de passe puisque le "sel" est conservé en clair dans le fichier de mots de passe ; par contre, elle élève le coût d'une attaque simultanée sur plusieurs mots de passe, puisqu'il faut prévoir 2t variations de chaque mot de passe.*

**Figure 27 : Texte Modèle en Français**

### e) La clé symétrique

```
Fils elu :
0 29 35 6 23 41 12 17 46 18 11 40 24 5 34 30 1 28 36 7 22 42
13 16 45 19 10 39 25 4 33 31 2 27 37 8 21 43 14 15 44 20 9 38
26 3 32

Cardinaux du fils elu :
7 2 2 127 7 1 91 23 1 76 80 1 9 66 8 4 235 15 1 62 3 1 89 15 1
63 59 1 2 37 10 4 147 4 1 54 28 6 30 17 1 5 103 1 7 296 1

Clé de Mutation :
0 1 5 7 11 12 13 15 16 17 19 20 21 23 25 26 27 28 29 30 31 32
33 34 35 36 37 38 39 40 41 42 2 8 43 44 14 45 24 22 3 46 6 9
18 4 10

Clé génétique :
0 2 35 6 8 41 3 14 46 9 11 40 24 5 34 30 1 28 36 7 22 42 13 16
45 19 10 39 25 4 33 31 27 15 37 32 44 43 18 20 21 23 26 38 29
12 17
```

**Figure 28 : La clé symétrique pour le texte français**

### f) Le message chiffré

```
Lyeu7àdeubyunfeeyu[àledmd,yldu,lyudy[qlmp,yubxf,dqyldmcm[fdmàlu,lmbm.y
[dmàllyIIyud.:eu. nflb,yuju,lu,dmImefdy,.ucà,.lmdueàlumbyldmd
uydu,lu7àdubyunfeeyunà,.uf[[ by.u2u,lyu.yeeà,.[yéuTlu7àdubyunfeeyu
[àledmd,yubàl[u,luey[.ydunf.dft
uyld.yuIx,dmImefdy,.uyduIyuesed:7yuf,n.:eub,p,yIumIuexf,dqyldmcmyujun.à,vy.up
,xmIu[àllfrdu[yuey[.ydubàllyuIxfee,.fl[yup,yueàlumbyldmd uyedu[à..y
[dyéuLfucfmgIyeeyun.ml[mnfIyubyu[yddyudy[qlmp,yun.àvmyldu],edy7yldubyu
[;up,;uI;eu7àdeub;unfee;un;,v;lducf[mI;7;ld9Ud.;9b vàmI e9à,9b
[à,v;.deé9L;e9esed:7;e9bxf,dq;ldmcm
[fdmàl9nf.97àd9b;9nfee;9eàld9e,];d929nI,em;,.e9dsn;e9bxfddfp,;eU9;l9nf.dm
[,Im;.9If9.;[q;.[q;9;oqf,edmv;9b;97àde9b;9nfee;929nf.dm.9b;9I;,.9d;od;9[qmcc.
9;d9I;9.;];,9b;97àîe9b;9nfee;éL;9eîà["ft;9b;e97àîe9b;9nfee;9eà,e9cà.7;9b;9cm
[qm;.e9[qmcc. e9[àleîmî,;9,l;9n. [faîmàl9 I 7;lîfm.;97fme9faîà.Me;9
[;n;lbflî9I;e9fîîfpa;e9;oqfaeîMvèeé9iI9èeî9nf.9èoè7nIè9nàeeMgIè9bxèêêfsè.9èoq
faêîMvè7èlî9îàaê9Ièê97àîê9bè9nfêêè9nàêêMgIèê9èî9nàa.û[qf[alUûbèû
[à7nf.è.ûêàlû[qMcc.è7èlîû2ûIfûvfIèa.ûbaû
[qMcc.è7èlîûbaû7àîûbèûnhêêèûbèûIxaîMIMêhîèa.ûêîà[" èûbhlêûIèûcM
[qMè.éûLèûêhIhtèûbèû7àîêûbèûnhêêèûiiiûiiiUûnh.ûèoè7nIèûaîMIMê ûbhlêûTlMoUûh7
IMà.èûIhûê [a.Mî ûbèû[èûê[q 7hûjûMIû[àlêMêîèû2ûh]àaîè.û2û
[qhpaèû7àîûbèûnhêêèUûIà.êpaxMIûêêîûMlî.àbaMîûMlMîMhIè7èlîUûalèû
[qhrlèûbèûîûgMîêûhI hîàM.èêUûhnnèI èûiêèIiUûhvhlîûbxhnnIMpaè.ûalèûcàl
[îMàlûbèûqh[qhtkû2ûêklêûalMpakéûLkû7àîûbkûnhêêk'qh[q
'kî'Ik'iêkIi'êàlî'kl.ktMêî. ê'bhlê'Ik'cM
[qMk.'bk'7àîê'bk'nhêêké'Lk'êhIhtk'bxal'7àî'bk'nhêêk'lxhat7klîk'nhê'Ih'bMccM
[aIî 'bk'.k[qk.
[qk'nh.'alk'hîîhpak'koqhaêîMvk'êa.'al'êkaI'7àî'bk'nhêêk'naMêpak'Ik'iêkIi'kêî'
[àlêk.v 'kl'[IhM.'bhlê'Ik'cM[qMk.'bk'7àîê'bk'nhêêk'i'nh.'[àlî.kU'kIIk'
I:vk'Ik'[àiî'bxalk'hîîhpak'êM7aIîhl
k'êa.'nIaêMka.ê'7àîê'bk'nhêêkU'naMêpaxMI'chaî'n.
vàM.'iî'vh.MhîMàlê'bk'[qhpak'7àî'bk'nhêêké
```

**Figure 29 : le chiffré du texte français**

## 13. Paramétrage du Système

### a) Taille de la Population :

D'après les résultats mentionnés dans le tableau ci-dessous La meilleure valeur de l'optimum a été atteinte dans les populations de tailles 16 et 24. Cependant nous remarquons que plus la taille de la population est grande plus le nombre des itérations augmente et par conséquent le temps de la convergence du système devient plus important. Nous nous contentons alors, de rester sur une population de taille 16 pour une meilleure solution et dans un temps raisonnable.

| | | Taille de la Population | | | |
|---|---|---|---|---|---|
| | | **16** | **24** | **30** | **32** |
| 1er Message (1327 caractères) | Nombre des itérations | 38 | 50 | 50 | 50 |
| | Valeur de convergence | 860 | 860 | 860 | 860 |
| | Temps d'exécution (ms) | 110 | 321 | 318 | 345 |
| 2ème Message (1615 caractères) | Nombre des itérations | 42 | 50 | 50 | 50 |
| | Valeur de convergence | 1007 | 1007 | 1007 | 1007 |
| | Temps d'exécution (ms) | 280 | 389 | 436 | 500 |
| 3ème Message (2679 caractères) | Nombre des itérations | 46 | 50 | 50 | 50 |
| | Valeur de convergence | 1809 | 1809 | 1809 | 1809 |
| | Temps d'exécution (ms) | 858 | 1046 | 1065 | 1201 |
| 4ème Message (3856 caractères) | Nombre des itérations | 52 | 50 | 52 | 52 |
| | Valeur de convergence | 1189 | 1178 | 1168 | 1178 |
| | Temps d'exécution (ms) | 1769 | 1740 | 1772 | 1769 |

**Figure 30 : Taille de la population**

### b) Taille de la clé

La taille de la clé est calculée en fonction des différents composants définissant cette clé.

Le tableau suivant, montre la relation entre la taille de la clé en fonction du nombre des caractères et de la taille du texte à chiffrer :

| longueur du message | Nombre de caractères | Taille de la clé |
|---|---|---|
| 539 | **38** | **1216** |
| 1327 | **53** | **1696** |

| | | |
|---|---|---|
| 1615 | **53** | **1696** |
| 2679 | **50** | **1600** |
| 4458 | **56** | **1792** |

**Figure 31 : Relation entre la taille de la clé, le nombre des caractères et la taille du texte à chiffrer**

Nous pouvons constater que la taille de la clé n'est pas liée forcément à la taille du texte mais plutôt au nombre des caractères utilisés dans ces textes. Car un texte de petite taille peut contenir le même nombre des caractères que celui d'une grande taille. D'où l'intérêt de l'utilisation de notre système sur les grands textes plutôt que sur les petits.

De plus, la taille de la clé est très importante et reste dans les normes définies pour les systèmes symétriques. Chose qui lui rend plus résistant face aux attaques exhaustives possibles sur la clé.

**c) Taux des opérateurs génétiques :**

Pendant chaque itération, nous avons recourt à appliquer l'opérateur génétique de croisement avec un taux de 60% à 100% des individus afin de générer la nouvelle population. Cependant, la mutation est un phénomène rare qui ne doit être appliqué que rarement et après plusieurs générations. Et c'est bien le cas dans notre système, nous convenons d'appliquer cet opérateur qu'une seule fois dans toutes les itérations réalisées et de préférence au cours de la dernière itération.

Le taux de l'application de l'opérateur de la Mutation est aussi faible car ça engendre une augmentation en double de la taille de la clé. De plus, appliquer la mutation une seule fois est très suffisant pour garantir un meilleur résultat en intégrant la meilleure partition.

## 14. Étude Analytique du chiffré

Étant donné que notre objectif principal de cette nouvelle conception du système SEC, est d'avoir un chiffré aléatoire dont tous les caractères ont la même fréquence d'apparition, une étude dans ce sens est très importante afin de prouver notre approche sur le plan expérimental aussi.

L'expérience consiste à comparer pour chaque caractère la liste de ses positions dans le texte clair et les nouvelles listes de ce même caractère dans le chiffré réalisé avec

notre nouveau système. Cette expérience a pour objectif de mettre en évidence la différence entre les deux listes de positions d'un caractère qui peut être démontré par étude de sa fréquence d'apparition dans le texte avant et après le chiffrement. Nous avons fait cette étude sur plusieurs textes et les résultats étaient toujours similaires à ceux présentés dans le graphe comparatif ci-dessous :

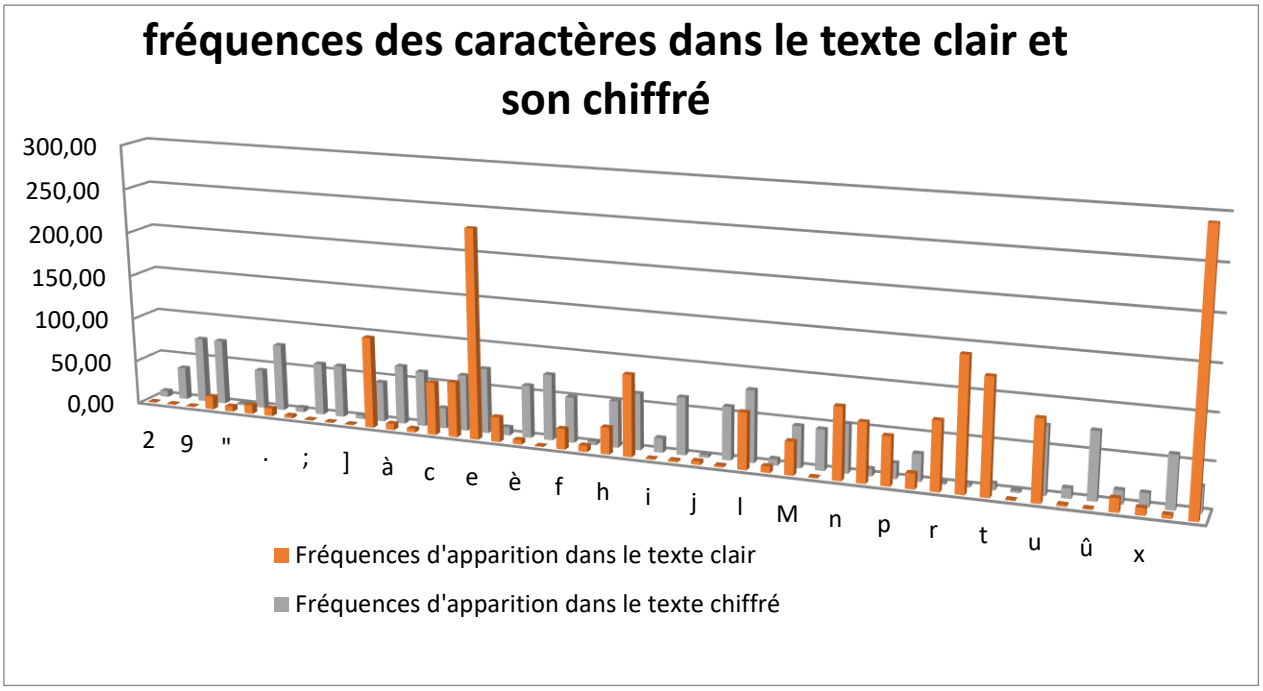



**Figure 32 : Étude statistique des fréquences d'apparition des caractères avant et après le chiffrement**

**Interprétation :**

Comme prévu, cette représentation graphique montre que les fréquences d'apparition des caractères dans le texte clair sont tout à fait différentes de celles du texte chiffré. Ce qui permet de montrer que la permutation et la partition appliquées dans le système ont bien perturbé la répartition des caractères dans le texte clair. Cependant, ce résultat ne nous permettrait pas de conclure qu'on a pu trouver une solution d'un chiffrement aléatoire car il faut, de plus, que la plupart des caractères auront la même fréquence d'apparition chose qu'on ne voit pas clair dans cette expérience.

Notre deuxième expérience, consiste alors à comparer l'intervalle de variation des fréquences d'apparition avant et après notre chiffrement. Les résultats de cette expérience sont donnés dans les deux graphes qui suivent.

Le premier graphe représente la variation des fréquences d'apparition des caractères dans le texte clair, tandis que le deuxième représente cette variation pour les mêmes caractères bien sûr, mais cette fois-ci dans le texte chiffré:

**a) Variation des fréquences d'apparition des caractères dans le texte clair**

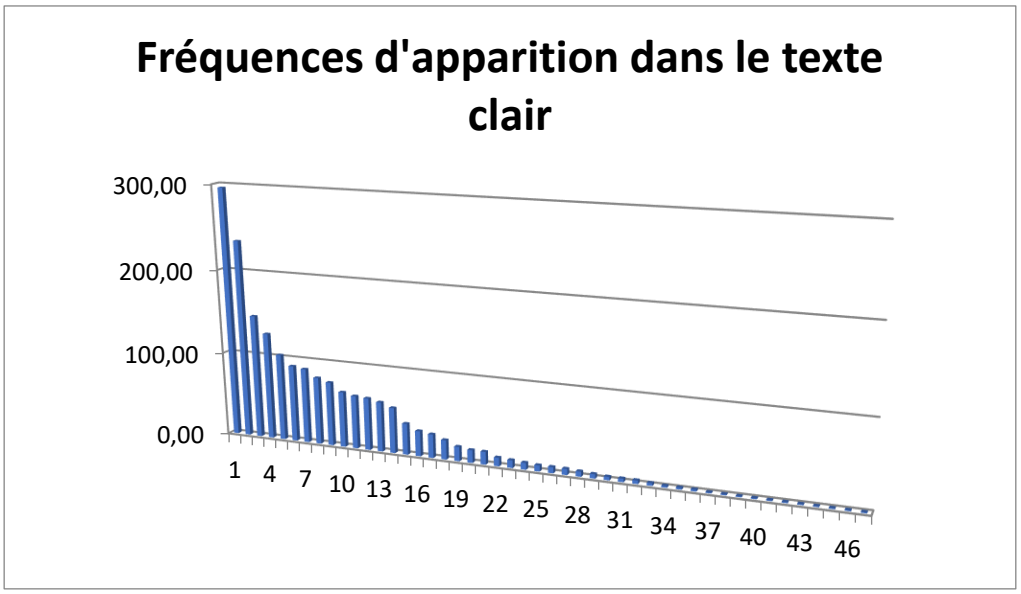



**Figure 33 : Fréquences d'apparition des caractères dans le texte clair**

**b) Variation des fréquences d'apparition des caractères dans le texte chiffré**

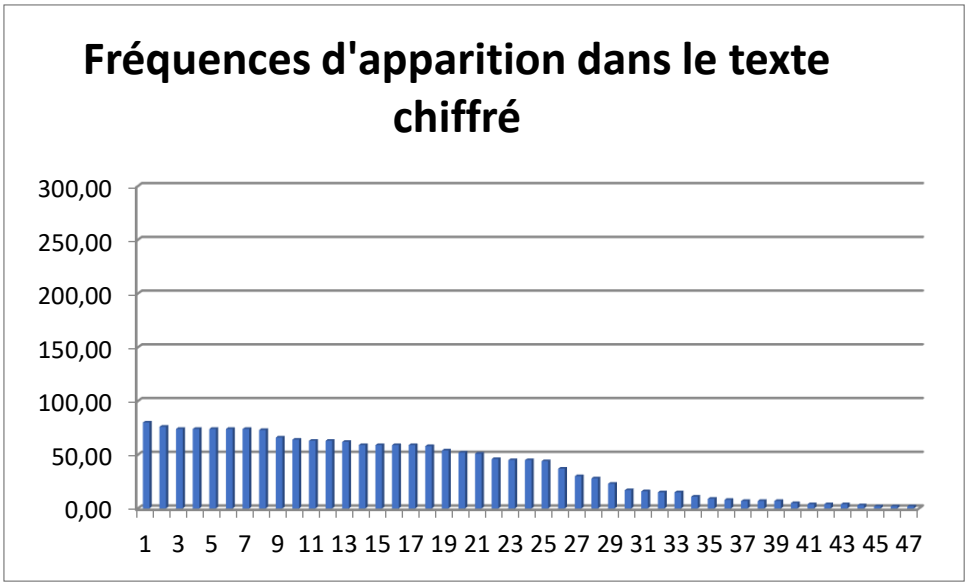



**Figure 34 : Fréquences d'apparition des caractères dans le texte chiffré**

**c) Interprétation des résultats**

A partir des résultats présentés dans les deux graphes ci-dessus, nous constatons que la courbe des fréquences d'apparition a complètement changé. La variation de ces fréquences prend actuellement à peu près les mêmes valeurs et ne dépasse pas '73 occurrences' alors qu'ils étaient entre 1 et 296, sachant que nous avons 47 listes pour un message de taille de presque 2000 positions. Ce qui donne une moyenne de 43 positions par liste.

Dans le deuxième graphe, plus que 50% des caractères ont actuellement des listes avec une taille moyenne et 36% avec une valeur plus petite mais proche de cette moyenne, tandis que le reste, ont des petites tailles mais ils représentent un pourcentage faible ne permettant pas de réaliser une étude statistique dans ce sens.

Ces résultats montrent alors que les nouvelles techniques adoptées par notre système nous ramènent à avoir un chiffré aléatoire loin d'être sujet d'une cryptanalyse par fréquence d'apparition

## 3. Discussion

A partir des différents textes traités dans nos expériences, nous pouvons conclure que nous avons pu élaborer un système de chiffrement complet, bien étudié et très efficace.

La robustesse de ce système réside dans le fait que nous avons pu intégrer deux problèmes difficiles dans le processus de chiffrement :

Le problème des Partitions

Le problème des permutations

Ainsi, nous sommes restés sur le même principe de SEC en se basant sur les AG qui offrent un cadre tout à fait aléatoire, chose qui est très intéressante dans le domaine de la cryptographie.

De plus, la conception de la nouvelle fonction d'évaluation a joué un rôle très important pour trouver les solutions qui respectent bien les caractéristiques tracées dans l'algorithme.

# CHAPITRE 7

# NOUVEAU SYSTÈME DE CHIFFREMENT HYBRIDE H-ASEC ET ÉTUDE DE SA SÉCURITÉ

## Sommaire

## 1. Introduction

Les systèmes de chiffrement symétriques, inventés bien avant les systèmes de chiffrement asymétriques, restent le type de crypto-système le plus utilisé dans les applications et les systèmes d'information.

L'utilisation généralisée des systèmes de chiffrement symétriques est principalement due à leur simplicité, leur vitesse, leur capacité de traiter plus de quantité de données et finalement sécurité par rapport aux systèmes de chiffrement asymétriques. Dès lors, la principale difficulté réside dans le partage et l'accord sur les clés pour permettre aux entités concernées par cette communication de partager le même secret initial sans qu'aucun attaquant potentiel ne l'intercepte **[70]**. La remise de la clé secrète doit bénéficier de tous les moyens de protection possibles pour assurer l'authentification, l'intégrité et la confidentialité de toutes les informations échangées.

Whitfield Diffie et Martin Hellman ont pu mettre fin à ce problème et éviter l'écueil des systèmes symétriques en utilisant un nouveau mécanisme basé sur deux clés, une publique et une privée. L'émergence des crypto-systèmes asymétriques, ou crypto-systèmes à clé publique, apporte une réponse indubitable au problème d'échange des clés. La robustesse de ce type d'algorithmes repose sur la difficulté et la complexité de résolution de certains problèmes mathématiques. Cependant, ces algorithmes manquent de rapidité et sont pratiquement inutilisables surtout pour un échange en ligne avec de gros volumes de données. Cependant, dans le processus de chiffrement, ce type d'algorithme est utilisé plus dans les crypto-systèmes hybrides **[71]**. Ces derniers représentent une nouvelle approche qui consiste en une combinaison d'algorithmes symétriques et asymétriques afin de profiter des avantages de chacun d'entre eux et les rendre complémentaires.

Dans cette contribution, nous nous sommes inspirés de l'approche des crypto-systèmes hybrides pour développer un algorithme de chiffrement symétrique évolutif appelé «Hybrid- Advanced Symetrical Evolutionary Ciphering » (H-ASEC). Comme son nom l'indique, ce nouveau système est une hybridation de notre système de chiffrement ASEC et le système de chiffrement asymétrique RSA afin de remédier aux problèmes de partage de la clé de session générée par ASEC. **[72]**

## 2. Travaux connexes

Philip Zimmermann a été le premier à introduire des systèmes cryptographiques hybrides. Il a réussi à combiner le système de cryptage symétrique IDEA avec le système de cryptage asymétrique RSA. Son travail a donné naissance au crypto-système Pretty Good Privacy (PGP). PGP a été le premier système de chiffrement hybride créé. Depuis, PGP a intégré d'autres concepts cryptographiques pour couvrir non seulement la confidentialité des données mais aussi les différentes exigences de sécurité pour l'échange, le stockage et la divulgation de données à usage privé (signature, compression, etc.) **[73]**.

## 3. Contexte

### 3.1 Description de l'ASEC

Le système ASEC a été présenté et détaillé dans le chapitre 6 intitulé : *Nouvelle Fonction d'évaluation pour le système de chiffrement SEC intégrant le problème des Partitions*.

## 4. Crypto-système hybride évolutionnaire

*Problématique :*

ASEC peut être considéré comme un système de chiffrement symétrique car il utilise la même clé pour le chiffrement et le déchiffrement. La seule différence est que notre système ressemble à un mécanisme de chiffrement des masques jetables. En effet, la clé de chiffrement n'est pas changée une seule fois mais elle change d'une exécution à l'autre. Elle est alors considérée comme une clé de session. Le problème qui se pose dans ce cas, est que cette clé

doit absolument être sécurisée chaque fois que l'on souhaite établir une communication utilisant ce système de chiffrement.

*Solution :*

Pour résoudre ce problème, nous proposons d'utiliser le principe des crypto-systèmes hybrides à l'exception de l'étape de génération des clés de session. Dans notre cas, les clés sont générées implicitement par notre système de chiffrement.

Principe :

Le principe est simple et peut être illustré par le schéma suivant :

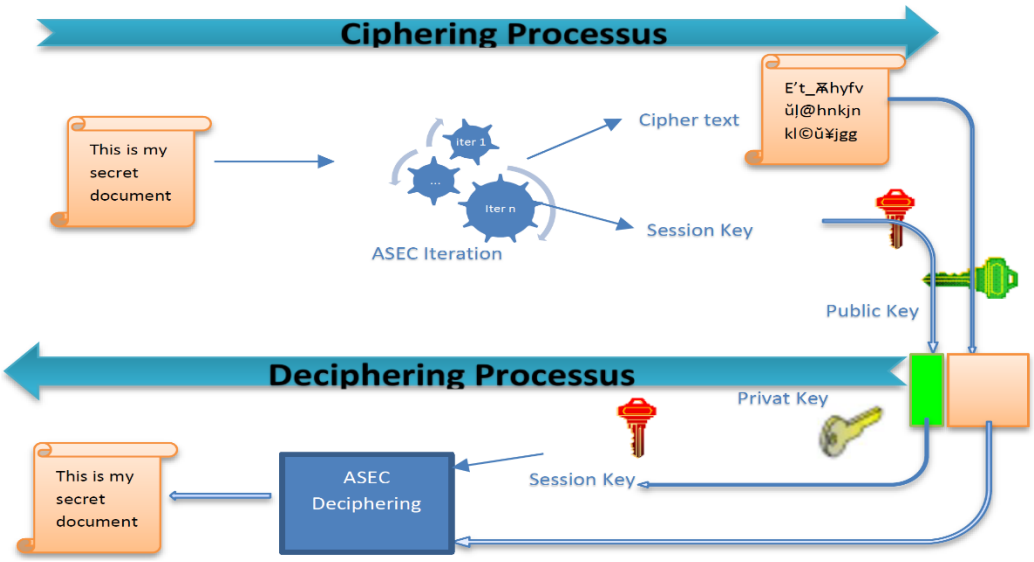



**Figure 35 : Principe du crypto-système hybride évolutif**

Comme présenté dans Figure 35, notre nouveau système H-ASEC fait appel au chiffrement ASEC pour réaliser et le chiffrement du texte clair et la génération de la clé de session correspondante. Cette dernière est ensuite chiffrée par le biais de la clé public RSA pour la joindre avec le chiffré dans un seul package et les envoyer au destinataire.

Le processus de déchiffrement est appliqué par le destinataire une fois recevoir le package en question. Il commence par déchiffrer la clé de session de ASEC par le biais de la clé privée de RSA puis il passe au déchiffrement du chiffré en utilisant sa clé de session.

*Algorithme H-ASEC:*

Algorithme

Input: **The plaintext M**

Output: **The Ciphertext C; Clé de chiffrement crypté.**

Begin

Step 0 : Codage du problème

$M(C_1,C_2,...,C_m)$--> $M((C_1,L_1)(C_2,L_2)...(C_m,L_m))$

Step 1 : Initialisation ==> Création de la population initiale

for j=1, j<= q

$X_j$ → **Permut**$\{L_1,L_2,...,L_m\}$

Fin for

$P_0$→$X_1,X_2,...,X_q$ (population initiale)

i=0

While ( !condition_arret)

Step 2 : //Evaluation

For each $X_j$ from $P_i$

Ej → Create_Partition($X_j$=$\{L_{j1},L_{j2},...,L_{jm}\}$)

Val[j] = Evaluate (Ej)

Fin for

Step 3 : Selection_roulette(Val[])

Step 3 : $P_{i+1}$→Croisement_MPX (Pi)

Step 4 : if is_mutation then

Apply_Partition(Eo) // Eo est la partition optimale

$X'_j$→ Change_Lists($X_j$ )

Key→ (Xo , Card(Xo), Eo)

Fin While

Step 5 : Chiffrement(M, Key)

Step 6 : Chiffrement (Key, RSA)

End

## 5. Étude de sécurité

En général, la sécurité de la nouvelle approche H-ASEC repose principalement sur la sécurité du chiffrement ASEC étant donné que RSA est déjà prouvé robuste dans le contexte du partage des clés. Donc, la première question qui se pose dans cette étude est : quel est l'avantage d'utiliser ASEC au lieu des autres systèmes symétriques utilisés dans le crypto-système PGP ?

Pour répondre à cette question, nous devons préciser les critères de sélection de ces systèmes symétriques et voir comment notre système peut répondre à ces critères.

En faisant, une étude de marché, nous constatons que le choix des algorithmes de chiffrement ne dépend pas seulement à son « niveau de sécurité » que nous les chercheurs essaient d'augmenter au maximum, mais aussi à d'autres critères, à savoir :

- Temps : Rapide ou lent.
- Matériel : La capacité de l'équipement utilisé pour chiffrer et déchiffrer les messages.
- Paramétrage : La taille de la clé et des blocs, permettant d'augmenter la force de l'algorithme contre les attaques par force brute.
- Puissance : Capacité à résister aux différentes attaques possibles selon la configuration de cet algorithme dans le système PGP.
- Réputation : Elle est liée principalement à l'ancienneté de l'algorithme dans le domaine.
- Breveté.

En fait, dans cette étude, nous ne pouvons pas compter sur tous ces critères pour démontrer l'efficacité de notre système car évidemment l'ASEC est nouveau et n'a pas encore été expérimenté hors laboratoire pour parler de sa réputation et de son accessibilité. Cependant, nous allons nous concentrer sur l'étude de sécurité de ce système car il représente l'un des facteurs les plus intéressants dans cette phase et ensuite nous ferons une étude comparative basée sur les autres critères tels que le temps d'exécution, le paramétrage et le matériel.

### 5.1 Paramétrage du système et attaque par force brute

Le paramétrage des systèmes de chiffrement symétriques est un prérequis indispensable à sa sécurité. En général, ce paramétrage comprend la taille de la clé et la taille du bloc nécessaires pour assurer la résistance du système vise à vis aux attaques par force brute.

Deux facteurs majeurs augmentent la résistance contre ce type d'attaque :

- La taille de la clé, qui doit être aussi grande que possible.
- La séquence représentative de la clé qui doit être indiscernable.

*Longueur de la clé :*

La taille de cette clé dépend du nombre de caractères différents du texte en clair à chiffrer en utilisant la relation suivante : **(8*n)*4**, **n** étant le nombre de caractères différents du texte en clair.

Une étude expérimentale est réalisée sur plusieurs textes clairs de différentes tailles et provenant de différentes sources. Le tableau et le graphique ci-dessous montrent la progression de cette clé en fonction de la taille du texte à chiffrer.

| Size of the message (characters) | Size of the message (bits) | Different characters | the key size |
|---|---|---|---|
| 642 | 5136 | 40 | 1280 |
| 864 | 6912 | 31 | 992 |
| 1204 | 9632 | 52 | 1664 |
| 1516 | 12128 | 41 | 1312 |
| 2893 | 23144 | 55 | 1760 |
| 4543 | 36344 | 66 | 2112 |
| 5514 | 44112 | 72 | 2304 |
| 6097 | 48776 | 80 | 2560 |
| 6181 | 49448 | 110 | 3520 |
| 5514 | 44112 | 72 | 2304 |
| 9162 | 73296 | 82 | 2624 |
| 14250 | 114000 | 83 | 2656 |
| 20531 | 164248 | 85 | 2720 |
| 23396 | 187168 | 100 | 3200 |
| 24280 | 194240 | 91 | 2912 |

**Table 10**: **Dépendance entre la clé et la taille du texte**

Pour mieux comprendre ce tableau, ci-dessous deux graphes représentatifs des résultats trouvés :

- Le premier graphe (figure 36) montre le pourcentage de la taille de la clé par rapport à la taille du texte clair et dans lequel on veut comprendre comment ce pourcentage varie dans les grands et les petits textes.
- Le deuxième (figure 37) montre l'évolution de taille de la clé en fonction de la taille du texte, et là aussi on veut mettre le point sur la convergence de cette taille vers une valeur fixe pour les grands textes.

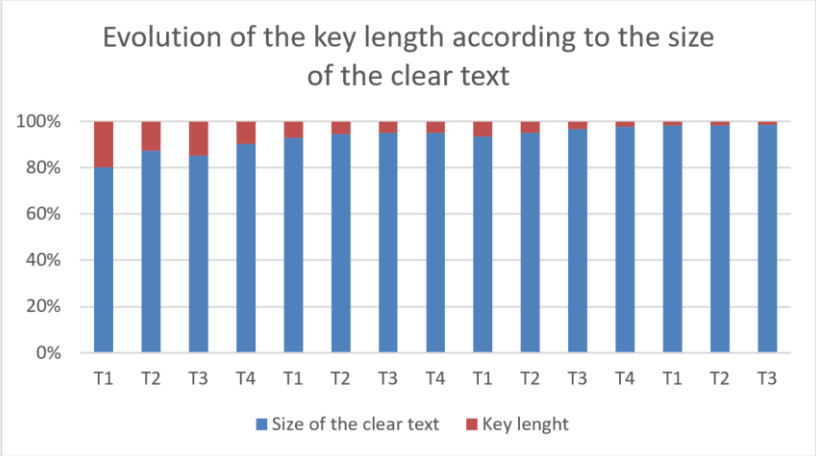



Figure 36 : Dépendance entre la clé et la taille du texte

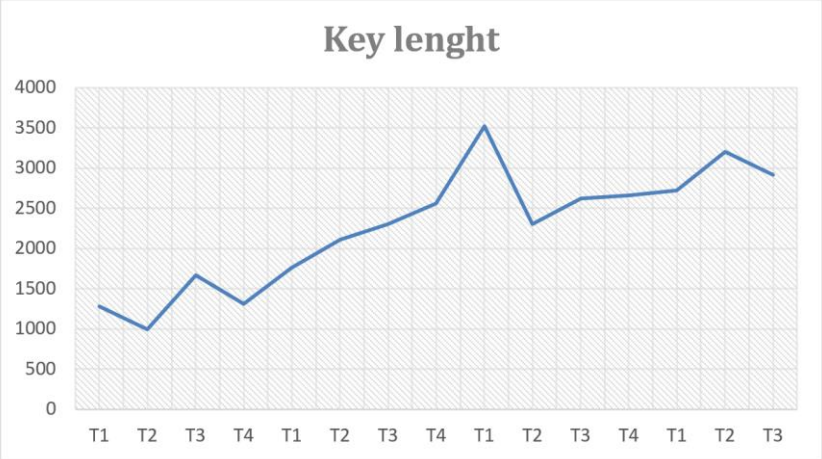



Figure 37: l'évolution de la clé par contribution au texte clair

*Discussion :*

Suite à cette étude expérimentale, on constate que l'augmentation de la taille de la clé est très faible (voir constante pour les plus gros textes) par rapport à la taille du message à chiffrer. Cela a du sens car avoir un texte plus gros n'implique pas nécessairement qu'il contienne plus de caractères qu'un petit texte. Par conséquence, la taille moyenne de la clé dans notre cas est de 2231 bits. Cependant, en théorie, la taille maximale pouvant être atteinte est de 8192 bits si l'on considère que le texte à chiffrer contient les 256 caractères possibles.

On peut alors dire que la taille de la clé ASEC assure la résistance aux attaques par force brute et offre une sécurité à long terme, même avec l'utilisation des machines quantiques.

*Génération de clé :*

La sécurité de ***ASEC*** n'est pas seulement liée à la taille de sa clé mais à d'autres points forts qui résident principalement dans la manière dont elle est générée, à savoir :

- Il n'utilise aucun système de génération de clé.
- La clé est générée automatiquement par le système.
- La clé est construite grâce à un algorithme non déterministe.
- Il utilise plusieurs mécanismes probabilistes qui reposent sur des choix aléatoires pour déterminer la solution optimale.
- Sa taille est variante.

*Clé de session :*

Chaque texte en clair chiffré par le système ***ASEC*** possède une et une seule clé qui dépend de sa structure, sa taille et sa nature. Une modification de l'un de ces critères donne naissance à une nouvelle clé. En conséquence, le même texte en clair peut conduire à deux textes chiffrés différents et ceci est obtenu en modifiant la population initiale en fonction de l'algorithme évolutif.

Avoir une clé de session dans notre système permet d'étendre l'authentification sur le support de communication et d'empêcher les différentes attaques cherchant à connaître la clé. En effet, trouver la clé ne sera pas très utile car elle ne sera utilisée que dans la transaction en cours.

## 5.2 Performances de l'algorithme

*Complexité* **:**

Le principe de l'algorithme cryptographique évolutif est basé sur l'idée de créer des partitions équiprobables dont la taille est presque la même. Cela introduit le problème de partition qui est un problème difficile à résoudre. Normalement, ce type de conception est utilisé dans les chiffrements asymétriques. Il augmente la complexité de la résolution de ce chiffrement de manière exponentielle. Cela rend le crypto-système beaucoup plus résistant aux différents types d'attaques.

*Test d'avalanche* :

Pour tester le caractère aléatoire du résultat du chiffrement ASEC, nous appliquons une distance de Hamming entre les messages d'entrée et de sortie. Pour chaque message Mi, nous exécutons l'algorithme de cryptage ASEC avec différents bits de clé modifiés. Ensuite, nous calculons la moyenne de la valeur de distance de Hamming entre le message et son chiffré.

En fait, la distance de Hamming du chiffrement obtenu devrait être la moitié de la taille de sortie.

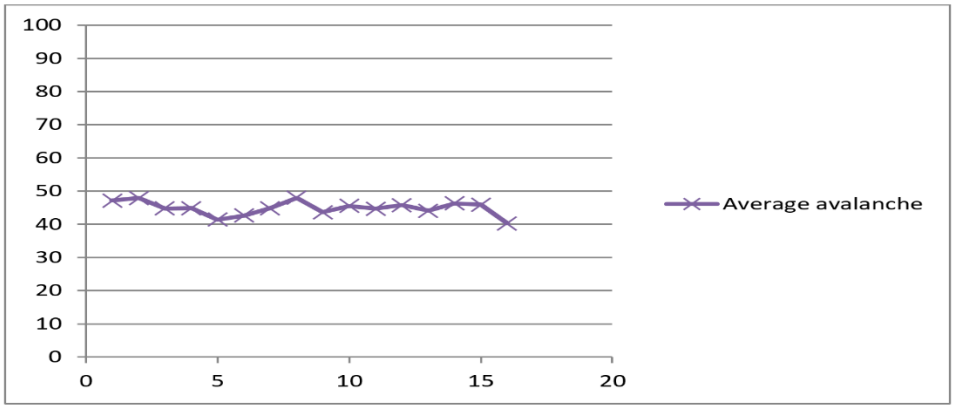



**Figure 38 : Distance de Hamming obtenue pour les différents textes chiffrés pour chaque message**

Nous concluons que la distance de Hamming du chiffre obtenu converge vers la moitié de la taille moyenne du texte en sortie (chiffré) ***(Hamming(H(x),H(y)) ≈ n/2***

Ce résultat prouve que le chiffrement est aléatoire.

***Test statique : Diehard***

Afin de contourner les attaques statistiques, nous avons appliqué divers tests statistiques inclus dans le package DIEHARD. Cette plateforme propose tous les outils de vérification des différents tests statiques pour démontrer la force et l'efficacité du chiffrement de séquences de bits généré par notre système. Il vérifie également le caractère aléatoire jusqu' un niveau extrême.

Le tableau ci-dessous montre le résultat donné par l'exécution des tests DIEHARD sur le fichier qui contient tous les messages de chiffrement ASEC :

| *Test* | *P-value* | *Interpretation* |
|---|---|---|
| ***Diehard birthdays*** | 0.50646335 | PASSED |
| ***Diehard operm5*** | 0.50241355 | PASSED |

| | | |
|---|---|---|
| ***Diehard_rank_32x32*** | 0.65910598 | PASSED |
| ***Diehard_rank_6x8*** | 0.73626155 | PASSED |
| ***Diehard_bitstream*** | 0.75444424 | PASSED |
| ***Diehard_opso*** | 0.84942117 | PASSED |
| ***Diehard_opso*** | 0.59952027 | PASSED |
| ***Diehard_dna*** | 0.09103884 | PASSED |
| ***Diehard_count_1s_str*** | 0.94765782 | PASSED |
| ***Diehard count1s byt*** | 0.77998540 | PASSED |
| ***Diehard parking lot*** | 0.69671967 | PASSED |
| ***Diehard 2d sphere*** | 0.03413893 | PASSED |
| ***Diehard 3d sphere*** | 0.09723242 | PASSED |
| ***Diehard squeeze*** | 0.28015448 | PASSED |

**TABLE 11: EXECUTION DES TESTS DE DIEHARD**

# CONCLUSION

À travers ces travaux, nous avons pu réaliser une nouvelle version du système de chiffrement évolutionniste SEC initié par notre Professeur Fouzia Omary. L'originalité de ce système réside dans le fait de ramener le problème de chiffrement à un problème d'optimisation combinatoire et faire appel aux algorithmes évolutionnistes pour le résoudre. En fait, c'est pour la première fois que les algorithmes évolutionnistes ont été utilisés en cryptographie. Plusieurs versions d'améliorations ont vu le jour par le concepteur de ce système cryptographique évolutionniste. Nos contributions dans ce contexte ont apporté de nouvelles améliorations par la mise en place de nouvelles techniques et formulations mathématiques, et aussi donner naissance à un nouveau système de chiffrement basé sur le SEC qui est le système de chiffrement hybride.

À travers la première contribution, nous avons pu mettre en place une fonction objective avec contrainte. Cette dernière a pu éliminer la possibilité d'avoir une solution optimale avec des caractères qui gardent les mêmes positions que dans le texte d'origine. Sans oublier que cette fonction garanti toujours les mêmes propriétés que la fonction originale. Des expériences ont été menées et ont prouvée l'efficacité de cette nouvelle fonction d'évaluation aussi bien sur le plan sécurité que de la convergence et la rapidité du système.

La deuxième contribution réalisée dans cette thèse, a permis de mettre en œuvre une nouvelle formule mathématique pour la fonction d'évaluation et aussi faire appel au problème de partition dans la partie de chiffrement. Cette contribution a permis de rendre nulle la probabilité pour qu'un caractère du message clair soit apparu dans ses positions initiales. Nous avons pu avoir des très bons résultats après la mise en place de cette nouvelle fonction dans notre système.

Finalement, nous avons pu, à travers notre dernière contribution construire un nouveau système de chiffrement basé sur le système SEC, avec une sécurisation de la clé toute en

utilisant le système de chiffrement RSA, ce nouveau système hybride a pu donner des résultats très satisfaisants.

**Perspectives :**

La clé générée par ASEC représente un point fort pour notre système dans la mesure où il permet de garantir sa sécurité : d'une part, par ce qu'elle est générée à travers le système lui-même et ne fait pas intervenir une autre application de génération de clé, et d'autre part par ce qu'elle s'agit d'une clé de session qui change d'une utilisation à une autre ce qui peut étendre non seulement la confidentialité du texte clair échangé mais aussi l'authentification de l'ensemble de la communication. Cependant, la Réduction de sa taille est sollicitée dans le cadre de ces perspectives.

En outre, nous avons d'autres perspectives dans le cadre de la performance de notre système aussi, et qui concerne l'optimisation de l'implémentation de ce système afin d'avoir un logiciel capable de conquérir les systèmes de chiffrement les plus répandus dans le domaine de la cryptographie en utilisant une interface graphique plus sophistiqué facile à manipuler par le grand public, ou le développement d'un module qui peut être utilisé est installé sur les différents Framework, afin de garantir la sécurité des données manipulés.

Des travaux sont en cours de réalisation avec mon équipe de recherche pour affranchir d'autres domaines informatiques à savoir les blockchain, les big data en se basant sur notre nouveaux systèmes.

# BIBLIOGRAPHIE

[1] "D.-E.-S. Agha et al., “Security Enhancing by using ASCII Values and Substitution Technique for Symmetric Key Cryptography,” Indian Journal of Science and Technology, vol. 10, no. 36, pp. 1–6, Sep. 2017, doi: 10.17485/ijst/2017/v10i36/119181".

[2] "A. Kerckhoffs, “LA CRYPTOGRAPHIE MILITAIRE.pdf.” Journal des Sciences Militaires, Jan. 1983.".

[3] "H. Lehning, “Les révolutions cryptographiques autour du XXe siècle,” p. 7.".

[4] Jonathan Katz and Yehuda Lindell, Introduction to Modern Cryptography2007, ISBN 9781584885511 - CAT # C5513 , 31 août 2007 par Chapman and Hall / CRC .

[5] Zimmer, Sébastien. «Mécanismes cryptographiques pour la génération de clefs et l’authentification.» thèse. 2008..

[6] "D. Lamas, “La cryptographie,” Haute école de gestion de Genève, 2015. Accessed: Mar. 29, 2022. [Online]. Available: https://doc.rero.ch/record/258630".

[7] W.Weaver, C.E. Shannon et. The mathematical Theory of communication, University of Illinois, Urbana III. 1949..

[8] "ANSSI-PG-083, “GUIDE DES MÉCANISMES CRYPTOGRAPHIQUES.” A N S S I P G- 0 8 3, Jan. 01, 2020".

[9] "P. Guillot, “6 La théorie cryptologique,” in 6 La théorie cryptologique, EDP Sciences, 2021, pp. 127–150. doi: 10.1051/978-2-7598-0995-0.c008.".

[10] Hellman, W. Diffie et M. E. W. Diffie and M. E. Hellman. New Directions in Cryptography. IEEE Transactions. 1976..

[11] RIVEST (R.L.), SHAMIR (A.) et ADLEMAN (L.M.).– A method for obtaining digital signatures and public-key cryptosystem. Communications of the ACM, 21 (2), p. 120-126 (1978)..

[12] Article F.OMARY, A.Tragha, A.Lbekkouri, A.Bellaachia, A.Mouloudi / An Evolutionist Algorithm to Cryptography- Brill Academic Publishers – Lecture Series And Computational Sciences Volume 4, 2005, pp.1749-1752.

[13] Thése F.Omary Applications des algorithmes évolutionnistes à la cryptographie 2006 - Université Mohammed V -Faculté des sciences de Rabat.

[14] Article F.OMARY, A.Tragha, A.Lbekkouri, A.Bellaachia, A.Mouloudi / A New Ciphering Method Associated with Evolutionary Algorithm – Lecture Notes in Computer Science – Publisher : Springer Berlin / Heidelberg –ISSN: 0302-9743 –Subject :Computer Science-Vol.

[15] "P. C. van Oorschot, Computer Security and the Internet, Information Security and Cryptography,2020".

[16] F. G. e. N. S, les techniques de la cryptographie CNAM, 2002.

[17] B. Schneier, Cryptographie appliqué, seconde édition (John Wiley & Sons,1996)., 1996.

[18] S. V. A. M. P. OORSCHOT, Handbook of Applied Cryptography (CRC Press, 1997)., 1997.

[19] D. Stinson, CRYPTOGRAPHIE -Théorie et pratique, 2 eme édition ed., CRC Press, 2002.

[20] T. d. L. V. Bruce Schneier, CRYPTOGRAPHIE APPLIQUEE - Protocole, algorithme et codes source en C, Vuibert Informatique.

[21] D. S. -. T. d. S. V. G. A. e. P. Jundo, Cryptographie Théorie et pratique 2e édition, 2 ed., Paris, Paris, p. 12.

[22] "Z. Amrani, “Cryptage d’Images par Chiffrement de Vigenère Basé sur le Mixage des Cartes Chaotiques,” p. 5, 2007".

[23] B. Schneier, "Cryptographie appliquée. Seconde édition (John Wiley & Sons, 1996).".

[24] G. D. e. D. S., "http://www.cryptologie.com/Web_Ter," Travaux d’études et de recheche Cryptographie, Octobre 2001. [Online].

[25] L. G.(2001), "Introduction à la cryptographie.http://www.hsc.fr/ressources/cours/crypto/index.html.fr," 2001.

[26] "N. S. (BRIS) European Network of Excellence in Cryptology II, “ECRYPT II Yearly Report on Algorithms and Keysizes (2011-2012).” ICT-2007-216676.".

[27] B. V. Alice Lan, "Panorama des algorithmes de Cryptographie.," 2011.

[28] "Ö. Ismet and S. Ibrahim, “Analysis and Comparison of Image Encryption Algorithms.pdf.” International Journal of Information Technology, Volume 1 Number 2".

[29] "D.-J. Mercier, “Cryptographie classique et cryptographie publique à clé révélée,” p. 17.".

[30] "B. Gérard, “Cryptanalyses statistiques des algorithmes de chiffrement à clef secrète.,” p. 225.".

[31] "M. N. A. Wahid, A. Ali, B. Esparham, and M. Marwan, “A Comparison of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish for Guessing Attacks Prevention,” p. 7, 2018".

[32] S. Descombes, Vocabulaire et algorithmes de la cryptographie. Vol. JDNet. 2002..

[33] "W. Stallings, Cryptography and network security: principles and practice. Pearson Prentice Hall, 2017.".

[34] A. Menezes, P. van Oorschot, and S. Vanstone, Handbook of Applied Cryptography, Chapter7 :Block Ciphers page 223;CRC Press, 1996..

[35] Franck Leprévost, Touradj Ibrahimi,Bertrand Warusfel, Enjeux de la sécurité multimédia, Hermés Science Lavoisier, 2006.

[36] Douglas Stinson, Cryptographie, théorie et pratique,2 eme éditions Vuibert 2003.

[37] E.Bresson. CRYPTOGRAPHIE - Chiffrement symétrique. SGDN/DCSSI laboratoir de cryptographie. s.d. PDF..

[38] "A. Biryukov, Feistel Cipher. In: van Tilborg H.C.A., Jajodia S. (eds) Encyclopedia of Cryptography and Security. Springer, Boston, MA. ,2011".

[39] Jacques Patarin, Valerie Nachef, and C ´ ome Berbain. Generic Attacks on Unbalanced Feistel Schemes with Expanding Functions. In editor, Advances in Cryptology - ASIA CRYPT 2006, Lecture Notes in Computer Science. Springer, 2007..

[40] A. Menezes, P. van Oorschot, and S. VanstoneHandbook of Applied Cryptography,CRC Press, 1996.Chapitr 7:Block Ciphers page:263.

[41] "E. Biham, "Differential Cryptanalysis," Encyclopedia of Cryptography".

[42] "A. Biryukov and C. Cannière, "Linear Cryptanalysis for Block Ciphers"".

[43] "J.-P. Aumasson, Serious cryptography: a practical introduction to modern encryption. San Francisco: No Starch Press, 2018".

[44] "C. Carlet.Two new classes of bent functions. In Advances in Cryptology - EUROCRYPT'93, volume 765 de Lecture Notes in Computer Science, pages 77-101. Springer-Verlag, 1994.".

[45] "J. Hong et P. Sarkar. " Rediscovery of Time Memory Tradeoffs". Rapport Technique 2005/090, Cryptology ePrint Archive, 2005.http://eprint.iacr.org/2005/090.," [Online].

[46] "Nicolas Courtois and Willi Meier. Algebraic Attacks on Stream Ciphers with Linear Feed back. In Eli Biham, editor, Advances in Cryptology - EUROCRYPT 2003, volume 2656 of Lecture Notes in Computer Science, pages 345–359. Springer, 2003.," [Online].

[47] "M. U. Bokhari, S. Alam, and F. S. Masoodi, "Cryptanalysis Techniques for Stream Cipher: A Survey," International Journal of Computer Applications, vol. 60, no. 9, pp. 29–33, 2012.".

[48] "M. E. Hellman, A cryptanalytic time-memory trade off, IEEE Transactions on Information Theory, IT-26, pp. 401–406, 1980.".

[49] "Babbage, SH, «amélioré» recherche exhaustive "attaques sur flux chiffres," sécurité et de détection, 1995., Convention européenne, vol, non, pp.161-166, 16-18.. mai 1995".

[50] "Golic, J., "cryptanalyse des alléguée A5 chiffrement de flux" Lecture Notes in Computer Science, Advances in Cryptologie - EUROCRYPT '97, LNCS 1233, pp.239-255, Springer-Verlag 1997".

[51] "J. Hong, P. Sarkar.- New Applications of Time Memory Data Tradeoffs -. Advances in Cryptology - ASIACRYPT 2005, 3788 Lecture Notes in Computer Science, 353-372. Springer-Verlag, 2005.," [Online].

[52] "A. K. Lenstra and E. R. Verheul, “Selecting Cryptographic Key Sizes,” J. Cryptology, vol. 14, no. 4, pp. 255–293, Sep. 2001, doi: 10.1007/s00145-001-0009-4.".

[53] "T.Siegenthaler.- Decrypting a class of stream ciphers using ciphertext only-. IEEE Transaction. Computers; C-34(1):81-84, 1985.," [Online].

[54] L. G. N. P. Jacques Stern, «Conception et preuves d’algorithmes cryptographiques,» 2004.

[55] "A. Corfdir and H. Gilbert. "A known plaintext attack of feal-4 and feal-6". -In Advances in Cryptology -CRYPTO 91-; Santa Barbara, Californie, États-Unis, pages 172-181. Lectures Notes in Computer Science 576, Springer-Verlag, 1991.," [Online].

[56] "M. Matsui." Linear cryptanalysis method for des cipher"." In Advances in Cryptology -EUROCRYPT 93", Lofthus, Norvége, pages 386-397. Lecture Notes in Computer Science 765, Springer-Verlag, 1993.," [Online].

[57] "E. Biham and A. Shamir, Differential Cryptanalysis of the Data Encryption Standard. New York, NY: Springer New York, 1993. doi: 10.1007/978-1-4613-9314-6.".

[58] "– ,Data Encryption Standard (DES). United States: National inst of standards and technology gaithersburg md, 1999.".

[59] "C. Bidan, “Cryptographie et Cryptanalyse,” p. 81.".

[60] "Bernhard Korte,Jens Vygen.Optimisation combinatoire Théorie et algorithmes.Traduit de l’anglais par Jean Fonlupt et Alexandre Skoda,ISBN : 978-2-287-99036-6 Springer Paris Berlin Heidelberg New York-Springer-Verlag France 2010," [Online].

[61] Hopcroft, John E .; Motwani, Rajeev; Ullman, Jeffrey, Introduction to Automata Theory, Languages, and Computation (2d Edition) Addison-Wesley. ISBN 81-7808-347-7 .2001.

[62] Integer and Combinatorial Optimization- Nemhauser, G. and L. A. Wolsey 1988. John Wiley and Sons, New York..

[63] Melissa DeLeon, - A Study of Sufficient Conditions for Hamiltonian Cycles -, Department of Mathematics and Computer Science, Seton Hall University..

[64] Holland J.H., Adaptation in Natural and Artificial Systems.Cambridge, Mass: MIT press,1975..

[65] S. Kirkpatrick, C.D. Gelatt et M.P. Vecchi ,"Optimization by Simulated Annealing", Science, 220 (1983), pp. 671-680, 1983.

[66] Fred Glover et Manuel Laguna (1997), la recherche Tabou . 408 pages. ISBN 0-7923-9965-X , ISBN 0-7923-8187-4.

[67] Recherche tabou générique pour problèmes à contraintes binaires Alexandre Gondran- Alexandre Gondran. Recherche tabou générique pour problèemes à contraintes binaires. [Research Report] ENAC. 2014. HAL Id: hal-01063343- https://hal.archives-ouvertes.fr/ha, 2014.

[68] Rainer Storn, Kenneth Price, Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces, Journal of Global Optimization, volume 11, no 4, pages 341-359, 1997.

[69] ] Article F.OMARY, A.Tragha, A.Lbekkouri, A.Bellaachia, A.Mouloudi / A New Ciphering Method Associated with Evolutionary Algorithm – Lecture Notes in Computer Science – Publisher : Springer Berlin / Heidelberg –ISSN: 0302-9743 –Subject :Computer Science-V.

[70] "G. J. Simmons, “Symmetric and Asymmetric Encryption,” in Secure Communications and Asymmetric Cryptosystems, Routledge, 1982.".

[71] "P. Sindhu and D. G. Indirani, “IoT based Wheat Leaf Disease Classification using Hybridization of Optimized Deep Neural Network and Grey Wolf Optimization Algorithm,” vol. 24, no. 2, p. 19, 2020.".

[72] "M. Bougrine, F. Omary, and S. Trichni, “Security of a new hybrid ciphering system,” International Journal of Advanced Computer Science and Applications, vol. 11, no. 6, pp. 694–699, 2020.".

[73] "L. Ben, “What is a Digital Signature?,” SearchSecurity. https://www.techtarget.com/searchsecurity/definition/digital-signature".

[74] P. ZIMMERMANN, "Cryptographie et réseaux," *POUR LA SCIENCE,* vol. 260, no. 260, pp. 40-44, 1999.

[75] E. Biham and A. Shamir, "Differential Crypt analysis of DES-like Cryptosystems (Extended Abstract)," *Springer-Verlag Berlin Heidelberg,* vol. CRYPTO '90, no. LNCS 537, pp. 2-21, 1991.

[76] S. T. a. all, "A New Approach Of Mutation Operator Applied To The Ciphering System Sec," *Iccit,* Vols. vol 63, no. 9, no. sep 2013, 2011.

[77] C. C.-. H. P.-. M.-C. Portmann, "Genetic Algorithms And Their Application To Scheduling Problems (Les Algorithmes Genetiques Et Leur Application Aux Problemes D'ordonnancement)," *Apii,* Vols. Volume 29-N◦ 4-5, pp. 409-443, 1995.

[78] F. G. E. N. S:, Techniques Of Cryptography., Cnam : Cnam , 2002.

[79] G. D.E, Genetic Algorithms In Search Optimisation & Machine Learning, Inc: Addison-Wesley Publishing Company,, 1989.

[80] G. J.J, "Optimization Of Control Parameters For Genetic Algorithms," *Ieeetrans. On Smc, ,* Vols. Vol. 16, , no. 1, pp. 122-128., 1986.

[81] K. P. C, "Heuristic And Evolutionary Algorithms," University Of Lille, Lille, 1988.

[82] S. Drukmann:, "Evolutionary Algorithms.," *Encyclopedia Of Computational Neuroscience,* pp. 1-7, 2014.

[83] T. Back, "Evolutionary Algorithms In Theory And Practice," Oxford University Press, Oxford , 1996.