REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DE GUELMA
FACULTE DES SCIENCES ET DE L'INGÉNIERIE

Mémoire de Magister

Présenté au département d'informatique
Pour l'obtention du diplôme de

Magister en Informatique

Option : Intelligence Artificielle et Imagerie

Par M^lle^ SOUICI Ismahane

Cryptographie

Nouvel Algorithme de Chiffrement Evolutionnaire basé Occurrences (ACEO)

JURY

Président	Mr	N. FAREH	M.C	Université d'Annaba
Rapporteur	Mr	H. SERIDI	M.C	Université de Guelma
Examinateur	Mr	T. KHADIR	M.C	Université d'Annaba
Examinateur	Mme	L. SOUICI	M.C	Université d'Annaba

2008

Remerciements

Mes remerciements vont tout premièrement à Dieu tout puissant pour la volonté, la santé et la patience qu'il m'a donné durant toutes mes années d'études.

J'exprime toute ma gratitude à mon encadreur monsieur **Seridi Hamid**, maître de conférences à l'université de Guelma, pour la confiance qu'il m'a témoigné, la liberté d'initiative qu'il m'a toujours accordé, sa constante disponibilité ainsi que l'intérêt qu'il a porté à l'achèvement dans les meilleurs conditions possibles de ce travail.

Je tiens à remercier également :

Monsieur **Fareh Nadir**, maître de conférence à l'université de Annaba, d'avoir accepter d'honorer cette soutenance comme président de jury. Qu'il me soit permet de lui exprimer ma plus haute considération.

Messieurs **Khadir Tarek**, maître de conférence à l'université de Annaba, madame **Souici Labiba**, maître de conférence à l'université de Annaba, d'avoir accepter d'évaluer ce travail, en tant que membre de jury.

Il m'est très agréable de témoigner ma grande reconnaissance à monsieur **Pierre Collet**, de l'Université Louis Pasteur à Strasbourg, à monsieur **Marc Schoenauer**, de l'Université Paris Sud, et à monsieur **Boukadoum Mounir**, de l'Université du Québec à Montréal, pour leur riches commentaires et pour les discussions fructueuses que nous avons eu. Qu'ils trouvent ici l'expression de ma profonde estime.

Enfin, je remercie tous ceux qui m'ont accompagné ou que j'ai croisé durant ces deux années à l'université de Guelma, tous ceux qui m'ont soutenu, encouragé et donné l'envie de mener à terme ce travail. Merci à : **Nadjet, Loubna, Mounira, Razika, Souhila, Karima, Soulef, Saoussan,** pour leur sympathie et leur soutien.

A tous ceux qui me sont chers, pour leur amour et leur patience…

Dédicace

Je dédie ce modeste travail à :

Mes très chers parents

Mes frères et sœurs

Mes amis

Résumé

La cryptographie est une promesse de sécurité dans un monde en réseau présentant de véritables gros risques, ce qui le rend impropre à toute utilisation militaire, commerciale, …, où l'intégrité et la confidentialité des données ainsi que l'authentification des correspondants sont fondamentaux. Toutefois, les vrais algorithmes de chiffrement sont de nombre limité. Dans cette perspective nous avons développé un nouvel algorithme de chiffrement s'inscrivant sous une nouvelle approche cryptographique, fertile et très prometteuse : la cryptographie évolutionniste. Partons du codage du message à chiffrer, cet algorithme cherche la solution soi-disant optimale, qui est un message chiffré en maximisant la différence entre les nombres d'occurrences des caractères dans les deux messages. Grâce à cette propriété, en plus de la large utilisation du caractère aléatoire par les algorithmes évolutionnaires, l'algorithme proposé a montré un considérable pouvoir de confusion assurant une force irrésistible face à plusieurs types d'attaques.

Mots-clés : Algorithmes évolutionnistes, cryptographie, intégrité, confidentialité, authentification, chiffrement, cryptanalyse, clé de session.

Abstract

Cryptography is a promise of security in a networked world presenting real big risks, which makes it unsuitable for any military use, commercial, …, where the integrity and confidentiality of data and correspondents authentication are fundamental. However, the real encryption algorithms are rare. In this perspective, we have developed a new ciphering algorithm enrolling in a new cryptographic approach, fertile and very promising: evolutionary cryptography. Let's encoding of the plaintext, this algorithm searches for the solution so-called optimal, which is a ciphertext by maximising the difference between the numbers of occurrences of characters in the two messages. Due to this property, in addition to the large use of the random character by the evolutionary algorithms, the proposed algorithm showed a considerable confusion power assuring an irresistible force face several kinds of attacks.

Key words: Evolutionary algorithms, cryptography, integrity, confidentiality, authentication, ciphering, cryptanalysis, session key.

الخلاصة

التشفير هو وعد أمن في عالم الشبكات الذي يمثل خطرا حقيقيا، الأمر الذي جعل منه غير ملائم للاستعمالات العسكرية، التجارية، ...، حيث أن سلامة و سرية البيانات إضافة إلى تحديد هوية المراسلين، تعد أمورا أساسية. و بما أن خوارزمات التشفير الجيدة محدودة العدد، و من هذا المنطلق قمنا بتطوير خوارزم تشفير جديد يندرج ضمن نهج تشفيري جديد بدوره : التشفير التطوري. انطلاقا من رمز الرسالة المراد تشفيرها، يقوم الخوار زم بالبحث عن الحل الأمثل إلى حد ما، و الذي يتمثل في الرسالة المشفرة، و ذلك بتعظيم الفارق بين أعداد ظهور الحروف في الرسالتين. اعتمادا على هذه الخاصية إضافة إلى الاستعمال الواسع لعنصر العشوائية من قبل الخوارزمات التطورية، أظهر الخوارزم المقترح قدرة تشويشية كبيرة تمده بقوة لا سبيل إلى مقاومتها عند تعرضه لمختلف الأنواع من الهجمات.

المفاتيح: الخوارزمات التطورية، علم التشفير، النزاهة ، السرية ، التوثيق ، التشفير، تحليل الشفرات، مفتاح الدورة

Sommaire

Chapitre II : Algorithmes évolutionnaires

Chapitre III : Algorithme de Chiffrement Evolutionnaire basé Occurrences (ACEO)

Chapitre IV : Expérimentation et Résultats

Liste des figures

Liste des tableaux

Sigles et acronymes

3DES :	Triple DES
ACEO :	Algorithme Evolutionnaire de Chiffrement à base des Occurrences
ADN :	Acide DésoxyriboNucléique
AES :	Advanced Encryption Standard
AEs :	Algorithmes Evolutionnaires
ASCII :	American Standard Code for Information Interchange
CGA :	Canonical Genetic Algorithm
CSC :	CS Cipher
DEA :	Data Encryption Algorithm
DES :	Data Encryption Standard
DSA :	Digital Signature Algorithm
FBI :	Federal Bureau of Investigation
FIPS :	Federal Information Processing Standard Publication
GCHQ :	Government Communications HeadQuarters
GPG :	Gnu Privacy Guard
GPL :	General Public License
GSM :	Global System for Mobile communications
IA :	Intelligence Artificielle
IBM :	International Business Machines
IDEA :	International Data Encryption Algorithm
IETF :	Internet Engineering Task Force
LISP :	List Processing
MIT :	Massachusetts Institute of Technology
MPX :	Maximal Preservative X
NIST:	National Institute of Standards and Technology
NSA :	National Security Agency

OTL : Omary, Tragha et Lbakkouri

OX : Order Cross-over

PGP : Pretty Good Privacy

RC : Rivest Cipher

RSA : Rivest, Shamir, et Adleman

SSL : Secure Sockets Layers

Introduction générale

Depuis toujours, l'homme a ressenti le besoin de transmettre des informations de manière sûre, surtout avec le progrès de l'informatique et l'apparition de l'Internet où la **cryptographie**, qui est la science du chiffrement, s'est imposée comme passage incontournable dans le transit des informations sensibles. Ainsi, on la retrouve dans des domaines variés tels que : la protection des données confidentielles (bases de données, email, ...), la sécurisation des communications, le paiement sécurisé (cartes bancaires) ...

Malgré ses diverses formes, sans cesse en progrès, le fonctionnement de la cryptographie reste peu connu par le grand public mise à part quelques formes primitives englobant des algorithmes rudimentaires dans leur ensemble du fait qu'ils consistaient notamment au remplacement de caractères par d'autres. Quand aux applications modernes, avec usage des mathématiques, de la physique quantique, et des technologies les plus avancées, elles sont encore beaucoup moins connues. Ces dernières permettent de protéger le contenu d'un message en s'aidant le plus souvent de clés, et suivant que ces clés de chiffrement sont gardées secrètes ou pas, les algorithmes de chiffrement se scindent en deux principales catégories: les algorithmes à clé publique (RSA, ElGamal,…) et les algorithmes à clé privée (DES, 3DES, AES, IDEA,…). La cryptographie hybride consiste en une association des meilleures fonctionnalités des deux techniques de cryptage précédentes (PGP ,…). Mais avec la puissance montante des ordinateurs, la cryptographie reste un domaine en plein mouvement pour suivre les nombreux progrès des cryptanalyses. Ainsi, certains algorithmes ont été remis en cause, tel que le DES qui a été remplacé par 3DES puis récemment par AES. De plus, la sécurité de RSA est directement liée au progrès de la factorisation d'entiers qui peut être révélé un jour.

En parallèle, ces dernières années ont vu l'émergence de techniques de vie artificielle imitant les processus de l'évolution naturelle pour résoudre des problèmes complexes. Ces méthodes d'optimisation ou d'apprentissage permettent de résoudre des problèmes auxquels

les méthodes classiques n'apportent pas de réponses satisfaisantes. Les algorithmes dits **évolutionnaires** représentent une importante catégorie de ces méthodes. Ils couvrent un ensemble de techniques, nommées algorithmes génétiques, stratégies d'évolution, programmation évolutionnaire, et programmation génétique. Leurs domaines d'application n'ont cessé de s'élargir vu leurs multiples points forts (performance, rapidité, simplicité de leurs opérateurs …), où ils connaissent un grand succès surtout dans le domaine de l'intelligence artificielle et recherche opérationnelle pour résoudre des problèmes d'optimisation. En plus, leur large exploitation du caractère aléatoire dans la majorité des étapes de leur processus, encourage leur exploitation dans le domaine de la cryptographie, puisque ceci complique grandement la tâche des cryptanalystes.

L'objectif de notre travail est de concevoir et implémenter un nouvel algorithme de chiffrement évolutionnaire qui vienne s'ajouter à la bibliothèque cryptographique. Pour se faire, le mémoire est structuré en quatre chapitres.

Le *chapitre I* est consacré à l'état de l'art sur la cryptographie depuis sa première apparition jusqu'à nos jours, à travers lequel nous présentons les principes fondateurs de la cryptographie à savoir la notion de cryptologie, de clé de chiffrement, les fonctions de la cryptographie, les différents types d'attaque. Ensuite, nous traitons les grandes catégories d'algorithmes de chiffrement qui sont la cryptographie classique et la cryptographie moderne.

Le *chapitre II* énumère les grandes familles d'algorithmes évolutionnaires et présente le vocabulaire de ce domaine, les différentes étapes du processus évolutionnaire ainsi que les techniques de base utilisées pour la conception d'un algorithme évolutionnaire simple, robuste et efficace. La dernière section de ce chapitre expose les principales méthodes de parallélisation des algorithmes évolutionnaires.

Le *troisième chapitre* est consacré à la présentation du nouvel **A**lgorithme de **C**hiffrement **E**volutionniste basé sur les **O**ccurrences (**ACEO**), où les détails de ses différentes étapes sont illustrés d'exemples.

Les résultats expérimentaux obtenus sont ensuite présentés dans le *quatrième* et dernier *chapitre*, suivis d'une discussion d'évaluation de cet algorithme en comparaison avec les plus connus des algorithmes de chiffrements des différentes catégories.

Nous clôturons ce mémoire par une conclusion générale en donnant une synthèse sur tout ce qui a été fait ou acquis tout en mentionnant les améliorations qui pourraient être apportées et les perspectives qu'elles offrent.

Chapitre I

Cryptographie

1. Introduction

Depuis toujours, l´être humain a cherché à conserver certaines informations ou données secrètes, à défaut, à en restreindre l´accès à certaines personnes. C'est pourquoi, et dès l'antiquité, les peuples employèrent des codes secrets dans certains de leurs textes: les archéologues en ont découvert dans des hiéroglyphes égyptiens [1]. De même, les Hébreux dissimulaient parfois leurs écrits en inversant l'alphabet, c'est-à-dire en employant la dernière lettre de l'alphabet à la place de la première, l'avant-dernière lettre à la place de la deuxième, et ainsi de suite. Sur le champ de bataille, les Spartes communiquaient souvent avec leurs généraux par le biais de messages écrits sur un ruban de parchemin enroulé en spirale sur un bâton de diamètre défini, appelé scytale. Une fois le ruban déroulé, on ne pouvait lire le message qu'en enroulant le ruban autour d'une règle identique. Jules César se servit également de codes secrets pour correspondre avec ses hommes, et laissa même son nom à un chiffre particulier.

Jusqu'au début du XXème siècle, la cryptographie a gardé une importance mineure, et les méthodes utilisées étaient bien souvent rudimentaires [2]. Mais lors de la seconde guerre mondiale, et avec l´apparition de technologies de communication évoluées, telles que la radio, a rendu nécessaire la mise au point de mécanismes de cryptage empêchant l´interception des signaux par l´ennemi. Il était devenu indispensable de chiffrer les données transmises par les ondes (Enigma).

Avec l'avènement des réseaux, et tout particulièrement Internet, la cryptographie prend maintenant une nouvelle dimension, économique cette fois [3]. C'est en effet toute la sécurité du commerce électronique qui dépend maintenant de l'inviolabilité des codes cryptés. Ainsi la cryptographie s'élargit du domaine confidentiel de la protection des gros serveurs (universités,

entreprises, état) à la consommation de masse par les particuliers (commerce électronique, confidentialité des mails,...). À l'inverse, la cryptanalyse (craquage des codes cryptés) change elle aussi d'acteurs et d'objet. Jusqu'alors arme militaire et jeu de quelques génies travaillant pour la célébrité, elle devient une véritable arme de vol à grande échelle (détournement de codes de carte bleue, de fonds,...) et de guerre économique (vol de secrets industriels ou commerciaux).

2. Principes fondateurs de la cryptographie

2.1. Notion de cryptologie

La cryptographie compte parmi les différents systèmes d'écriture permettant de modifier de façon volontaire les caractères d'un message. Donc, ce procédé protège une communication qui devient lisible uniquement par l'expéditeur et par le destinataire auquel le message est adressé.

La *cryptographie* appartient à la **cryptologie** du grec *kruptos* « secret, caché » et *logos* « discours », qui est la science de l'écriture secrète englobant des pratiques concurrentes à savoir la **cryptographie**, le **déchiffrement** et la **cryptanalyse**. La première pratique qui est la cryptographie (du grec *kruptos* et *graphein* [4]) est : « la discipline incluant les principes, les moyens et les méthodes de transformation des données, dans le but de masquer leur contenu, empêcher leur modification ou leur utilisation illégale, ainsi que les opérations inverses, pour rendre le document à nouveau intelligible » [5]. Le déchiffrement est le processus permettant de transformer le message chiffré en message clair. Quand à la cryptanalyse, qui est un terme créé par le cryptologue américain *William Friedman* en 1920 (du grec *kruptos* et *analusis* « résolution, dissolution » [4]), est l'art de décoder un message chiffré en mêlant une intéressante combinaison de raisonnement analytique, d'application d'outils mathématiques, de découverte de redondances, de patience, de détermination, et de chance.

Les deux disciplines de cryptographie et de cryptanalyse s'alimentent l'une l'autre [6]. On ne peut pas évaluer la sécurité d'un mécanisme sans le soumettre à des attaques qui, à leur tour, conduisent à des critères de conception pour rendre les procédés plus sûrs. Ces derniers seront à nouveau passés au crible du cryptanalyste…

2.2. Terminologie

La cryptologie, et par conséquent la cryptographie, est essentiellement basée sur l'arithmétique [7]. Il s'agit dans le cas d'un texte de transformer les lettres qui composent le message en une succession de chiffres, puis ensuite de faire des calculs sur ces chiffres pour :

- ✓ d'une part les modifier de telle façon à les rendre incompréhensibles. Le résultat de cette modification (le message chiffré) est appelé **cryptogramme** (en anglais *ciphertext*) par opposition au message initial, appelé **message en clair** (en anglais *plaintext*) ;
- ✓ faire en sorte que le destinataire saura les déchiffrer.

Le chiffrement se fait généralement à l'aide d'une *clef de chiffrement*, le déchiffrement nécessite quant à lui une *clef de déchiffrement*. On distingue généralement deux types de clefs :

- **Les clés symétriques:** il s'agit de clés utilisées pour le chiffrement ainsi que pour le déchiffrement. On parle alors de *chiffrement symétrique* ou de *chiffrement à clé secrète*.
- **Les clés asymétriques:** il s'agit de clés utilisées dans le cas du *chiffrement asymétrique* (aussi appelé *chiffrement à clé publique*). Dans ce cas, une clé différente est utilisée pour le chiffrement et pour le déchiffrement.

On appelle *décryptement* (*décryptage*) le fait d'essayer de *déchiffrer illégitimement* le message (que la clé de déchiffrement soit connue ou non de l'*attaquant*). Lorsque la clef de déchiffrement n'est pas connue de l'attaquant on parle alors de **cryptanalyse** ou **cryptoanalyse** (on entend souvent aussi le terme plus familier de *cassage*).

Le processus cryptographique peut être récapitulé par la figure ci-dessous :

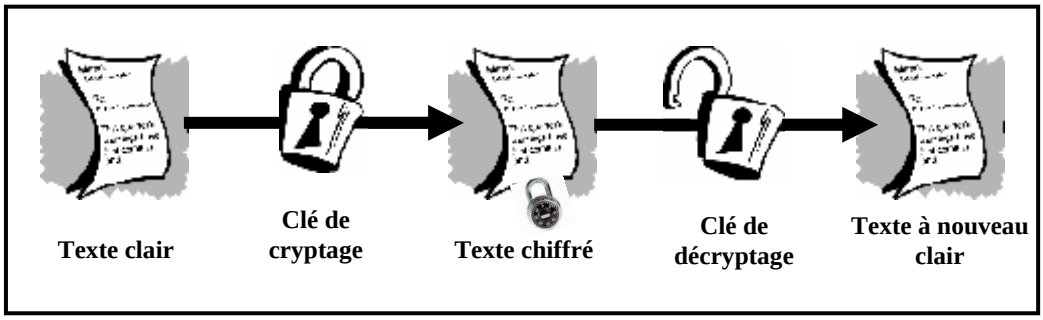


Figure I.1. *Processus cryptographique*

2.3. Les grandes menaces et les fonctionnalités offertes par la cryptographie

2.3.1. Les grandes menaces

De façon générale, les grands types de menaces que peut subir un message lors de son échange peuvent être récapitulés à travers les points suivants :

a. Les attaques passives : Avec ce type d'attaque et lors d'une communication élaborée entre deux personnes souvent nommées *Alice* et *Bob*, *Oscar* qui est l'opposant ou l'attaquant, se contente d'écouter le message tout en essayant de menacer sa confidentialité. Dans ce cas, il se peut qu'une information secrète parvienne également à une personne autre que son destinataire légal.

b. Les attaques actives : Ici, *Oscar* peut menacer l'intégrité, qui sera présentée dans la section suivante. Ainsi, ces informations vont parvenir d'une personne autre que leur véritable auteur. Et comme exemple d'attaques actives, on peut citer [8]:

- ✓ l'usurpation d'identité (de l'émetteur ou du récepteur) ;
- ✓ l'altération / modification du contenu des messages ;
- ✓ la destruction de messages/ le retardement de la transmission ;
- ✓ la répétition de messages (jusqu'à engorgement) ;
- ✓ la répudiation de message : l'émetteur nie avoir envoyé le message.

c. La cryptanalyse : Elle permet d'étudier la sécurité des procédés de chiffrement utilisés en cryptographie. Ainsi, elle désigne habituellement les techniques qui permettent d'extraire de l'information sur des secrets en observant uniquement les données publiques d'un cryptosystème. Les deux types de secrets sont le message clair et la clé. Ce qui compte avant tout dans une cryptanalyse, c'est de gagner de l'information sur le message clair. Ceci dit, il va de soi que gagner de l'information sur la clé de chiffrement privé permettant de déchiffrer tous les messages, ce qui résout définitivement le problème. Et suivant les données qu'elle nécessite, on distingue habituellement quatre méthodes de cryptanalyse :

- **attaque sur texte chiffré seul (ciphertext-only) :** le cryptanalyste possédant des exemplaires chiffrés des messages, essaye de faire des hypothèses sur les messages originaux qu'il ne possède pas en vue de retrouver la clé de déchiffrement. Dans ce cas, la cryptanalyse sera très difficile à cause du manque d'informations à disposition.
- **attaque à texte clair connu (known-plaintext attack) :** le cryptanalyste essaye de retrouver la clé de déchiffrement à partir de messages ou de parties de messages en clair possédés et de leurs versions chiffrées correspondantes.
- **attaque à texte clair choisi (chosen-plaintext attack) :** Consiste à retrouver la clé de déchiffrement à partir de messages en clair, et en ayant la possibilité de générer les versions chiffrées de ces messages avec un algorithme considérer comme une boîte noire.

- **attaque à texte chiffré choisi (chosen-ciphertext attack) :** le cryptanalyste possède des messages chiffrés et demande la version en clair de certains de ces messages pour mener l'attaque (retrouver la clé de déchiffrement).

2.3.2. Les fonctions de la cryptographie

La cryptographie est traditionnellement utilisée pour dissimuler des messages clairs aux yeux de certains utilisateurs pour assurer leur *fiabilité* et *confidentialité* surtout s'ils feront l'objet des communications via Internet. Désormais, les fonctions de la cryptographie se sont étendues pour englober de nouvelles fonctions en plus de la fiabilité et la confidentialité, il s'agit de garantir l'*intégrité* et l'*authenticité* des données échangées. Ceux-ci sont les fonctions principales de la cryptographie. Elle a d'autres fonctions, dites secondaires, qui sont [9] : l'*horodatage*, le *témoignage*, l'*accusé de réception* et la *révocation.*

a. La confidentialité : Permet de protéger le contenu des informations sauvegardées ou transmises sur un réseau. Seules les personnes autorisées doivent pouvoir accéder aux informations ainsi protégées. Le *chiffrement de l'information* permet de résoudre le problème de la confidentialité: une personne souhaitant transmettre un message lui applique au préalable une fonction dite de chiffrement, et transmet le résultat au destinataire. Ce dernier retrouve le message original en utilisant une fonction de déchiffrement suivant le modèle de la cryptographie utilisé. Dans le modèle de la cryptographie à clé secrète les deux parties partagent la même clé de chiffrement et de déchiffrement, qui doit être gardée secrète. Les deux personnes jouent ainsi un rôle symétrique, tandis que, dans le modèle de la cryptographie à clé publique, le chiffrement est publique et le déchiffrement est confidentiel. Pour envoyer un message chiffré, on applique une fonction de chiffrement utilisant la clé publique du destinataire. Ce dernier est le seul qui peut retrouver le message original à l'aide de sa clé privée. Les deux clés sont liées mathématiquement, mais il doit être impossible dans la pratique de retrouver la clé privée à partir de la clé publique (plus de précisions, ainsi que quelques exemples de méthodes sur les deux modes de chiffrement, symétrique et asymétrique, seront donnés en avant respectivement dans les sections 3.3.2 et 4).

b. L'intégrité : C'est la capacité à reconnaître qu'une information a été altérée [10]**,** soit de manière accidentelle ou intentionnelle.

c. L'authentification : Consiste à assurer l'identité d'un utilisateur, c'est à dire de garantir à chacun des correspondants que son partenaire est bien celui qu'il doit être.
On distingue deux types d'authentification :

- **Authentification d'un tiers :** C'est l'action qui consiste à prouver son identité. Ce service est généralement rendu par l'utilisateur d'un « échange d'authentification » qui implique un certain dialogue entre les tiers communicants.
- **Authentification de l'origine des données :** Elle sert à prouver que les données reçues ont bien été émises par l'émetteur déclaré. Dans ce cas, l'authentification désigne souvent la combinaison de deux services : authentification et intégrité.

d. La non-répudiation : La non-répudiation de l'information est la garantie qu'aucun des correspondants ne pourra nier la transaction [11].

Ces fonctionnalités représentent des solutions aux problèmes causés par les menaces citées précédemment, ainsi :

- ✓ Pour assurer la confidentialité, on utilise un algorithme de chiffrement.
- ✓ Contre l'usurpation d'identité, on utilise des algorithmes d'authentification.
- ✓ Pour empêcher l'altération de données, on utilise des algorithmes de contrôle d'intégrité.
- ✓ Contre la répudiation, des algorithmes de signatures ont été proposés.

Les besoins de sécurisation et de confidentialité s'imposent à divers degrés dans différentes applications. Citons à titre d'exemple :

- ✓ Confidentialité des transactions bancaires,
- ✓ Protection de secrets industriels ou commerciaux,
- ✓ Protection des secrets médicaux,
- ✓ Protection des systèmes informatiques contre les intrusions,
- ✓ Protection de la confidentialité des communications dans le cadre d'une association d'un parti politique, d'un syndicat…
- ✓ Protection de la vie privée,
- ✓ Jeux
- ✓ Etc…

3. Algorithmes cryptographiques

Comme nous l'avons déjà mentionné, le but de la cryptographie est de permettre à deux personnes de s'échanger des informations en toute sécurité à travers un canal peu sûr, qui peut être une ligne téléphonique ou tout autre réseau de communication. L'information que l'on souhaite transmettre et que l'on appelle texte clair, sera donc chiffrée par un procédé de chiffrement et en utilisant une clé prédéterminée. Le destinataire est le seul qui peut retrouver l'information originale suite à une opération de déchiffrement de l'information chiffrée en utilisant une clé de déchiffrement sans laquelle ce procédé est impossible.

Le processus de chiffrement ou de déchiffrement utilise une fonction mathématique. C'est l'**algorithme cryptographique** ou encor appelé **chiffre**. La sécurité des données chiffrées est entièrement dépendante de deux choses : la force de l'algorithme cryptographique et le secret de la clé. Un algorithme cryptographique, plus toutes les clés possibles et tous les protocoles qui le font fonctionner constituent un **cryptosystème.**

3.1. Le principe de kerckhoffs

Longtemps, la sécurité d'un système cryptographique a reposé sur le secret qui l'entoure (cas du chiffre de César, du code ADFVGX utilisé par les Allemands durant la première guerre mondiale, …). Cette idée s'est ensuite abandonnée du fait qu'un tel secret peut toujours être révélé par un espion, sinon une étude approfondie finira par percer son fonctionnement. C'est par exemple ce qu'ont réussi les Polonais en reconstituant les organes d'une machine Enigma et plus récemment, l'algorithme de chiffrement du GSM qui n'a jamais été officiellement révélé, on le trouve en détails sur le web. C'est pourquoi un système cryptographique doit dépendre d'un paramètre aisément modifiable : sa clé.

Le premier à avoir formalisé ce principe est le hollandais *Auguste Kerckhoffs* en écrivant en janvier 1883 dans le « Journal des sciences militaires » un article intitulé « La cryptographie militaire », où il disait [12]:

« Il faut bien distinguer entre un système d'écriture chiffrée, imaginé pour un échange momentané de lettres entre quelques personnes isolées, et une méthode de cryptographie destinée à régler pour un temps illimité la correspondance des différents chefs d'armée entre eux. Ceux-ci, en effet, ne peuvent, à leur gré et à un moment donné, modifier leurs conventions; de plus, ils ne doivent jamais garder sur eux aucun objet ou écrit qui soit de nature à éclairer l'ennemi sur le sens des dépêches secrètes qui pourraient tomber entre ses mains.

Un grand nombre de combinaisons ingénieuses peuvent répondre au but qu'on veut atteindre dans le premier cas; dans le second, il faut un système remplissant certaines conditions exceptionnelles, conditions que je résumerai sous les six chefs suivants:

1) *le système doit être matériellement, sinon mathématiquement, indéchiffrable.*
2) *Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénients tomber entre les mains de l'ennemi.*
3) *La clé doit pouvoir en être communiquée et retenue sans le secours de notes écrites, et être changée ou modifiée au gré des correspondants.*
4) *Il faut qu'il soit applicable à la correspondance télégraphique.*
5) *Il faut qu'il soit portatif, et que son maniement ou son fonctionnement n'exige pas le concours de plusieurs personnes.*
6) *Enfin, il est nécessaire, vu les circonstances qui en commandent l'application, que le système soit d'un usage facile, ne demandant ni tension d'esprit, ni la connaissance d'une longue série de règles à observer. »*

Les points 2 et 3 sont les axiomes fondamentaux de la cryptographie suivant lesquels l'attaquant possède tous les détails de l'algorithme sans pouvoir rien faire puisqu'il lui manque la clé spécifique pour le chiffrement. Cela mène vers la certitude suivante : si on ne sait pas casser un algorithme même en sachant comment il fonctionne, on ne sait certainement pas le casser sans cette connaissance. Donc, un chiffre basé uniquement sur le secret de l'algorithme n'a aucun intérêt, car un jour ou l'autre ce secret sera découvert ou volé. Par exemple, même si on connaît le mode d'emploi du carré de Vigenère (voir la section 3.3.1.1.b), on ne pourra quand même pas, ou difficilement, décrypter un message si on ne connaît pas la clef. Par contre, le chiffre Atbash, qui est une méthode de substitution alphabétique inversée sans utilisation de clé, repose entièrement sur la manière de chiffrer. Actuellement, on va encore plus loin où le mécanisme de chiffrement est publié afin que les cryptanalystes puissent l'étudier. D'ailleurs, on suppose toujours, en cryptanalyse académique, que le système de chiffrement est connu.

L'interprétation de ce principe par *Bruce Schneier* a portée à l'« élégance » dans le cassage d'un cryptosystème [13]. Traduit de l'anglais : « *Le principe de Kerckhoffs s'applique au-delà des chiffres et des codes, c'est-à-dire aux systèmes de sécurité en général : tout secret est en fait un point de cassure possible. Par conséquent, le secret est une cause première de*

fragilité, donc cela même peut amener un système à un effondrement catastrophique. À l'inverse, l'ouverture amène la ductilité. ».

Ici par fragilité, *Bruce Schneier* accentue sur le fait de garder comme secret une information peu coûteuse à remplacer en cas où le secret sera divulgué. Par exemple, si la sécurité d'un cryptosystème implémenté sur du matériel informatique et des logiciels géographiquement distants et largement dispersés, dépend de garder cette distribution secrète, alors sa divulgation demanderait de grands efforts en terme de développement, de tests et de distribution de nouveaux algorithmes. Mais si le secret de l'algorithme était tout simplement une clé, sa divulgation entraîne moins de problèmes puisqu'il suffit d'en générer une nouvelle et de la distribuer. En bref, moins on a de secrets, moins on doit faire de maintenance.

3.2. Description formelle d'un algorithme cryptographique

D'une manière formelle, un cryptosystème est un quintuplet (*P*, *C*, *K*, *E*, *D*) satisfaisant les points suivants [14]:

1) *P* est un ensemble fini de blocs de textes clairs possibles.

2) *C* est un ensemble fini de blocs de textes chiffrés possibles.

3) *K* est un ensemble fini de clefs possibles.

4) Pour tout $k \in K$, il y a une règle de chiffrement $e_k \in E$ et une règle de déchiffrement correspondante $d_k \in D$. Chaque $e_k : P \rightarrow C$ et $d_k : C \rightarrow P$ sont des fonctions telles que $d_k (e_k(x)) = x$ pour tout texte clair $x \in P$.

La principale propriété est la quatrième. Elle précise que si un texte clair x est chiffré en utilisant e_k, et si le texte chiffré y obtenu est ensuite déchiffré en utilisant d_k, on retrouve le texte clair x original.

Alice et Bob peuvent employer le protocole suivant pour utiliser un cryptosystème spécifique. Tout d'abord, ils choisissent une clé quelconque $k \in K$. Cette opération est effectuée lorsqu'ils se rencontrent en un même endroit loin d'être observés par Oscar, ou bien à travers un canal de communication sûr. Supposant qu'ensuite, Alice souhaite communiquer un message à Bob par un canal peu sûr, ce message étant une chaîne :

$$x = x_1 x_2 \ldots x_n$$

avec : $n \in Z$, $n \geq 1$, $x_i \in P$ et $1 \leq i \leq n$.

Chaque bloc x_i est chiffré en utilisant la règle de chiffrement e_k spécifiée par la clé k choisie. Ainsi, Alice calcule $y_i = e_k (x_i)$, $1 \leq i \leq n$, et la chaîne chiffrée obtenue sera:

$$y = y_1 \, y_2 \, \ldots \, y_n$$

Cette chaîne est envoyée dans le canal et une fois reçue par Bob, il la déchiffre en utilisant la fonction de déchiffrement d_k pour récupérer le texte clair original $x_1 \, x_2 \, \ldots \, x_n$. Le procédé de communication est illustré sur la figure I.2.

Il est évident que chaque fonction de chiffrement e_k doit être injective [14] (c'est à dire ne pas chiffrer deux blocs différents en deux valeurs égales), sinon, le procédé de déchiffrement ne pourrait être fait sans ambiguïté. Plus précisément, si

$$y = e_k (x_1) = e_k (x_2)$$

Avec : $x_1 \neq x_2$, Bob n'as aucun moyen de savoir si y doit être déchiffré en x_1 ou en x_2.

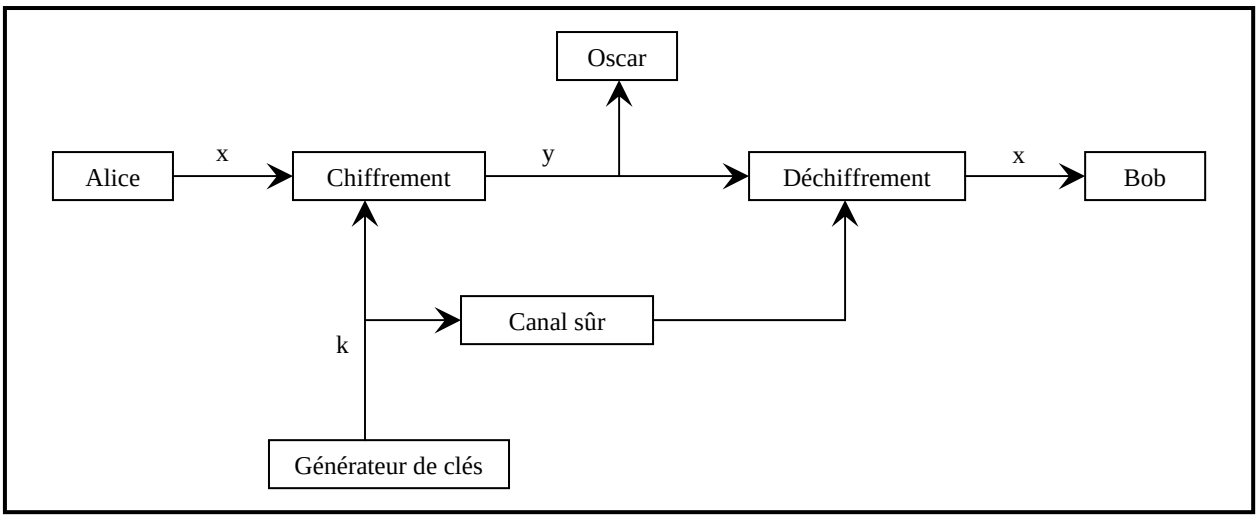


Figure I.2. *Le procédé de communication*

Les principes fondamentaux d'un algorithme de cryptographie sont basés sur deux notions essentielles, énoncées par *Shannon* en 1949:

- **Confusion :** Sert à cacher la relation entre le clair et le chiffré [15]. Donc, elle vise à rendre le texte aussi peu lisible que possible. Ceci peut se faire par une substitution méthodique de symboles, ou par un algorithme de codage aussi complexe que l'on veut. Comme ça aucune propriété statistique ne peut être déduite du message chiffré [16].
- **Diffusion :** Sert à cacher la redondance dans le message et à diffuser sur tout le chiffré l'influence du changement d'un bit de clef ou d'un bit du clair [15]. Donc, elle vise à rendre chaque élément d'information du texte chiffré dépendant d'un nombre aussi grand que

possible d'éléments d'information du texte clair. Ceci rend la découverte de l'algorithme, ou de la clé de cet algorithme, en principe plus difficile**.** Ainsi, toute modification du message en clair se traduit par une modification complète du chiffré [16].

3.3. Classes de la cryptographie

Le schéma suivant présente les différentes classes de la cryptographie :

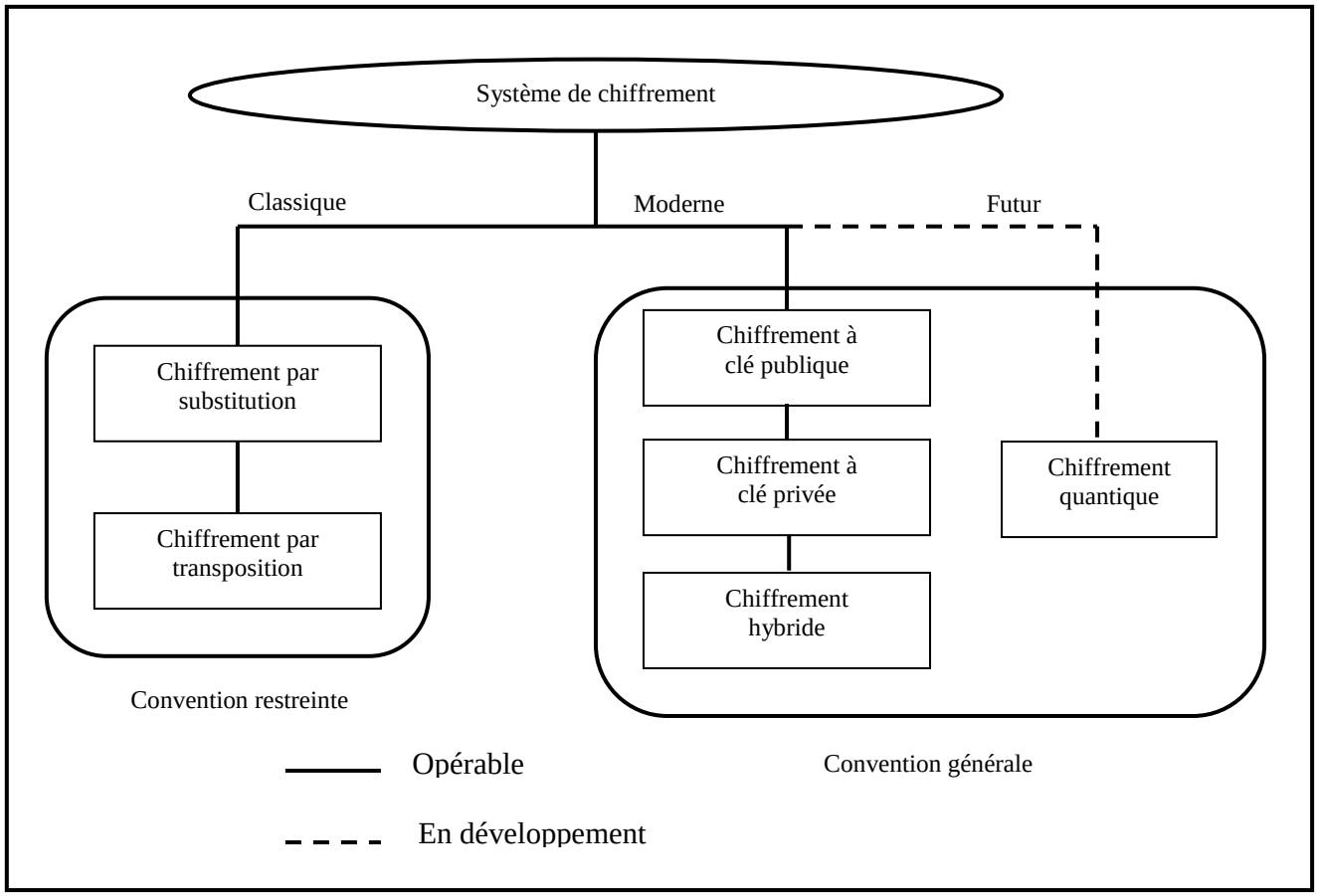


Figure I.3. *Les classes de la cryptographie*

3.3.1. La cryptographie classique

La cryptographie classique décrit la période avant les ordinateurs durant laquelle, les principaux outils utilisés consistent à remplacer des caractères par d'autres et les transposer dans des ordres différents tout en gardant secrètes les procédures de chiffrement ou de déchiffrement. Sans cela le système est complètement inefficace, puisque n'importe qui peut déchiffrer le message codé. On appelle généralement cette classe de méthodes : le chiffrement **à usage restreint**.

3.3.1.1. Cryptographie par substitution

Dans ce mode de cryptage, les lettres du message en clair sont remplacées par d'autres lettres, des chiffres ou d'autres symboles. Selon la façon de substituer, on distingue la substitution mono-alphabétique, la substitution homophonique et la substitution poly-alphabétique.

a. Substitution mono-alphabétique : C'est le plus simple des codages à réaliser. Il s'agit de remplacer chaque lettre par une lettre différente, ou même un autre symbole. Plus formellement [17]:

$$f : A_M \longrightarrow A_C$$

$$c_i = f(m_i) = m_i + k \pmod{|A_M|}$$

Où :

A_M et A_C sont respectivement l'ensemble d'alphabets du message en clair et l'ensemble d'alphabets du message chiffré.

$M = m_0\, m_1 \ldots m_{n-1}$ tel que : $\forall i, m_i \in A_M$.

$C = c_0\, c_1 \ldots c_{n-1}$ tel que : $\forall i, c_i \in A_C$.

La plus ancienne des méthodes s'inscrivant sous ce mode de chiffrement est le **chiffre de césar** utilisé par l'armée romaine (1^{ier} siècle avant JC). Il consiste à décaler les lettres de l'alphabet d'un nombre *n*. Par exemple, pour n=3, A sera remplacé par D, B par E, C par F...Son principe très simple à mettre en œuvre facilite sa cryptanalyse du fait que, le nombre de façon de chiffrer un message reste très faible, puisqu'il est égal au nombre de lettres de l'alphabet, c'est à dire que l'on a seulement 26 façons. Malgré ça, cette même simplicité a conduit les officiers sudistes à le réemployer durant la guerre de Sécession [18]. L'armée russe a fait de même en 1915. Une autre attaque possible contre ce système est la cryptanalyse fréquentielle qui se base sur le fait que les lettres les plus fréquentes dans le texte en clair restent les plus fréquentes dans le texte chiffré. Donc, ce système ne cache pas les fréquences d'apparition des caractères ce qui constitue une faiblesse importante permettant aux techniques statistiques d'associer une lettre probable aux lettres les plus fréquentes, et en appliquant une technique sémantique récursive, les algorithmes à base de substitutions mono-alphabétiques sont facilement cassés par les spécialistes.

Il est à noter que le code de César a été utilisé sur des forums Internet sous le nom de ROT13 (rotation de 13 lettres où A-->N...). Le ROT13 n'a pas pour but de rendre du texte

confidentiel, mais plutôt d'empêcher la lecture involontaire. Son utilisation est simple: il suffit de re-chiffrer un texte, codé en ROT13, une deuxième fois pour obtenir le texte en clair.

Dans cette catégorie, on peut citer aussi : les **alphabets désordonnés**, le **chiffre affine** [19], ...

Dans ce même mode de chiffrement, et lorsqu'une même lettre sera substituée par plusieurs lettres qui seront bien déterminées à l'avance ; par exemple, 'A' peut correspondre à 5, 13, 25 ou 56 ; 'B' à 7, 19, 31, ou 42,…; cette façon particulière de substitution mono-alphabétique est appelée **Substitution homophonique**. Ce procédé est plus sûr que le précédant (substitution mono-alphabétique), mais aussi craqué par les cryptanalystes ou par des espions expérimentés.

b. Substitution poly-alphabétique : Aussi appelée **à alphabets multiples**. Elle a été inventée par *Trithemius* en 1518 et cryptanalyser par *Kasiski* en 1863 [20]. Avec cette méthode, une même lettre peut être remplacée par plusieurs symboles pris aléatoirement. Cela est garantie grâce à une clé $k = k_0\ k_1\ \ldots k_{j-1}$ qui définie j fonctions distinctes fk_0, fk_1,…, fk_{j-1} définies comme suit [17]:

$$\forall i : 0 \le i < n \qquad f_{k_l} : A_M \longrightarrow A_C \qquad \forall l : 0 \le l < j$$

$$c_i = f_{k_i \bmod j}(m_i) = m_i + k_{i \bmod j} (\text{mod} |A_M|$$

Avec A_M, A_C, M et C auront la même signification que ceux utilisés lors de la présentation de la substitution mono-alphabétique.

L'exemple le plus fameux de chiffre poly-alphabétique est sans doute le **chiffre de Vigenère**, qui a résisté aux cryptanalystes pendant trois siècles. Ce chiffre a été présenté en 1586 par le diplomate français *Blaise de Vigenère* lors de la publication de son œuvre : « Traité des chiffres ou Secrètes manières d'écrire », tout en s'appuyant sur les bases de *Bellaso*, *Alberti*, *Porta* et *Trithème*. Il s'agit en réalité d'une amélioration du chiffre de César, puisqu'il exploite la même méthode, mais en changeant le décalage de lettre en lettre. De plus, il utilise les 26 alphabets écrits en carré, et en décalant d'une lettre à chaque fois (voir Figure I.4).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Figure I.4. *Le carré de Vigenère*

Pour coder un message en utilisant ce chiffre, on choisit une clé de longueur arbitraire et on la répète selon la longueur du message à coder. On l'écrit ensuite au dessus du message à coder lettre par lettre. La première lettre du message chiffré sera la lettre située à l'intersection de la ligne correspondant à la première lettre de la clé avec la colonne correspondant à la première lettre du texte à chiffrer (texte clair), et on continu ainsi jusqu'à terminer le chiffrement de tout le texte. Pour déchiffrer, il suffit de faire la même opération en sens inverse, c'est à dire que sur la ligne de la lettre de la clé on recherche la lettre du message codé, la véritable lettre se trouve alors au sommet de la colonne correspondante.

Son point fort, c'est qu'il offre des modes de codage et de décodage faciles à appliquer, et son plus grand intérêt est que la même lettre sera codée de différentes manières en pénalisant ainsi toute tentative de cryptanalyse fréquentielle à condition que la longueur du message à chiffrer ne soit pas bien plus longue que celle de la clé. Dans le cas contraire, il sera possible de repérer la longueur de la clé dans le message. Ainsi si, par exemple, la longueur de la clé est de 3, alors la première lettre du message est codée avec la première lettre de la clé, la deuxième avec la deuxième, la troisième avec la troisième, et on revient, la quatrième avec la première,... Dans ce cas, on peut déterminer les caractères de la clé un par un suite à une analyse de fréquences d'apparition. La solution consiste donc, à considérer une clé se rapprochant le plus possible de la longueur du message, malgré que même cela peu augmenter

les possibilités de commettre des erreurs vue que le travail devient difficile ; et par conséquence, le message devient indéchiffrable.

Beaucoup d'autres méthodes s'inscrivent sous ce mode de chiffrement. L'une qui a attiré l'attention des militaires allemands pendant la seconde guerre mondiale, est la **machine Enigma**, qui est une sorte de machine à écrire très complexe qui pouvait avoir plus de 10000000000000000 clés possibles.

3.3.1.2. Cryptographie par transposition

Ici, c'est l'ordre des éléments d'une information qui est modifié (caractères d'une phrase, pixels d'une image...), ce qui permet de mieux cacher le message. Plusieurs types de transposition existent.

a. Transposition simple par colonnes : On écrit le message horizontalement dans une matrice prédéfinie, et pour retrouver le texte chiffré, on lit la grille verticalement. Le procédé inverse représente le procédé de déchiffrement. La figure ci-dessous résume ce principe.

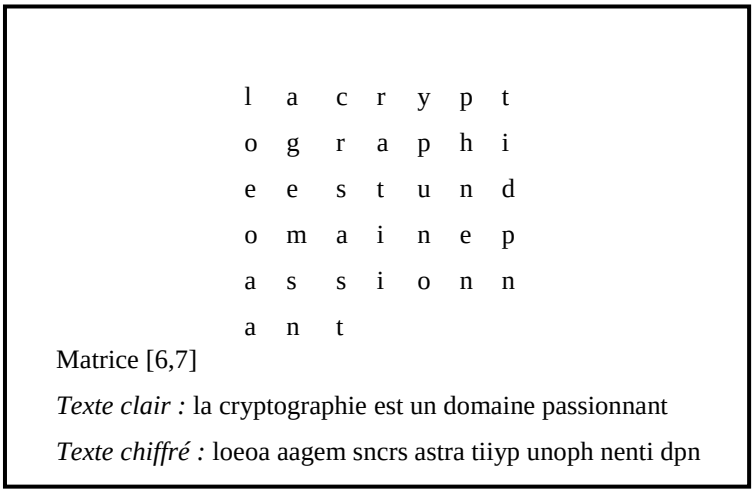


Figure I.5. *Transposition simple par colonne*

b. Transposition complexe par colonnes : Un mot clé secret constitué uniquement de caractères différents est utilisé pour construire une séquence de chiffres représentant les ordres d'apparition dans l'alphabet des différentes lettres composant ce mot. Le chiffrement se fait en écrivant d'abord le message par lignes dans un rectangle, comme le montre la figure I.6, puis on lit le texte par colonnes en suivant l'ordre déterminé par la séquence.

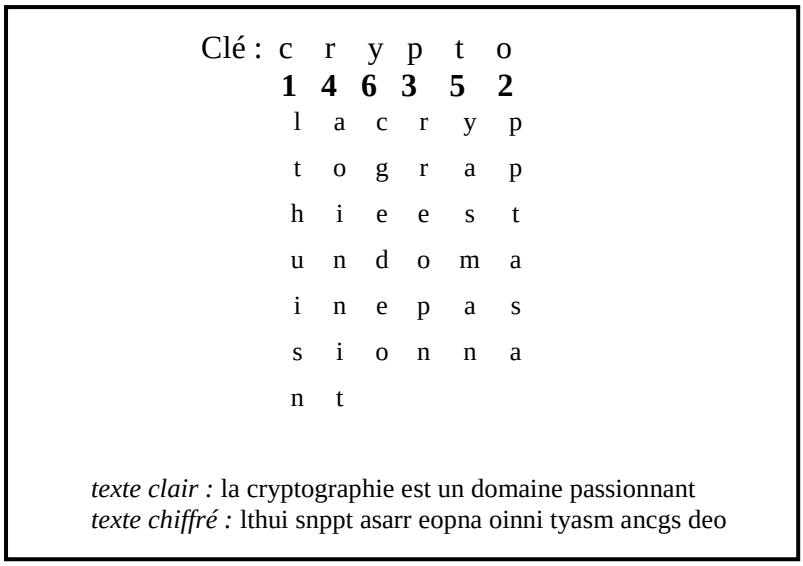
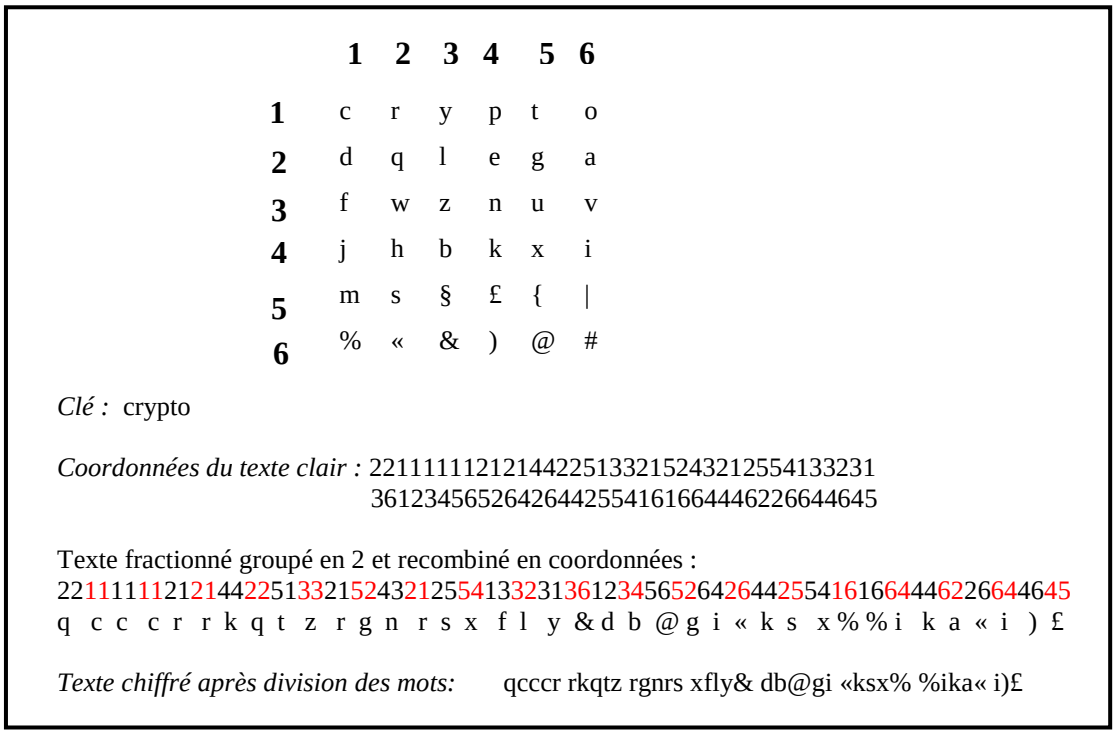


Figure I.6. *La transposition complexe par colonnes*

c. Transposition par carré polybique : Un mot clé secret est utilisé pour construire un alphabet dans un tableau, permettant d'extraire les coordonnées des lignes et des colonnes correspondant aux lettres du texte à chiffrer. Ainsi, chaque lettre du texte en clair est représentée par deux chiffres écrits verticalement sur deux lignes. L'étape qui suit, consiste à concaténer les deux lignes obtenues précédemment pour obtenir une seule ligne de chiffres, puis à recombiner ces chiffres deux par deux. Ces nouvelles combinaisons de chiffres représentent les coordonnées de lignes et de colonnes du texte chiffré.

Sur le même exemple de texte clair pris pour illustrer les façons précédentes de chiffrer (la cryptographie est un domaine passionnant), une illustration de ce mode de chiffrer est présentée dans la figure suivante.



	1	2	3	4	5	6
1	c	r	y	p	t	o
2	d	q	l	e	g	a
3	f	w	z	n	u	v
4	j	h	b	k	x	i
5	m	s	§	£	{	|
6	%	«	&	)	@	#

Clé : crypto

Coordonnées du texte clair : 2211111121214422513321524321255413323136123456526426442554161664446226644645

Texte fractionné groupé en 2 et recombiné en coordonnées :
22 11 11 11 21 21 44 22 51 33 21 52 43 21 25 54 13 32 31 36 12 34 56 52 64 26 44 25 54 16 16 64 44 62 26 64 46 45
q c c c r r k q t z r g n r s x f l y & d b @ g i « k s x % % i k a « i) £

Texte chiffré après division des mots: qcccr rkqtz rgnrs xfly& db@gi «ksx% %ika« i)£

Figure I.7. *Transposition par carré polybique*

Après la description de ces deux modes de cryptage (substitution et transposition), il est clair que les transpositions sont un peu plus sûres que les substitutions, mais elles ne fonctionnent que sur des messages à chiffrer d'une longueur limitée, en plus qu'elles sont plus gourmandes en mémoire ; ce qui limite leur utilisation dans les algorithmes.

3.3.2. La cryptographie moderne

Avec le développement des ordinateurs, les techniques de cryptographie ont clairement évolué, mettant à la touche ainsi les méthodes de cryptage manuel. Malgré ça, les procédés de substitution et de transposition restent toujours d'actualité mais en manipulant, cette fois-ci, des séquences de bits du fait que les ordinateurs ne manipulent que des données numériques ce qui rend les techniques de chiffrement actuelles plus sûres, voir même incassables pour certaines techniques, ou du moins prendraient des millions d'années avec la puissance actuelle des meilleurs supercalculateurs. D'autre part, il fait que maintenant les algorithmes ne sont plus cachés, mais au contraire sont connus de tous et leur sécurité est liée seulement aux clés utilisées.

La cryptographie moderne se scinde en deux parties nettement différenciées :

- ✓ La **cryptographie à clef secrète**, ou encore appelée **symétrique**;
- ✓ et la **cryptographie à clef publique**, dite également **asymétrique**.

La première, qui est la cryptographie symétrique, est la plus ancienne, et on peut la faire remonter à l'Égypte de l'an 2000 avant. J.C; la seconde, qui est la cryptographie asymétrique, remonte à l'article de *W. Diffie* et *M. Hellman*, « New directions in cryptography » daté de 1976 [21].

Ces classes visent, toutes les deux, à assurer la confidentialité de l'information communiquée, mais la cryptographie à clef secrète nécessite au préalable la mise en commun d'une certaine information entre les parties communicantes. Cette information est la clé nécessaire au chiffrement ainsi qu'au déchiffrement des messages, c'est pourquoi elle est dite *symétrique* Dans le cadre de la cryptographie à clé publique, deux clés différentes et qui ne peuvent se déduire l'une de l'autre, servent à faire les deux opérations opposées de chiffrement et de déchiffrement, d'où la notion d'*asymétrie.*

3.3.2.1. La cryptographie symétrique

a. Principe

Les algorithmes de chiffrement à clé secrète (ou symétriques ou encore dits conventionnels) sont ceux pour lesquels l'émetteur et le destinataire partagent une même clé secrète, autrement dit, les clefs de chiffrement et de déchiffrement sont identiques. L'emploi d'un algorithme à clé secrète lors d'une communication nécessite donc l'échange préalable d'un secret entre les deux protagonistes à travers un canal sécurisé ou au moyen d'autres techniques cryptographiques.

Un paramètre essentiel pour la sécurité d'un système à clé secrète est la taille de l'espace des clefs. En effet, il est toujours possible de mener sur un algorithme de chiffrement, une attaque dite *exhaustive* pour retrouver la clé. Cette attaque consiste simplement à énumérer toutes les clefs possibles du système et à essayer d'utiliser chacune d'entre elles pour décrypter un message chiffré. Si l'espace des clefs correspond à l'ensemble des mots de k bits, le nombre de tentatives d'attaque exhaustive en vue de décrypter le message chiffré est égal à 2^{k-1}. Donc, pour pénaliser une telle attaque, il faut que l'espace des clés soit suffisamment grand. À titre d'exemple et en janvier 1998 [22], une telle attaque a été réalisée contre le cryptosystème DES utilisant une clef secrète de 56 bits (voir section 4.1.2.2.c), en 39

jours sur 10 000 Pentium en parallèle, puis en 56 heures en juillet 1998 à l'aide d'une machine dédiée comportant 1500 composants DES.

Il existe d'autres types d'attaques sur les systèmes de chiffrement à clé secrète dont la plupart consistent à exploiter certaines structures particulières de l'algorithme ou certaines caractéristiques statistiques dans la distribution des couples de textes clairs-chiffrés. Les plus connues sont la *cryptanalyse différentielle,* inventée par les Israéliens *Biham* et *Shamir* en 1991 [22], et la *cryptanalyse linéaire* dont le principe a été initialement développé par *Gilbert, Chassé et Tardy-Corfdir* sur le chiffrement FEAL-8 puis appliqué au DES par le Japonais Matsui [23].

De façon générale, on considère qu'un chiffrement à clé secrète présente une bonne sécurité s'il n'existe pas d'attaque dont la complexité soit inférieure à celle de la recherche exhaustive. Et d'après les constatations empiriques qui sont la seule mesure de sécurité de ces systèmes, ils sont difficiles à cryptanalyser. Donc, la recherche en cryptographie symétrique se caractérise naturellement par l'enchaînement de phases de défense et d'attaque - on s'endort cryptographe et se réveille cryptanalyste, selon le bon mot de Marc Girault [23].

Les algorithmes symétriques sont de deux types :

- ✓ Les algorithmes de **chiffrement en continu**, qui agissent sur le texte en clair un bit à la fois. Ce mode de chiffrement est encore appelé **chiffrement en flux** (*Stream cipher* en anglais) ;
- ✓ Les algorithmes de **chiffrement par blocs** (*Bloc cipher* en anglais), qui opèrent sur le texte en clair par groupes de bits appelés blocs.

Comme algorithmes de chiffrement à clé secrète on peut citer : DES, 3DES, AES, Blowfish, RC (Rivest Cipher), CSC (CS Cipher), IDEA, etc.

3.3.2.2. La cryptographie asymétrique

a. Historique

Le concept de **cryptographie à clé publique,** qui est un autre nom de la **cryptographie asymétrique**, a été présenté pour la première fois par *Whitfield Diffie* et *Martin Hellman* dans leur article écrit pour le *National Computer Conference* en 1976, puis publié quelques mois plus tard dans *New Directions in Cryptography* sans pouvoir donner un exemple d'un système à clé publique. Il fallut attendre 1978 où la version académique du

premier cryptosystème à clé publique a fait l'objet d'un article intitulé : « A Method for Obtaining Digital Signatures and Public-key Cryptosystems » écrit par *Ronald Rivest, Adi Shamir,* et *Leonard Adleman*. C'est le cryptosystème RSA dont l'appellation est tirée des trois noms de ses auteurs.

En réalité, *James Ellis*, qui travaillait au service du chiffre britannique (GCHQ, *Government Communications Headquarters*), avait eu cette idée un peu avant [24]. En 1973, *C.C. Cocks* décrivit (pour le même service du chiffre) ce qu'on a appelé l'algorithme RSA. Enfin, en 1974, *M. J. Williamson* invente un protocole d'échange de clé très proche de celui de Diffie et de Hellman. Ces trouvailles n'ont été rendues publiques qu'en 1997 par le GCHQ.

b. Principe

La cryptographie à clé publique évite le partage d'un secret entre les deux interlocuteurs puisque, chaque utilisateur dispose d'un couple de clés : une clé publique qu'il met en général à disposition de tous dans un annuaire, et une clef secrète connue de lui seul. Ces deux clés, en plus d'être distinctes, elles ne peuvent se déduire l'une de l'autre. Alors, pour envoyer un message confidentiel à Bob, Alice chiffre le message clair à l'aide de la clé publique de Bob. Ce dernier, à l'aide de la clé secrète correspondante, sera le seul en mesure de déchiffrer le message reçu.

La notion primordiale sur laquelle repose le chiffrement à clé publique est celle de *fonction à sens unique avec trappe*. Sachons qu'une fonction est appelée à *sens unique* si elle est aisément calculée, mais difficile à inverser, ou plus exactement, infaisable en un temps réalisable avec une puissance de calcul raisonnable. Et une telle fonction sera dite *à trappe*, si le calcul de l'inverse devient facile dès que l'on possède une information supplémentaire qui est la *trappe*. Donc, la construction d'un système de chiffrement à clé publique à partir d'une telle fonction, sera une chose très simple où la procédure de chiffrement consiste simplement à appliquer la fonction au message clair. L'utilité d'utilisation d'une telle fonction, difficile à inverser si on ne connait pas la trappe, dans le domaine de la cryptographie, réside dans le fait de rendre difficile la détermination du message en clair à partir du message chiffré sans connaitre la clé secrète de déchiffrement. Cependant, la définition de ces fonctions particulières n'est pas aussi facile puisqu'elle s'appuie généralement sur des problèmes mathématiques réputés difficiles tel que le problème de la factorisation de grands nombres entiers. Mais, si quelqu'un trouve un jour le moyen de simplifier la résolution d'un de ces

problèmes, l'algorithme correspondant, c'est à dire construit autour de ce problème, finira par être abandonné.

Les systèmes asymétriques les plus connus sont [25]: RSA, ElGamal, Rabin, McEliece, Courbes elliptiques (ECC), etc.

c. Applications

- **Transmission sécurisée de la clé symétrique**

Pour résoudre le problème d'échange de la clé secrète utilisée lors d'un chiffrement symétrique, le chiffrement asymétrique a été envisagé comme solution. Cette dernière consiste à chiffrer la clé secrète en utilisant un mécanisme de chiffrement asymétrique assurant, ainsi, un partage sécurisé de cette clé et évitant son interception par une personne tierce non autorisée. Donc, le chiffrement asymétrique intervient dans la seule phase d'échange de la clé symétrique, qui sera utilisée par la suite pour le reste de l'échange d'informations.

- **Mécanismes d'identification**

Le problème qui se pose avec le mode de chiffrement asymétrique est celui dit d'identification. Il est dû au fait que la clé publique est distribuée à toutes les personnes, ainsi, lors de la réception puis du déchiffrement d'un message chiffré, on a aucun moyen de vérifier avec certitude son origine. Afin de résoudre ce problème, on utilise des mécanismes d'identification permettant de garantir la provenance des informations chiffrées. Ces mécanismes sont fondés sur le chiffrement asymétrique dont nous décrivons ses étapes à travers l'exemple suivant [26] **:**

Si Bob souhaite envoyer des données chiffrées à Alice en garantissant la provenance de celles-ci, et par conséquence, l'authentification de ses messages, le principe d'identification par chiffrement asymétrique sera la solution. Il se résume en la succession des étapes suivantes :

1) Alice crée une paire de clés asymétriques : clé privée K_{prA}, clé publique K_{puA} ;

2) Alice envoie sa clé publique à Bob pour qu'il puisse l'envoyer des données chiffrées par la suite ;

3) Lorsque Bob va envoyer des informations chiffrées à Alice, il procèdera à signer numériquement ces informations afin de garantir à Alice que celles-ci proviennent effectivement de lui :

3.1) Bob doit donc, avant d'envoyer ces données chiffrées, créer une paire de clés asymétriques : clé publique K_{puB}, clé privée K_{prB}

3.2) Bob envoie sa clé publique K_{puB} à Alice

3.3) Bob chiffre son message avec sa clé privée K_{prB}, ce qui représente la *signature numérique* du message par chiffrement, puis chiffre une seconde fois le message précédent avec la clé publique d'Alice K_{puA}. C'est la phase du chiffrement réel du message.

4) Alice reçoit le message chiffré de Bob, le déchiffre avec sa clé privée : K_{prA}. À ce stade le message n'est pas encore lisible car il a été chiffré deux fois de suite.

5) Alice déchiffre une seconde fois le message avec la clé publique de Bob : K_{puB}.

Si le message ainsi déchiffré par Alice sera un message lisible, alors on conclu qu'il provient effectivement de Bob, sinon l'identité de l'expéditeur a été altérée.

Ce qu'on remarque ici, c'est que la clé publique peut être utilisée soit pour chiffrer ou déchiffrer des données selon l'application réalisée, soit chiffrement ordinaire ou authentification d'identité.

3.3.2.3. La cryptographie hybride

La cryptographie hybride consiste, comme son nom l'indique, en une association des deux techniques de cryptage précédentes où on code tout d'abord les données avec une clé privée dite *clé de session*, ensuite cette clé est cryptée à l'aide d'une clé publique classique. Dans cette politique de cryptage le choix de crypter la clé d'une manière publique au lieu de crypter les messages est du au fait que, la clé est souvent de petite taille par rapport aux données à chiffrer, donc, elle consomme beaucoup moins de temps lors de son chiffrement par rapport aux données. C'est pourquoi ces dernières sont chiffrées de manière symétrique. Ensuite, il ne reste qu'à transmettre le package contenant les données cryptées avec une clé privée, cryptée de son tour avec une clé publique. Une fois le package sera reçu, le récepteur procède inversement. Tout d'abord, il déchiffre la clé chiffrée à l'aide de sa clé privée pour obtenir la clé de session, qui sera utilisée, par la suite, pour déchiffrer les données chiffrées. Ainsi, les performances seront améliorées en associant la rapidité des systèmes de chiffrement symétriques et la bonne sécurisation des systèmes de chiffrement asymétriques.

Les logiciels comme PGP et GnuPG reposent sur ce concept permettant de combiner les avantages des deux systèmes.

3.3.3. La cryptographie quantique

L'**Informatique quantique**, née de la rencontre de physiciens et de théoriciens de l'informatique, commence à jeter les bases d'un nouvel espace technique malgré qu'on est encore loin de réalisations de taille industrielle, mais ce domaine est suffisamment prometteur pour que l'on doive dès aujourd'hui s'y intéresser, car cela risque bien de bouleverser le paysage informatique d'ici 10 à 20 ans et avec lui quelques-unes de nos certitudes [27]. C'est ce que *Jacqueline Dousson* a écrit dans son article qui date de 1999.

3.3.3.1. Historique

L'idée d'utiliser des lois de la mécanique quantique dans le domaine de la sécurité, remonte au début des années 1970 lorsque *Stephen Wiesner* de l'université de Columbia, a écrit un rapport présentant des idées tout à fait nouvelles [28]. Ainsi, il a proposé d'utiliser la mécanique quantique pour coder des billets de banques dont l'infalsifiabilité serait garantie par le principe d'*incertitude d'Heisenberg*. De plus, il a proposé aussi d'utiliser la mécanique quantique pour construire un canal multiplexeur permettant d'entremêler deux messages d'une façon qu'on ne puisse en lire qu'un seul et qu'en le lisant, on rende l'autre illisible. Bien que les idées de *Wiesner* n'ont été publiées qu'en 1983 dans la revue *Sigact News*, elles ont incité et ont été la source d'inspiration de deux autres chercheurs en essayant de la concrétiser. C'était vers le milieu des années quatre vint, où les deux chercheurs américains, *Charles Bennett*, chercheur chez IBM, et *Gilles* Brassard, de l'université de Montréal, qui se sont rencontrés en 1979 sur une plage de Porto Rico [29], ont décidé d'aborder ce domaine. Ainsi, ils ont construit le premier prototype opérationnel, à travers lequel, ils ont montré qu'il était possible grâce au canal quantique, hautement sécurisé, de transmettre des clefs secrètes de plusieurs centaines de bits à une vitesse de 10 bit/s entre deux points distants de 32 cm [28], et ce même si ce canal est espionné tout au long de la transmission.

3.3.3.2. Rappels et principe

Tout d'abord, il est important de signaler que le fait de parler de « cryptographie quantique », relève d'un petit abus de langage. Il s'agit en fait d'échange quantique de clés. Pour se faire, des impulsions de photon individuel correspondant aux bits de la clé de chiffrement, sont transmises à travers une fibre optique en manipulant des variables physiques

des photons. Mais tant que ce système cryptographique assure une très bonne sécurisation de l'information échangée, pourquoi ne pas l'utiliser pour communiquer le message au lieu de la clé ? Celle-ci est la première question qui peut être posée. En fait, cela est dû aux deux raisons essentielles suivantes [30]:

- ✓ En cryptographie quantique, les bits de l'information communiquée sont générés aléatoirement. Ce qui fait de ce principe, le meilleur des principes de génération de clés, mais en aucun cas, il ne peut être utilisé pour communiquer des informations bien précises (messages).
- ✓ Il est toujours possible en cryptographie quantique de détecter l'espionnage d'une communication, mais il est aussi possible qu'une telle détection ne sera signalée qu'après l'interception des bits d'informations par l'espion. Cela ne représente pas un grand problème si l'information correspond à une clé aléatoire qui peut être jetée tout simplement, mais il l'est au cas où l'information correspond au message.

a. Rappels des propriétés quantiques d'un photon polarisé

Pour comprendre la cryptographie quantique, uniquement fondée sur les propriétés quantiques des photons polarisés, il faut tout d'abord connaître et comprendre ces propriétés que nous résumons à travers les points suivants [30]:

1) Un photon peut être polarisé selon un axe quelconque (Sachons que la polarisation d'un photon est la direction d'oscillation du champ électromagnétique qui lui est associé).

2) Un photon polarisé selon un axe d'angle 'a' passant dans un filtre polarisant d'axe 'b' possède une chance égale à $\cos^2(b-a)$ de passer le filtre polarisant. Donc :

- ✓ si le filtre est orienté précisément dans l'axe de polarisation du photon (b = a), le photon traversera certainement le filtre (prob = $\cos^2(b-a) = \cos^2(0) = 1$).
- ✓ si le filtre est orienté à 90° de l'axe de polarisation du photon (b = a+90), le photon sera certainement arrêté par le filtre (prob = $\cos^2(b-a) = \cos^2(90) = 0$).
- ✓ si le filtre est orienté à 45° de l'axe de polarisation du photon (b = a+45), le photon aura une chance sur deux de passer le filtre (prob = $\cos^2(b-a) = \cos^2(45) = 1/2$).

3) Les propriétés ci-dessus sont encore du domaine « classique ». Les propriétés purement quantiques utilisées par la cryptographie quantique sont :

- ✓ Quand la probabilité de passer le filtre est ni 0 ni 1, le passage d'un photon individuel à travers le filtre est fondamentalement imprévisible et indéterministe.
- ✓ On ne peut connaître l'axe de polarisation qu'en employant un filtre polarisant (ou plus généralement, en faisant une mesure dont le résultat est OUI ou NON). Sachons qu'il

n'existe pas de mesure directe, donnant un angle par exemple, de l'axe de polarisation du photon.

- ✓ On ne peut connaître l'axe de polarisation initial du photon que si l'axe du filtre est orienté précisément à 0° ou à 90° par rapport à celui du photon. Dans le cas où le filtre est transverse (45° par exemple), il n'y a fondamentalement aucun moyen de savoir quel était l'axe de polarisation initial du photon.

La figure I.8, [31]**,** présente un exemple relatif à la possibilité soit qu'un photon traverse le filtre ou pas, selon l'orientation de sa polarisation. Sachons qu'un filtre permet de distinguer entre les photons polarisés horizontalement (0°) et verticalement (90°) ; un autre entre les photons polarisés en diagonale (45°, 135°).

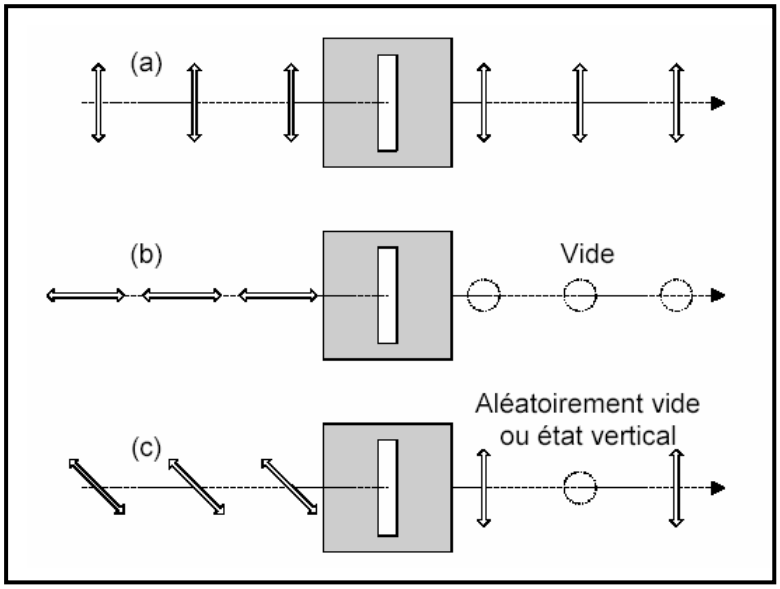


Figure I.8. *Photon unique traversant un filtre ne laissant passer que la lumière polarisée verticalement.*
***(a)** Les états polarisés verticalement traversent le filtre sans être absorbés.*
***(b)** Les états polarisés horizontalement sont tous absorbés.*
***(c)** Les états polarisés diagonalement sont aléatoirement absorbés ou transmis.*
Un observateur placé après le filtre ne pourrait donc pas, de manière déterministe, distinguer l'état du photon avant le filtre, par rapport à un état de polarisation soit verticale soit horizontale.

b. Principe

Tous les systèmes faisant partie de la cryptographie moderne, prennent pour acquis que les communications numériques peuvent être toujours espionnées d'une façon passive, c'est à dire sans détecter une modification de l'intégrité des données échangées ; ou

enregistrées par un tiers pour un usage ultérieur. Toutefois, de telles informations enregistrées peuvent paraître sans importance du fait qu'elles sont incompréhensibles. Mais quelqu'un qui espérerait découvrir à une date ultérieure la clé secrète, et après avoir accumulé suffisamment de textes chiffrés, pourra plus facilement mener sa cryptanalyse. Et s'il réussisse à révéler la clé secrète de chiffrement, il sera en mesure de décrypter tous les messages secrets accumulés.

La cryptographie quantique repose sur le *principe d'incertitude* d'*Heisenberg,* du nom du physicien qui l'a décrit en 1927, selon lequel la mesure d'un système quantique perturbe ce système. Donc, une oreille indiscrète sur un canal de transmission quantique engendre des perturbations inévitables qui alertent les utilisateurs légaux (*Alice* et *Bob*).
Imaginons que l'espion *Oscar* parvienne à placer des détecteurs sur le canal de communication quantique d'*Alice* et *Bob* [32]: qu'est-ce qui l'empêcherait d'intercepter les photons, de les analyser et d'en renvoyer une copie conforme, ni vu ni connu?
C'est ce même principe qui lui interdit de procéder à une copie. En effet, si l'on cherche à mesurer, par exemple, la vitesse d'une particule quantique avec la plus grande précision possible, on ne peut pas déterminer sa position avec la même précision; inversement, si vous voulez savoir où est le photon, il sera impossible de connaître sa vitesse. C'est donc la nature elle-même qui interdit de connaître, à chaque instant, la description complète de l'état quantique d'une particule. Donc, un espion ne pourra jamais copier des photons afin d'obtenir un double de la clef de chiffrement qu'ils transportent. Ainsi, il est possible de distribuer une clef secrète aléatoire à deux utilisateurs qui ne partagent initialement aucun secret, de façon sécurisée contre des espions même de puissance de calcul infinie. Une fois cette clef secrète sera reçue, elle peut être utilisée avec un système cryptographique classique. Donc, il nous faut deux canaux de communication, à savoir le canal quantique servant à l'échange des clés, et le canal classique permettant la transmission des données chiffrées.

Le grand avantage de cette technologie réside dans le fait qu'elle permet de changer de clés plusieurs fois par seconde [33]. A long terme, il n'y aura ainsi jamais d'ordinateur dont la puissance de calcul sera suffisamment performante pour casser ces clés. De plus, la cryptographie quantique ne nécessite aucune hypothèse, comme par exemple : factoriser est difficile, ainsi ses preuves de sécurité reposent simplement sur la correction des principes quantiques.

3.3.3.3. En pratique

L'émetteur émet la clé bit par bit, photon par photon à intervalle régulier, en choisissant aléatoirement le mode de codage ou plutôt le mode de polarisation (voir figure I.9) pour chaque photon émis ; tout en notant pour chaque bit le mode de polarisation choisi. Cette opération correspond à celle du chiffrement de la clé.

Les deux modes de polarisations possibles sont les suivants [30]:

- **Mode 1 :** "0" est codé par un photon d'axe de polarisation 0° et "1" par un photon de polarisation 90°.
- **Mode 2 :** "0" est codé par un photon d'axe de polarisation 45° et "1" par un photon de polarisation 135°.

Pour plus de clarté, ces deux modes peuvent être schématisés comme suit :

Figure I.9. *Les deux modes de polarisation.*
(a) *Mode 1*, **(b)** *Mode 2*

Lors de la réception des photons, le récepteur et après avoir positionné son filtre polarisant, aléatoirement aussi, à 0° ou à 45°, il note l'orientation choisie du filtre ainsi que le résultat qui peut être soit, le photon a passé le filtre, ou le photon n'a pas passé le filtre. Cette opération correspond au déchiffrement de la clé reçue.

Concernant le choix de la polarisation de chacun des bits, deux cas de figures sont possibles :

✓ l'émetteur et le récepteur ont choisi, par hasard, la même orientation de polarisation. Dans ce cas, le photon reçu est représentatif du bit émis, puisque en traversant le filtre correctement sa polarisation n'est pas affectée. Donc, il sera traduit directement en bit.

✓ l'émetteur et le récepteur ont choisi une orientation séparée de 45°. Dans ce cas, la polarisation du photon traversant ce faux filtre se transforme de façon aléatoire. Donc, le photon reçu ne contient aucune information.

La figure ci-dessous présente en schémas les deux cas de figures précédents.

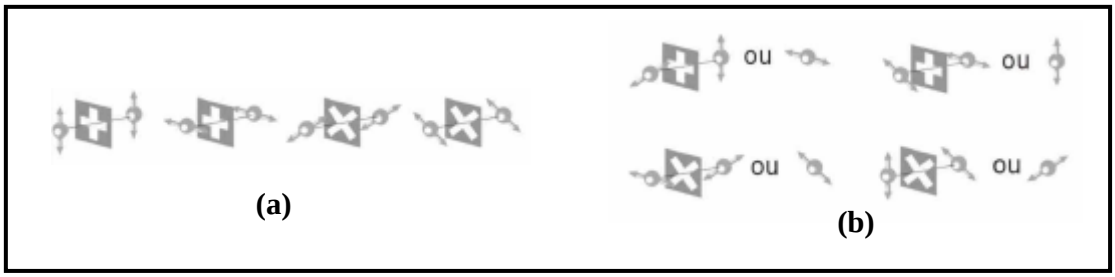


Figure I.10. *Les cas de polarisation.*
(a) *Polarisation non affectée,* **(b)** *Polarisation affectée*

Après la réception des 2*n bits correspondant à une clé de n bits utiles par le récepteur, l'émetteur lui transmis, non forcement à travers un canal de communication sûr, le mode de polarisation employé pour chaque bit. Ainsi, le récepteur est en mesure d'extraire, de manière certaine, les n bits formant la clé en moyenne de 2*n bits, pour lesquels l'orientation de polarisation a été la même. Mais reste de tester la sûreté de cette clé. Pour se faire, l'émetteur transmet les valeurs d'un certain nombre de bits pour lesquels les orientations choisies par l'émetteur et le récepteur sont les mêmes. Si au minimum, la valeur d'un de ces bits dits sacrifiés, du fait qu'ils sont transmis par la voie d'un canal non sûr, se diffère entre l'émetteur et le récepteur, la clé est jetée et le processus est recommencé.
Le processus général de polarisation, de transfert et enfin de filtrage peut être récapitulé sur la figure suivante [34]:

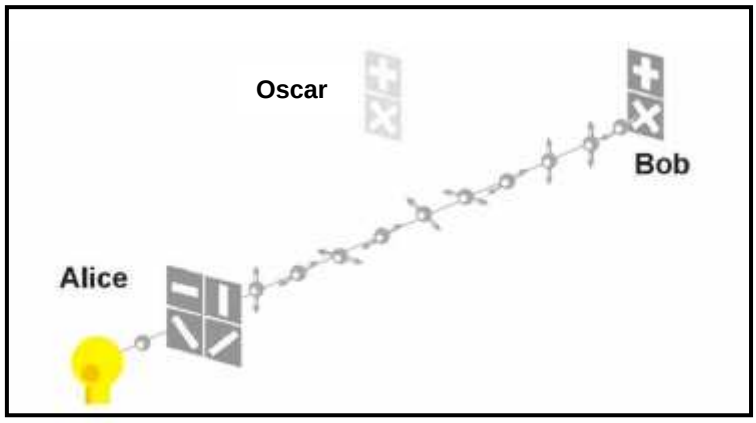


Figure I.11. *Principe général de la cryptographie quantique*

3.3.3.4. Difficultés

Aujourd'hui, les principaux désavantages qui alourdissent le développement de la cryptographie quantique sont spécialement dus à l'imperfection de l'appareil utilisé engendrant des problèmes de polarisation des photons, ainsi qu'à la difficulté d'amplification par la route des transmissions quantiques. Et sachant que les fibres optiques sont encore les mieux adaptées en cryptographie quantique, d'autres moyens de transfert sont envisagés. À titre d'exemple et en 2002, une équipe allemande a réussi à transmettre, grâce à un laser, un message crypté entre deux sommets alpins distants de 23 kilomètres, profitant ainsi d'un air pur en altitude [35]. D'ici à dix ans, l'Agence spatiale européenne compte même mettre en place plusieurs satellites expérimentaux en orbite basse qui communiqueraient avec la terre grâce à cette technique.

3.3.3.5. La cryptographie quantique: un sujet très interdisciplinaire

D'après notre brève présentation de la cryptographie quantique à travers les points précédemment vus, on a pu constater qu'elle rassemble plusieurs disciplines. Ainsi, on y trouve des questions de mathématiques et de physique fondamentale très abstraites, portant sur la mécanique quantique, des questions de recherche de performances alourdies par l'instrumentation optique utilisée (laser, détecteur, fibre optique), des questions liées à l'adaptation des résultats à des fins industrielles, sans parler des questions éthiques que posent la confidentialité dans le monde d'aujourd'hui [31]. En bref, c'est un sujet d'actualité qui recouvre un très large éventail de compétences, allant de la physique fondamentale jusqu'aux applications industrielles.

3.3.3.6. Applications

À ce stade, nous énumérons les principales applications pratiques faites dans le domaine de la cryptographie quantique [36].

- ✓ En 1989, la première expérience de transmission quantique de clés a été menée, en propagation libre, dans les laboratoires d'IBM sur une distance de 30 cm.
- ✓ En 1991, une équipe de chercheurs renommés, réussie une expérience pratique, où 715000 impulsions d'intensité moyenne de 0.12 photons par impulsion ont été transmis sur une distance de 32 cm pour obtenir une clef de 2000 bits contenant 4% d'erreurs de transmission. La correction d'erreurs a permis de les éliminer tous en ne révélant que 550 bits d'information à *Oscar*. En sacrifiant 1172 bits (550 bits compris), *Alice* et *Bob* ont échangé

une clef finale de 828 bits, avec un rythme de transmission de 1 bit/seconde, mais avec une probabilité inférieure à 10^{-6} qu'*Oscar* connaisse un bit de cette clef. Nous pouvons donc dire pour cette première expérience que l'intégrité est quasi-absolument vérifiée…mais que la rapidité de transmission est quelques peu à…améliorer !

✓ En 1993, les laboratoires de recherche de British Telecom ont effectué des expériences semblables, sur des fibres optiques d'une dizaine de kilomètres. C'est aujourd'hui sur des longueurs de 20 à 30 kilomètres que l'université de Genève, le laboratoire de recherche de Los Alamos ou le Laboratoire d'optique Pierre Michel DUFFIEUX mènent de semblables expériences.

✓ En 1994, *Peter SCHORR* des laboratoires Bell AT&T a trouvé un programme qui définissait une série d'étapes à suivre pour factoriser des nombres premiers, soit ce qu'il est nécessaire d'avoir pour casser RSA. Quand *Martin GARDNER* avait posé son énigme RSA dans la magasine *Scientific American*, il avait fallu plusieurs mois à plusieurs PC pour factoriser un nombre de 129 chiffres. Pour comparaison, le programme de *SCHORR*, s'il avait pu être implémenté sur un véritable ordinateur quantique, aurait permis de factoriser un nombre un million de fois plus grand en moins d'une minute…

✓ En 1995, des chercheurs de l'Université de Genève ont réussi à monter un système de cryptographie quantique qui fonctionne sur une fibre optique rejoignant Genève à Nyon (23 km).

✓ En 1996, *Lov GROVER* a développé un programme qui permet de scruter une liste à une grande vitesse, soit ce qu'il faut pour casser une clé DES. Un ordinateur classique capable de tester 1 million de clefs par minute mettrait 1000 ans quand un ordinateur quantique mettrait moins de quatre minutes avec l'algorithme de *GROVER*.

✓ Récemment, des scientifiques de Los Alamos ont réussi à transmettre des données dans l'air, mais sur une distance de 1 km seulement. Leur idée était de trouver un moyen de transmission via des satellites, ce qui permettrait d'entrevoir des applications plus commerciales.

3.3.3.7. Conclusion

Le domaine de la cryptographie quantique, et de l'informatique quantique de façon générale, qui parait à première vue difficile même très difficile, cache derrière une extraordinaire richesse et potentiel de création d'un ordinateur quantique qui, comme l'avançait *David Deutsch* [37], un ordinateur de 100 qubits nous permettrait de simuler un cerveau humain et un ordinateur de 300 qubits nous permettrait de simuler l'évolution de

l'univers depuis le Big Bang, sachons qu'actuellement seul un ordinateur de 7 qubits existe, construit par IBM en 2001. Nous sommes donc encore loin d'atteindre les objectifs exposés par *David Deutsch*.

Cependant, IBM, Stanford, Los Alamos, AT&T [27], ces quelques noms d'entreprises et d'équipes menant des recherches approfondies sur la cryptographie quantique, devraient suffire à nous convaincre que le sujet est pris très au sérieux par des intérêts qui ne sont pas qu'académiques. Seul le futur nous dira si cette nouvelle approche remplacera les systèmes utilisés actuellement. Mais, du moment où les réponses aux problèmes posés restent incertaines, cela permet encore de beaux jours aux AES, RSA et autres standards de chiffrement moderne.

En effet, l'informatique quantique n'a pas fini de nous étonner, et celui qui espérera trouver une faille dans la sécurité quantique n'a qu'à reconsidérer les lois de la physique quantique difficilement mise en place depuis une centaine d'années…

3.3.4. Analogies

Les deux modes de cryptage moderne, qui sont le cryptage symétrique et le cryptage asymétrique, peuvent être comparés à des moyens physiques d'échange de messages confidentiels. Ainsi, un système à clé secrète correspond à un coffre-fort, puisqu'avec ce dernier, et si Alice veut communiquer un message à Bob en le déposant dans ce coffre, ils doivent tout d'abord partager une information secrète qui est la combinaison indispensable pour toute opération de dépôt ou de récupération de documents. Donc, cette information doit être connue par Alice et Bob seulement. Du coté des systèmes à clé publique, ils correspondent à une boite aux lettres. Dans ce cas, toute personne souhaitant transmettre un document confidentiel à Bob n'a qu'à le mettre dans sa boite aux lettres qui ferme à clé. Seul Bob possédant cette clé peut ouvrir la boite et lire les documents qui lui sont destinés.

4. Exemples d'algorithmes de chiffrement

4.1. Algorithmes de chiffrement symétriques

A ce niveau, on va présenter, plus ou moins en détails, les plus fameux des algorithmes de chiffrement s'inscrivant sous ce mode.

4.1.1. Masque jetable (one-time pad)

Cet algorithme, inventé en 1917 et connu aussi sous le nom de **chiffre de Vernam**, est un algorithme de chiffrement prouvé inconditionnellement sûr. Dans ce système, la clé possède la même taille que le texte à chiffrer et est appelée *masque jetable*. « Masque », car cette clef est combinée par *ou exclusif* avec le texte en clair pour obtenir le texte chiffré ; « jetable », car une clef ne doit servir qu'une seule fois [38].

La sécurité de ce système repose sur la génération complètement aléatoire de la clé, ce qui représente le grand avantage de ce système. Par conséquence, si le cryptanalyste ne possède aucune information sur laquelle son attaque va appuyer, tous les masques seront équiprobables. En effet, si M est le message à chiffrer, C le message chiffré correspondant et K le masque jetable, nous avons :

$$C = M \oplus K$$

Supposons que le cryptanalyste connaisse C ; alors :

$$\forall M', \exists K' : C = M' \oplus K'$$

Donc, c'est : $K' = M' \oplus C \Rightarrow$ tous les messages en clair sont équiprobables

$\Rightarrow$ Impossible de savoir quel est le bon texte en clair sans connaître la clé.

Malgré cela, ce système est limité à des applications extrêmes et ne peut être utilisé pour chiffrer des flux importants de données à cause de la taille de la clé nécessitant des générateurs aléatoires pour sa création.

4.1.2. DES

Le 15 mai 1973, le *National Bureau of Standards* des Etats-Unis lança un appel d'offre de système cryptographique dans le *Federal Register*, qui est un journal officiel américain [14]. Cet appel déboucha sur le **Standard de chiffrement de données DES** qui est devenu le système cryptographique le plus utilisé dans le monde. IBM développa initialement DES comme modification d'un système antérieur appelé **LUCIFER.** DES, ou encore appelé **DEA** (**Data Encryption Algorithm**), fut publié dans le *Federal Register* le 17 mars 1975. Après un nombre considérable de débats publics, on adopta DES comme standard pour des applications non classifiées le 15 janvier 1977. Depuis son adoption, DES a été réévalué par le National Bureau of Standards tous les cinq ans, approximativement. La plus récente révision date de janvier 1994 où il a été renouvelé jusqu'en 1998. Il est prévu que le standard s'arrête à cette date.

4.1.2.1. Description

Une description complète de DES est donnée dans le *Federal Information Processing Standard Publication* (FIPS) № 46 du 15 janvier 1977 [14]. C'est le cryptosystème qui a été le plus utilisé, de plus, il a bien résisté aux efforts des cryptanalystes pendant 25 ans.

Ce cryptosystème est un système de chiffrement *par blocs*. Cela signifie, comme nous l'avons déjà présenté dans la section 3.3.2.1.a, qu'il ne chiffre pas les données caractère par caractère, mais il découpe le texte clair en blocs de 64 bits, dans le cas de ce cryptosystème. Ces blocs sont chiffrés séparément, puis concaténés. Ainsi, les données en entrée de cet algorithme seront des blocs de 64 bits du texte clair, et les données en sortie seront aussi des blocs de 64 bits de texte chiffré. C'est seulement la courte longueur de la clé, utilisée lors du chiffrement qui est de 56 bits, qui ne lui permet pas, aujourd'hui, d'assurer un bon niveau de sécurité, malgré qu'elle a été largement suffisante au moment de sa conception.

L'algorithme est relativement simple puisqu'il combine des permutations et des substitutions. On donne tout d'abord une description générale de ce système qui se déroule en trois étapes :

1) Etant donner un bloc de texte clair x. Une chaîne de bits x_0 est construite en changeant l'ordre des bits de x suivant une *permutation initiale* IP fixée [14]. On écrit :

$$x_0 = IP(x) = L_0R_0$$

Où L_0 contient les 32 premiers bits de la chaîne x_0 et R_0 contient les 32 bits restants.

2) 16 itérations (ou 16 tours) d'une certaine fonction sont effectuées. Chaque tour suit le même schéma qui consiste à prendre en entrée 32 bits : L_{i-1} et R_{i-1} du tour précédent et de produire de nouveau 32 bits L_i et R_i de la manière suivante [39]:

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

et

$$R_i = L_{i-1} \oplus f(R_{i-1}, k_i)$$

3) Après le dernier tour, les moitiés gauche et droite (L_{16} et R_{16}) sont échangées puis le texte sera permuté bit à bit par IP^{-1} pour obtenir le bloc de texte chiffré y. Plus formellement, y s'obtient comme suit [14]:

$$y = IP^{-1}(R_{16}L_{16}).$$

Cette description peut être résumée par le schéma général illustré par la figure suivante (figure I.12), où on a seulement représenté quelques-unes des 16 étapes.

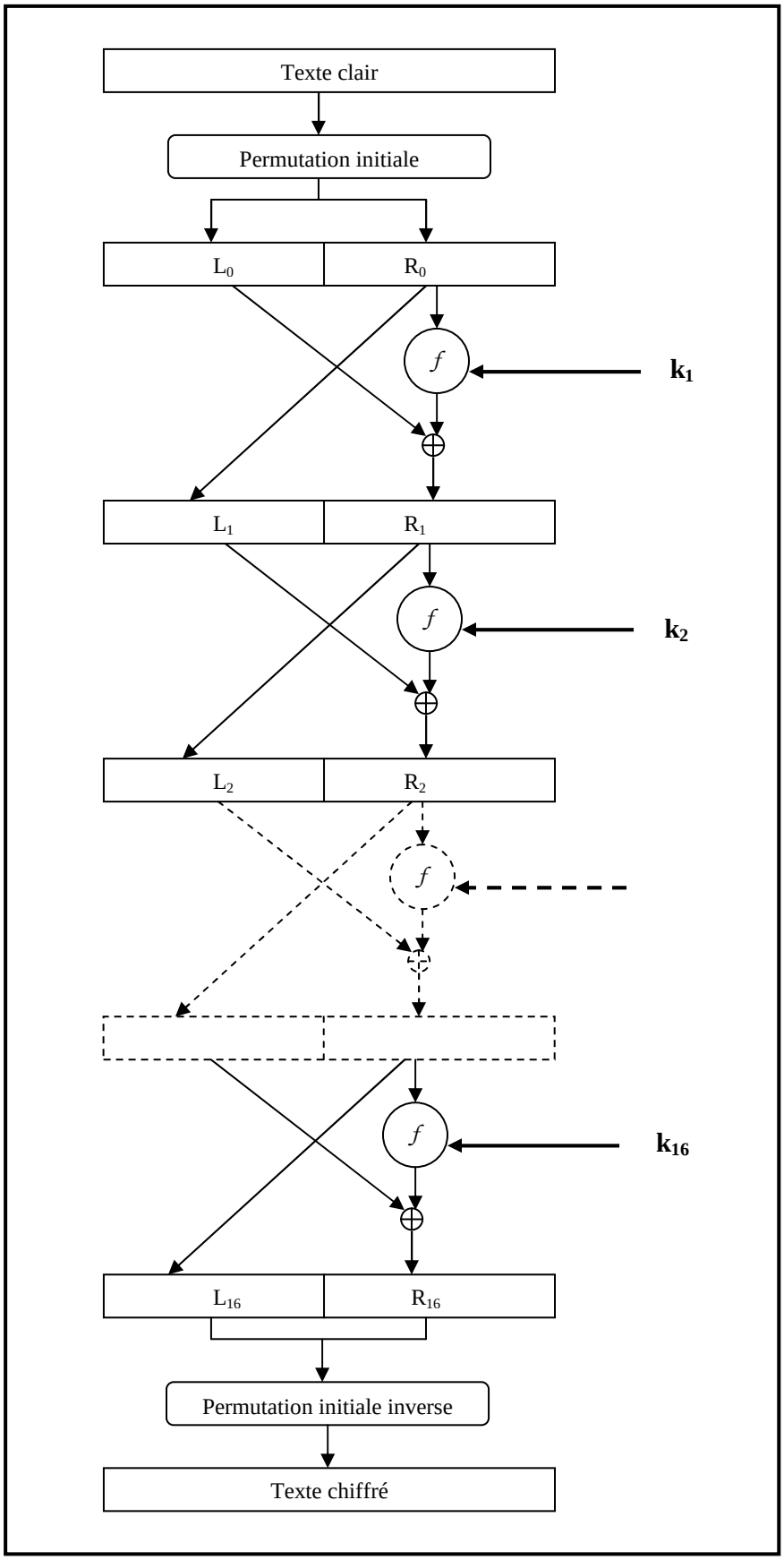


Figure I.12. *Schéma général de DES*

Nous présentons maintenant, plus ou moins en détail, chacune des étapes du système. Alors, la première et la dernière opération lors du chiffrement d'un bloc DES, qui sont l'application d'une permutation initiale et de son inverse à la fin, n'ont aucun rôle dans la sécurité de l'algorithme [40]. La figure I.13 résume ces deux phases où les deux suites de nombres, qui correspondent à la permutation initiale et à la permutation initiale inverse, se lisent de gauche à droite et de haut en bas.

Quand à leurs significations, par exemple, la permutation initiale (IP) signifie que le 58ème bit de la chaine à chiffrer, *x*, est le premier bit de IP(*x*), le 50ème bit de *x* est le deuxième bit de IP(*x*),… [14]. De façon générale, le n^{ème} nombre est la position avant permutation du bit qui se trouve en n^{ème} position après permutation.

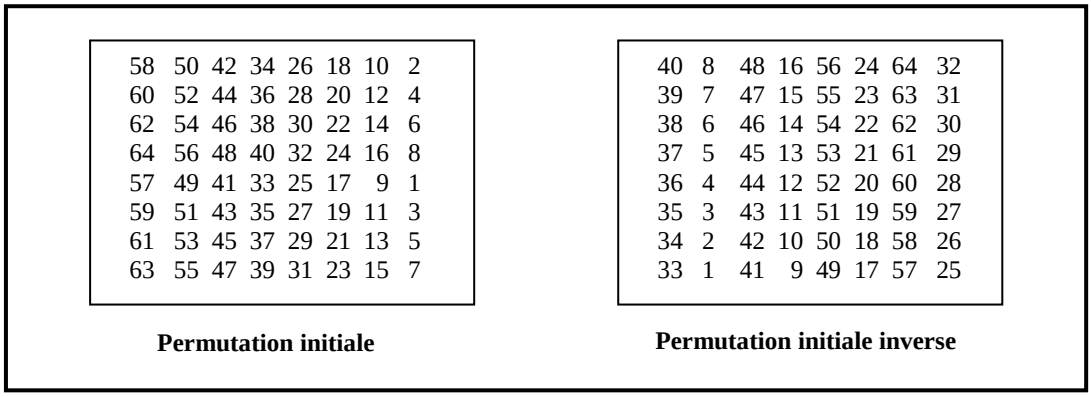


Figure I.13. *La permutation initiale et son inverse*

Après la permutation initiale, le bloc est divisé en deux sous-blocs de 32 bits, désignées par L_0 et R_0. À chacune des 16 itérations, on calcul deux nouveaux groupes de 32 bits L_i et R_i en fonction des deux groupes, notés L_{i-1} et R_{i-1}, de l'itération précédente. Pour cela, on utilise une clé intermédiaire k_i de 48 bits, obtenue par diversification à partir de la clé k de 56 bits, et on applique les formules suivantes :

$$L_i = R_{i-1} \quad \text{et} \quad R_i = L_{i-1} \oplus f(R_{i-1}, k_i)$$

D'après la formule qui corresponde au calcul de R_i, on constate que la fonction *f* utilise deux arguments ayant des tailles différentes : R_{i-1} de 32 bits et k_i de 48 bits. Ainsi, R_{i-1} sera expansé en 48 bits en redoublant, au hasard, 16 bits parmi les 32 bits initiaux. L'opération qui suit consiste à calculer un ou exclusif entre le résultat obtenu jusqu'à maintenant, codé sur 48 bits, et L_{i-1} codé sur 32 bits. Donc, le résultat final de *f* doit être codé sur 32 bits au lieu de 48 bits. Pour cela, la chaine de 48 = 8 × 6 bits sera transformée en une chaîne de 32 = 8 × 4 bits en utilisant des dispositifs appelés *boites-S*. Elles sont au nombre de huit, où chacune calcule

un bloc de 4 bits à partir d'un bloc de 6 bits. Enfin, on applique une permutation à ces 32 bits pour obtenir la valeur finale de f.

Cette succession d'opérations constituant la fonction f, peut être schématisée à travers la figure I.14 [40]**.**

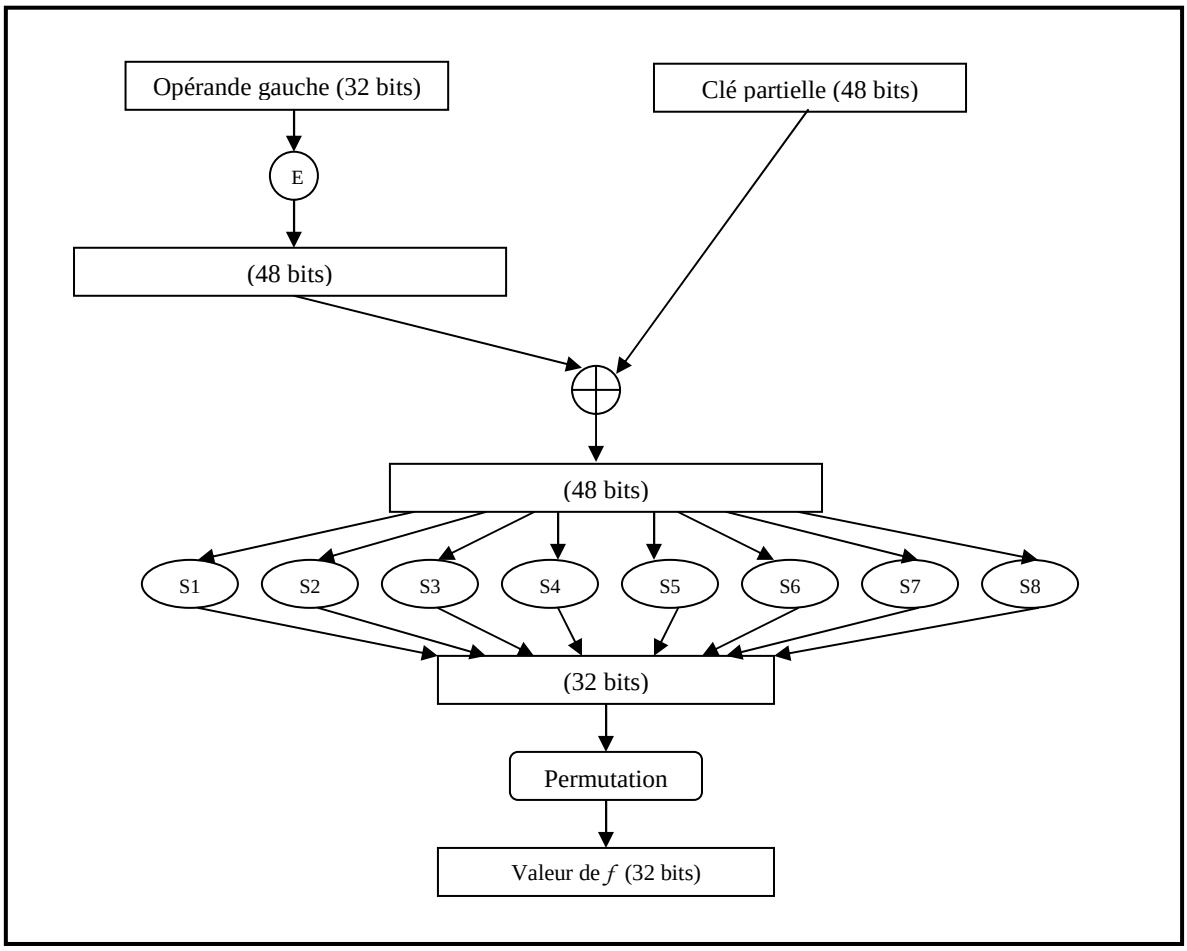


Figure I.14. *Schéma de la fonction f*

Remarque :

Le déchiffrement suit le même algorithme avec la même clef *K*. Seules les sous-clés sont appliquées dans le sens inverse.

4.1.2.2. Cryptanalyse de DES

a. Cryptanalyse différentielle

En 1990, *Eli Biham* et *Adi Shamir*, introduisent la méthode de ***cryptanalyse différentielle*** [41]. C'est grâce à cette méthode qu'ils ont pu trouver une attaque à texte clair

efficace contre le DES. Cette attaque cherche des paires de texte en clair et des paires de texte chiffré, puis elle les analyse en comparant les différences notables entre ces deux paires.
Ainsi, un DES à 8 ou à 10 tours peut facilement être cassé, mais le DES complet à 16 tours est resté hors de portée de cette attaque.

b. Cryptanalyse linéaire

La ***cryptanalyse linéaire*** a été introduite par *H.Gilbert* et *M. Matsui* dans le cas du DES [40]. C'est une attaque à messages clairs connus, qui utilise de légers défauts statistiques des étages de substitutions, correspondant aux boites-S dans le cas de DES. Elle n'est utilisable que pour un DES restreint à quelques tours, mais le DES réel n'est pas menacé par cette attaque.

c. Recherche exhaustive

Une autre attaque qui est celle par ***recherche exhaustive*** de la clé, est rendue envisageable vu l'accroissement de la puissance des ordinateurs, de leur nombre et des facilités de communication entre eux. Ainsi, les laboratoires RSA ont lancé en Janvier 1997 un défi consistant à décrypter par recherche exhaustive un message chiffré par DES pour démontrer que la taille des clés DES, qui est de 56 bits, a devenu insuffisante. Une équipe coordonnant des milliers d'ordinateurs du monde entier a abouti à la découverte de la bonne clé le 17 Juin 1997, après avoir exploré environ un quart de l'espace des clés [40]. Bien qu'une telle recherche demande des moyens considérables, elle a montré que le DES n'offre plus aujourd'hui une grande sécurité.

4.1.3. Triple DES (3DES)

Pour palier à l'insuffisance cryptographique observée du cryptosystème DES, due à la faible longueur de sa clé, il a été indispensable de chercher une solution rapide à cette situation. La première idée qui vienne à l'esprit est de combiner plusieurs chiffrements DES pour obtenir un système ayant une clé plus longue. Tout d'abord, une tentative consistant à combiner deux chiffrement DES a été essayée, mais il a vite paru qu'avec une attaque à message clair, dite « *par le milieu* », ce système sera remis en cause. Cette attaque s'appuie sur le message intermédiaire inconnu apparaissant entre les deux chiffrements DES successifs. Ainsi, elle construit la liste des messages intermédiaires possibles en chiffrant par DES un texte clair avec les 2^{56} clés possibles [40]. En déchiffrant par DES le chiffré correspondant avec des clés différentes, on obtient une autre liste de messages intermédiaires

possibles et le véritable message intermédiaire est dans l'intersection des deux listes. Le coût en mémoire de cette attaque est très important mais son coût en temps n'est pas significativement plus élevé que l'attaque exhaustive sur DES.

Ensuite, et en 1978, le **triple DES** (**3DES**) a été conçu par *Whitfield Diffie*, *Martin Hellman* et *Walt Tuchmann*. Il consiste à composer deux chiffrements DES de même clé séparée par un déchiffrement DES avec une autre clé. Donc, ce principe peut être formulé comme suit [40]:

$$\textbf{Triple-DES}_{k1,k2} = \textbf{DES}_{k1} \ \textbf{O} \ \textbf{DES}^{-1}_{k2} \ \textbf{O} \ \textbf{DES}_{k1}$$

On peut constater que le DES sera retrouvé comme cas particulier de la formule ci-dessus, lorsque $k_1 = k_2$. Le déchiffrement de son tour est formulé par [40]:

$$\textbf{Triple-DES}^{-1}_{k1,k2} = \textbf{DES}^{-1}_{k1} \ \textbf{O} \ \textbf{DES}_{k2} \ \textbf{O} \ \textbf{DES}^{-1}_{k1}$$

Cette méthode de chiffrement reste hors portée de l'attaque exhaustive vu la taille de la clé 3DES qui est composée de deux clés DES et donc composée de 112 bits. Une autre variante à trois clés DES différentes peut être conçue. D'une façon plus formelle, son principe peut être donné par la formule suivante :

$$\textbf{Triple-DES}_{k1,k2,k3} = \textbf{DES}_{k1} \ \textbf{O} \ \textbf{DES}_{k2} \ \textbf{O} \ \textbf{DES}_{k3}$$

Malgré cela, cette variante reste aussi fragile à une attaque de coût en 2^{112} s'appuyant sur l'un des deux messages intermédiaires.

4.1.4. Blowfish

Blowfish a été conçu par *Bruce Schneier* en 1993 comme étant une alternative aux algorithmes existants, en étant rapide et gratuit [42]. Ce cryptosystème est sensiblement plus rapide que le DES. La grandeur de ses blocs est de 64 bits et il peut prendre une longueur de clé variant entre 32 bits et 448 bits. Ainsi, et depuis sa conception, il a été grandement analysé et est aujourd'hui considéré comme étant un algorithme de chiffrement robuste, mais il n'est pas breveté. Ainsi, son utilisation est libre et gratuite.

4.1.5. AES

DES a très longtemps profité du soutien politique des USA. Par exemple, *Robert S. Litt* (Principal Associate Deputy Attorney General), a assuré le 17 mars 1998, que le FBI n'avait aucune possibilité technologique et financière de décoder un message codé avec un algorithme symétrique dont la clef secrète a une longueur égale à 56 bits [39]. Et pour compléter sa démonstration, il a déclaré aussi que 14000 PC Pentium durant 4 mois seraient nécessaires pour réaliser cela. C'est d'ailleurs, ce que *Louis J. Freeh* (Directeur du FBI) et

William P. Crowell (Deputy Director de la NSA) ont déclaré de leur tour. Malgré cela, pas mal d'attaques contre le DES ont été développées. Ce qui fait que, conscient des risques concernant DES, le NIST (National Institute of Standards and Technology) a demandé à la communauté cryptographique de réfléchir au successeur : **AES**.

AES est le sigle d'**Advanced Encryption Standard** (en français, standard de chiffrement avancé). C'est l'algorithme **Rijndael**, du nom de leurs concepteurs Belges *Joan Daemen* et *Vincent Rijmen* [43]. Il a été retenu par le NIST en octobre 2000 pour être l'algorithme AES, le nouveau standard de chiffrement pour les organisations du gouvernement des Etats-Unis, et ce, principalement pour des raisons de sécurité, performance, efficacité, facilité d'implémentation et flexibilité. De plus, son utilisation est très pratique car il consomme peu de mémoire.

Plus précisément, ce cryptosystème est issu d'un appel d'offre international lancé en janvier 1997 et ayant reçu 15 propositions en première ronde [44] : LOKI97 (Australia, Australian Defence Force Academy, *L.Brown, J.Pieprzyk, J.Seberry*), RIJNDAEL (Belgique, *J. Daemen, V. Rijmen*), CAST-256 (Canada, Entrust Technologies), DEAL (Canada, Outerbridge, Knudsen), FROG (Costa Rica, TecApro International S.A), DFC (France, Centre National pour la Recherche Scientifique), MAGENTA (Allemagne, Deutsche Telekom AG), E2 (Japon, Nippon Telegraph and Telephone Corporation), CRYPTON (Corée, Future Systems), HPC (Etats-Unis, Université Arizona, *Rich Schroeppel*), MARS (Etats-Unis, IBM), RC6 (Etats-Unis, RSA Laboratories), SAFER+ (Etats-Unis, Cylink Corporation), TWOFISH (Etats-Unis, *B. Schneier, J.Kelsey, D.Whiting, D.Wagner, C.Hall, N.Ferguson*), SERPENT (Grande-Bretagne, Israël, Norvège, *R. Anderson, Eli Biham, L. Knudsen*). Ces algorithmes ont été évalués par des experts, avec forum de discussion sur Internet, et organisation de conférences. Pour la même raison, 600 CD-ROM portant ces algorithmes ont été distribués dans plus de 50 pays en décembre 1998. Après la deuxième ronde qui a débuté le 22 Mars 1999 lors d'une conférence à Rome, Rijndael a été déclaré vainqueur par le NIST le 2 octobre 2000 sur les 5 candidats finalistes : MARS, RC6, Rijndael, Serpent et TwoFish. Toutefois, il est important de signaler que sur chacun des tests effectués, Rijndael n'est jamais sorti vainqueur [43]. Il s'est distingué à chaque fois par des performances intéressantes et donc au final pour sa polyvalence. Les résultats des votes étaient comme suit [42]:

- ✓ Rijndael : 86 votes,
- ✓ Serpent : 59 votes,

- ✓ Twofish : 31 votes,
- ✓ RC6 : 23 votes,
- ✓ MARS : 13 votes

Malgré cela, la NSA a annoncé que tous ces finalistes pouvaient être considérés comme sûrs et qu'ils étaient suffisamment robustes pour chiffrer les données non-classifiées du gouvernement américain.

Donc, le terme d'AES remplace désormais celui de Rijndael sans que l'algorithme ne soit modifié. Mais en réalité, AES est un sous-ensemble de Rijndael, puisque ce dernier offre des tailles de blocs et de clefs qui sont des multiples de 32 compris entre 128 et 256 bits, tandis que AES travaille avec des blocs de 128 bits seulement. Cette longueur de blocs, d'un coté, et la longueur des clés utilisées avec ce même cryptosystème (AES), qui peut être de 128,192 ou de 256 bits, d'un autre coté, ont été jugées suffisantes en juin 2003 par le gouvernement américain, et ce, pour protéger des documents classifiés jusqu'au niveau « Secret », alors que le niveau « Top-secret » nécessite des clés de 192 ou 256 bits [42].

4.1.5.1. Description

L'AES procède par blocs de 128 bits, avec une clé de 128, 192 ou 256 bits. Chaque bloc subit une séquence de transformations que nous résumons à travers les points suivants :

1) Addition de la clé secrète et du bloc en question avec un ou exclusif.

2) Les 128 bits sont répartis en 16 blocs de 8 bits (16 octets), qui sont ensuite placés dans une matrice de 4×4 après leur permutation selon une table définie au préalable. Les lignes de cette matrice sont soumises à une rotation vers la droite où l'incrément pour la rotation varie selon le numéro de la ligne.

3) Chaque colonne est transformée par combinaisons linéaires des différents éléments de la colonne. Cela revient à multiplier la matrice 4×4 par une autre matrice 4×4.

4) Une clé dite *de tour* est générée à partir de la clé secrète par un sous-algorithme dit *de cadencement*. Cette clé de tour est ajoutée par un ou exclusif au dernier bloc obtenu.

Ces différentes opérations, définissant un *tour*, sont répétées plusieurs fois. Pour une clé de 128, 192 ou 256, AES nécessite respectivement 10, 12 ou 14 tours. La figure suivante résume le principe de fonctionnement de cet algorithme de chiffrement.

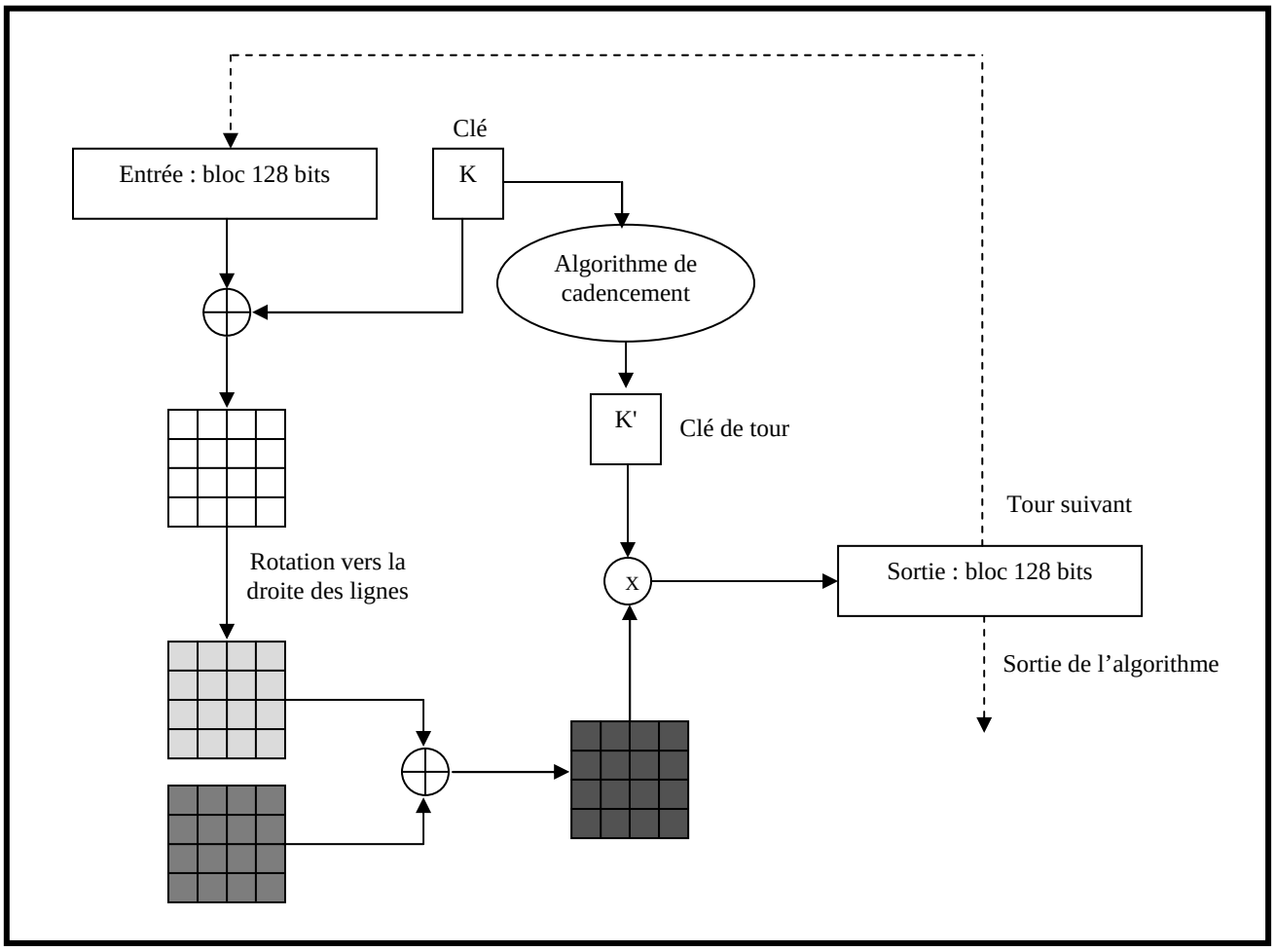


Figure I.15. *Schéma général de l'AES*

4.1.5.2. Cryptanalyse de l'AES

L'AES n'a pour l'instant pas été cassé et la recherche exhaustive demeure la seule solution.

a. Attaques sur des versions simplifiées

Niels Ferguson et son équipe ont proposé en 2000 une attaque sur une version à 7 tours de l'AES 128 bits [45]. Une attaque similaire casse un AES de 192 ou 256 bits contenant 8 tours. Un AES de 256 bits peut être cassé s'il est réduit à 9 tours avec une contrainte supplémentaire. En effet, cette dernière attaque repose sur le principe des « related-keys » (clés apparentées). Dans une telle attaque, la clé demeure secrète mais l'attaquant peut spécifier des transformations sur la clé et chiffrer des textes à sa guise. Il peut donc légèrement modifier la clé et regarder comment la sortie de l'AES se comporte.

b. Attaques sur la version complète

Certains groupes ont affirmé avoir cassé l'AES complet, mais après vérification par la communauté scientifique, il s'avérait que toutes ces méthodes étaient erronées. Cependant, plusieurs chercheurs ont mis en évidence des possibilités d'attaques algébriques, notamment l'attaque XL et une version améliorée, la XSL [45]. Ces attaques ont été le sujet de nombreuses discussions et leur efficacité n'a pas encore été pleinement démontrée, le XSL fait appel à une analyse heuristique dont la réussite n'est pas systématique. De plus, elles sont impraticables car le XSL demande au moins 2^{87} opérations voire 2^{100} dans certains cas. Le principe est d'établir les équations (quadratiques / booléennes) qui lient les entrées aux sorties et de résoudre ce système qui ne comporte pas moins de 8000 inconnues et 1600 équations pour 128 bits. La solution de ce système reste pour l'instant impossible à déterminer. En l'absence d'une preuve formelle sur l'efficacité d'attaques similaires au XSL, l'AES est donc considéré comme sûr.

4.1.6. Serpent

Serpent, inventé par *Ross Anderson*, *Eli Biham* et *Lars Knudsen*, est un cryptosystème symétrique chiffrant des blocs de 128 bits. Il a été développé en vu d'être un Advanced Encryption Standard. Et malgré que le choix du NIST pour AES s'est porté sur Rijndael, mais ça n'empêche de signaler que Serpent et Rijndael sont similaires, et que la principale différence entre eux, est que Rijndael est plus rapide mais Serpent est plus sûr. De plus, aucune attaque connue n'a réussie à casser cet algorithme.

4.1.7. Twofish

Twofish est un algorithme de chiffrement symétrique par bloc inventé et analysé par *Bruce Schneier*, *Niels Ferguson*, *John Kelsey*, *Doug Whiting*, *David Wagner* et *Chris Hall* au sein du Counterpane Labs, pour participer au concours AES, où, il a été l'un des cinq finalistes du concours sans pour autant être sélectionné pour le standard. Ce cryptosystème est conçu pour être très sûr et très flexible, en chiffrant des blocs de 128 bits avec une clé de 128, 192 ou 256 bits, et en reprenant quelques concepts présents dans le Blowfish du même auteur. Cependant, Twofish est légèrement plus lent que Rijndael mais plus rapide que les autres finalistes de l'AES.

En 2005, Counterpane Labs a passé un long temps en évaluant Twofish, sans pouvoir trouver d'attaques possibles sur la version complète de Twofish, qui semble être plus sûre que

la version initialement annoncée durant le concours AES. Ainsi, la recherche exhaustive reste le seul moyen pour le casser. Malgré ça, il reste relativement peu utilisé.

4.1.8. MARS

MARS est un algorithme de chiffrement symétrique par blocs créé par IBM comme algorithme pour le standard AES. *Don Coppersmith* était l'un des concepteurs de cet algorithme, qui prend en charge des blocs de 128 bits et des clés de dimensions variables entre 128 et 448 bits par incréments de 32 bits. Cet algorithme est unique, car il associe toutes les techniques de cryptage connues dans un seul produit. Ainsi, il utilise deux algorithmes séparés, de façon que si une partie de MARS est cassée, le reste des chiffres restera sécurisé et les données seront sauvegardées. De plus, MARS offre une meilleure sécurité que le triple DES et il est plus rapide que le DES.

4.1.9. RC6

RC6 est un algorithme de chiffrement par bloc publié en 1998, et conçu au sein de la société RSA Security par *Ron Rivest, Matt Robshaw, Ray Sidney* et *Yiqun Lisa Yin* dans le cadre du concours AES, où il parvient à atteindre la finale aux cotés de quatre autres systèmes de chiffrement. Il est basé sur un bloc de 128 bits et supporte des clés de 128, 192 et 256 bits.

4.1.10. Conclusion

A l'heure actuelle l'utilisation du DES est simplement déconseillée à cause de la grande puissance de calcul assurée par les ordinateurs les plus récents. Toutefois le triple DES, permet d'apporter un niveau de sécurité acceptable et de résister aux attaques les plus classiques. Le choix de l'AES reste néanmoins le meilleur choix, dans l'attente d'un remplaçant ou d'une méthode d'attaque efficace qui va le remettre en cause.

4.2. Algorithmes de chiffrement asymétriques

Plusieurs systèmes à clé publique ont été proposés. Leur sécurité repose sur divers problèmes calculatoire. Les plus connus sont les suivant :

4.2.1. RSA

En cryptographie à clé publique, les trois lettres **RSA** sont certainement les plus célèbres. Ce cryptosystème tire son nom des noms de ses trois inventeurs : *R. Rivest, A. Shamir,* et *L. Adleman*. Ce système, inventé en 1977, est le premier protocole de

cryptographie à clé publique, comme c'était déjà mentionné dans la section 3.3.2.2, présentant la cryptographie asymétrique. Il a été breveté par le MIT en 1983 aux États-Unis d'Amérique, mais le brevet a expiré le 21 septembre 2000 [46].

4.2.1.1. Description

Ce chiffrement est fondé sur la difficulté de factoriser un nombre qui est le produit de deux grands nombres premiers. Il utilise l'arithmétique de Z_n, qui est un anneau pour tout entier n supérieur à 2, et où n est le produit de deux nombres premiers impairs distincts p et q. Pour un tel n, on a [14]:

$$\emptyset(n) = (p-1)(q-1).$$

La description formelle du système est donnée dans la figure I.16.

Soit $n = pq$ où : p et q sont premiers, et soit $P = C = Z_n$.

On définit :

$K = \{(n, p, q, a, b) : n = pq, p \text{ et } q \text{ premiers} ; ab \equiv 1 \bmod \emptyset(n)\}$

Pour k = (n, p, q, a, b), on définit :

$$\mathbf{e_k(x) = x^b \bmod n}$$

$$\mathbf{d_k(y) = y^a \bmod n}$$

$(x, y) \in Z_n$. Les valeurs n et b sont publiques, et les valeurs p, q et a sont secrètes.

Figure I.16. *Le chiffrement RSA*

Pour s'assurer, ou pour démonter que le déchiffrement de y, qui est le texte chiffré, va bien donner le x, qui est le texte en clair, nous présentons la preuve suivante.

Preuve : [14] On a : $ab \equiv 1 \bmod \emptyset(n)$

Donc : $ab = t\,\emptyset(n) + 1$ avec: t un entier.

Alors : $y^a = (x^b)^a \equiv x^{t\,\emptyset(n)+1} \bmod n$.

$\equiv (x^{\emptyset(n)})^t \; x \bmod n$

$\equiv 1^t \; x \bmod n$

$\equiv x \bmod n$

La sécurité de RSA est basée sur l'hypothèse que la fonction $e_k(x) = x^b \bmod n$ est a sens unique, ce qui rend impossible à Oscar de décrypter un texte chiffré. Mais, Bob et à l'aide de la trappe qu'il garde secrète, qui est la factorisation $n = pq$, est le seul qui est en mesure d'accomplir l'opération de déchiffrement. Donc, il peut calculer $Ø(n) = (p-1)(q-1)$ et calculer l'exposant de déchiffrement a en utilisant l'algorithme d'Euclide étendu.

Il y a beaucoup de problèmes autour du chiffrement RSA à résoudre c'est pourquoi, il faut expliquer comment procéder efficacement au chiffrement et au déchiffrement. Ainsi, et pour mettre en œuvre le chiffrement RSA, Bob suit les étapes indiquées ci-dessous [14]:

1) Bob engendre deux grands nombres premiers p et q.

2) Bob calcul $n = pq$ et $Ø(n) = (p-1)(q-1)$.

3) Bob choisis un b aléatoire ($1< b< Ø(n)$), tel que $\text{pgcd}(b, Ø(n)) = 1$.

4) Bob calcul $a = b^{-1}$ en utilisant l'algorithme d'Euclide.

5) Bob publie b et n dans un répertoire.

Une attaque évidente à ce système consiste à tenter de factoriser n, alors que, l'intérêt du système RSA repose sur le fait, qu'à l'heure actuelle, il est pratiquement impossible de retrouver dans un temps raisonnable p et q à partir de n si celui-ci est très grand. Donc, Bob est le seul qui peut calculer a dans un temps court, sans que cela nécessite la transmission des entiers p et q, ce qui empêche leur piratage. Mais, si une méthode de factorisation rapide sera développée, ce système serait aussitôt périmé. Ainsi, la sécurité de RSA semble satisfaisante malgré qu'il ne soit pas prouvé mathématiquement qu'on ne puisse pas le casser. En effet, en augmentant constamment la taille des clés, ce système reste très fiable si ses utilisateurs suivent les conseils des spécialistes, qui peuvent porter sur la taille des clés, la forme des nombres employés ou sur les méthodes d'implémentation. De même, le bon choix de p et q est aussi, un point crucial assurant la bonne sécurisation de ce cryptosystème.

- **Choix de p et q :**

Le choix de p et q affecte grandement le niveau de sécurité de RSA. Pour cela, il faut évidemment se prémunir contre les algorithmes de factorisation dont la complexité dépend essentiellement de la taille du plus petit facteur premier de n [47], donc si possible choisir p et q de même taille. Il ne faut cependant pas les choisir trop proches, car alors une attaque exhaustive est possible: on suppose $q=p+k$ avec k un petit entier, et comme on connaît $n = pq$, on est ramené à résoudre une équation du second degré pour chaque valeur de k.

De plus, on impose usuellement que *p* et *q* soient des nombres premiers forts. Sachant, qu'un nombre premier *p* est dit *fort* lorsque :

a) p - 1 a un grand facteur premier r,

b) p + 1 a un grand facteur premier,

c) r - 1 a un grand facteur premier.

La condition (a) permet de se prémunir contre la factorisation de *p* par l'algorithme P-1 de *Pollard*, la condition (b) permet de se prémunir contre la factorisation de *p* par l'algorithme P+1, attribué à *Williams*. Enfin, la condition (c) permet de se prémunir contre les attaques cycliques.

4.2.1.2. Cryptanalyse de RSA

Depuis son apparition, plusieurs attaques ont été découvertes contre le RSA. Et même si aucune de ces attaques n'est réellement destructive, elles démontrent toutefois qu'il faut implémenter RSA avec beaucoup de précautions. Elles illustrent également la difficulté à définir des notions de sécurité convenables, en chiffrement asymétrique.

Phong Nguyen dans [15] avait groupé les attaques possibles en quatre grandes catégories :

- ✓ Attaques élémentaires,
- ✓ Attaques sur les implémentations de RSA,
- ✓ Attaques simples de RSA à petit exposant,
- ✓ Attaque par géométrie des nombres.

Dans ce qui suit, nous citons quelques exemples d'attaques tout en mentionnant la catégorie correspondante.

a. Recherche exhaustive : (1ère catégorie) **:**

Comme la fonction de chiffrement RSA : $m \longrightarrow m^b \bmod n$ est déterministe, c-à-d un message est toujours chiffré en le même chiffré, donc et si l'ensemble des messages possibles est connu et de petite taille, il sera facile de décrypter par une recherche exhaustive [15].

Pour éviter cette attaque, il est indispensable de randomiser les messages avant chiffrement. De ce fait, on aura une fonction de chiffrement probabiliste mieux que d'être déterministe. Cela revient à faire des transformations sur la fonction à sens unique à trappe avant d'être utilisée en chiffrement.

b. Attaque par chronométrage : ($2^{ème}$ catégorie) **:**

Considérons une carte à puce contenant une clef privée RSA. Normalement, un attaquant mettant la main sur la carte ne peut arriver à déterminer cette clef privée, en raison de protections physiques. *Kocher* a néanmoins démontré qu'en mesurant précisément le temps pris par la carte pour effectuer un déchiffrement RSA, un attaquant pouvait rapidement déterminer l'exposant privé *a* pour les implémentations usuelles du RSA [15].

4.2.2. Chiffrement d'ElGamal

L'algorithme **ElGamal,** créé par *Taher Elgamal*, est un algorithme de cryptographie asymétrique basé sur les logarithmes discrets. Cet algorithme est utilisé par de récentes versions de PGP, d'autres systèmes de chiffrement et même par le DSA (Digital Signature Algorithm**)**, qui est un algorithme de signature numérique standardisé par le NIST aux États Unis. Ainsi, il peut être utilisé pour le chiffrement et la signature électronique, rappelons que la signature est l'ensemble des mécanismes permettant d'assurer au destinataire que le message envoyé a bien été rédigé par l'émetteur légal. De plus, et contrairement à RSA, cet algorithme n'a jamais été sous la protection d'un brevet.

4.2.2.1. Problème du logarithme discret

Le problème du logarithme discret est décrit dans le corps fini Z_p, où *p* est un nombre premier, sachons que le groupe Z_p^* est cyclique, et que ses générateurs sont appelés racines primitives modulo *p* [14]. La figure suivante résume le problème du logarithme discret.

Instance du problème : I = (p, α, ß) où p est premier, $\alpha \in Z_p$ est primitif et $ß \in Z_p^*$.

Question : Trouver l'unique a, $0 \leq a \leq p-2$ tel que :

$$\alpha^a \equiv ß \pmod{p}$$

On note cet entier $\log_\alpha ß$

Figure I.17. *Problème du logarithme discret dans Z_p*

Le problème du logarithme discret dans Z_p, faisant l'objet de nombreuses études, est réputé difficile. Sachons qu'aucun algorithme polynomial n'a été défini pour le résoudre. Cependant, et pour éviter les attaques connues, *p* doit être convenablement choisi, et $p-1$ doit avoir un grand facteur premier.

L'utilité de ce problème en cryptographie provient du fait que calculer des logarithmes discrets est certainement difficile, tandis que calculer l'opération inverse d'exponentiation peut se faire efficacement avec l'algorithme d'exponentiation modulaire [14]. En d'autres termes, l'exponentiation modulo *p* est une fonction à sens unique pour des nombres premiers *p* convenables.

4.2.2.2. Description

La figure suivante présente, d'une manière formelle, l'algorithme ElGamal [14].

Soit *p* un nombre premier tel que le problème du logarithme discret dans Z_p soit difficile, et soit $\alpha \in Z_p^*$ un élément primitif. Soit $P = Z_p^*$. $C = Z_p^* \times Z_p^*$ et

$$K = \{(p, \alpha, a, ß) : ß \equiv \alpha^a \pmod{p}\}$$

Les valeurs *p*, α et ß sont publiques, et *a* est secret.

Pour $k = (p, \alpha, a, ß)$, et pour un $k \in Z_{p-1}$ aléatoire (secret), la fonction de chiffrement est définit par :

$$\mathbf{e_k(x, k) = (y_1, y_2)}$$

Où

$$\mathbf{y_1 = \alpha^k \bmod p}$$

Et

$$\mathbf{y_2 = x\, ß^k \bmod p}$$

Pour $y_1, y_2 \in Z_p^*$, la fonction de déchiffrement est définit comme suit :

$$\mathbf{d_k(y_1, y_2) = y_2(y_1^a)^{-1} \bmod p}$$

Figure I.18. *Chiffrement d'ElGamal*

Informellement, le fonctionnement du chiffrement ElGamal peut être décrit par la suite des points suivants :

- ✓ Le texte clair est masqué par la multiplication par $ß^k$, en produisant y_2 ;
- ✓ la valeur α^k est également transmise en tant que partie du texte chiffré ;
- ✓ Bob, qui connaît l'exposant secret *a*, peut calculer $ß^k$ à partir de α^k. Il peut alors « enlever le masque » en divisant y_2 par $ß^k$ et obtenir le texte clair x.

D'après la description, soit formelle ou informelle, de cet algorithme, il est clair que le chiffrement d'ElGamal est non déterministe, ou encore dit probabiliste, du moment où, l'opération de chiffrement dépend du texte clair, x, et d'une valeur aléatoire, k, choisie par Alice. Donc, plusieurs textes chiffrés peuvent correspondre à un même texte clair.

4.2.2.3. Cryptanalyse d'ElGamal

Une attaque possible à ce cryptosystème est celle dite ***man in the middle***. Son principe dans le cas d'ElGamal fonctionnant avec le mode *Diffie-hellman* est illustré sur la figure I.19.

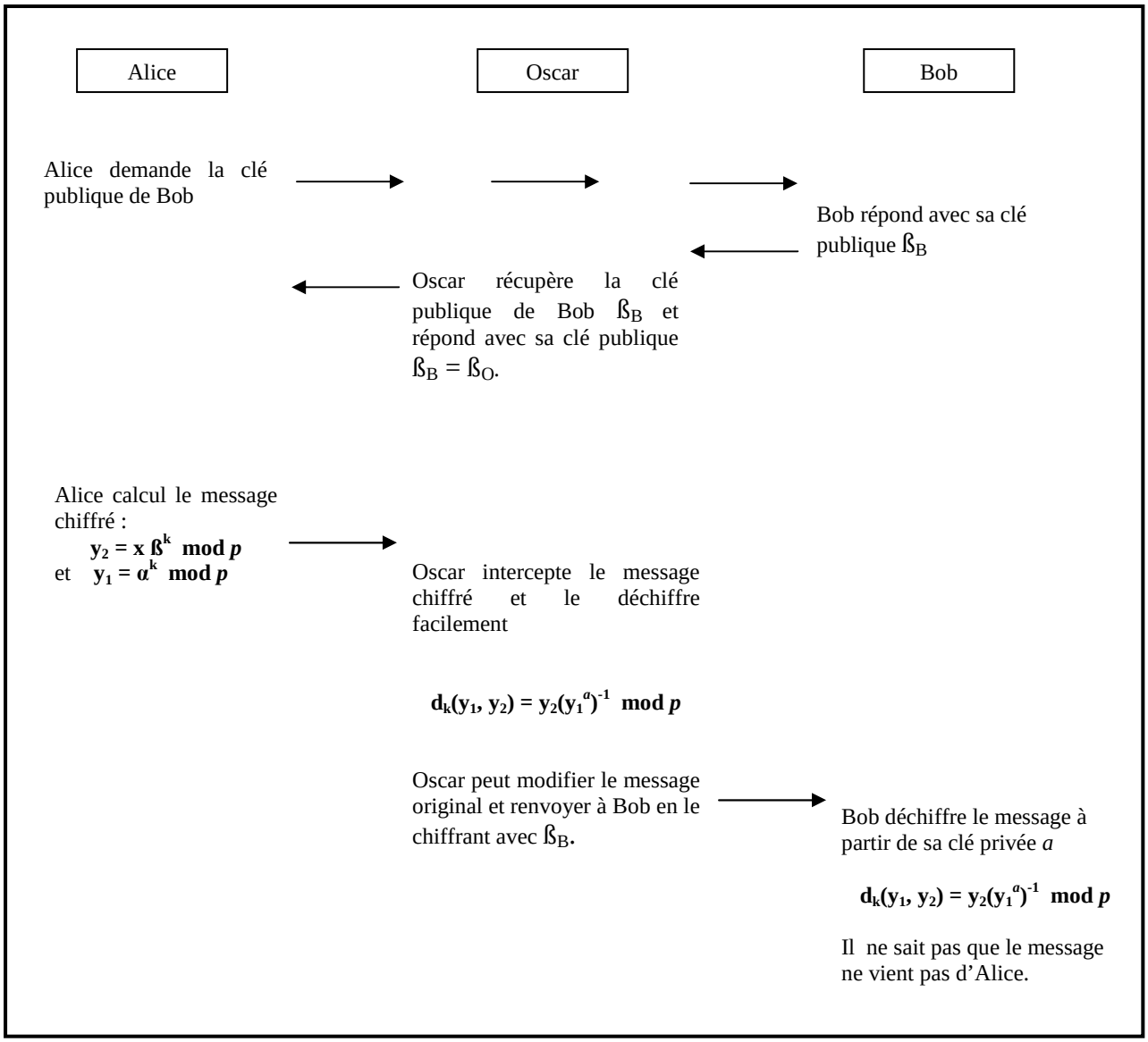


Figure I.19. *Principe de l'attaque « man in the middle »*

Pour contrer l'attaque, il faut être sûr de la provenance de la clef $ß_B$. Ainsi, deux solutions sont envisageables [48]. La première consiste à signer le message ; la deuxième solution propose d'avoir un recours à un organisme tiers pour certifier la clé.

4.2.3. Conclusion

Les calculs faits en 1995 ont ouvris un vaste horizon devant le chiffre RSA, du fait que le cassage des clés de 130 chiffres, utilisées à l'époque, nécessite 150 ans. Alors que va-t-on dire avec les clés utilisées aujourd'hui, qui comportent plus de 300 chiffres et qui sont donc plusieurs milliards de fois supérieures ? Donc, la méthode est officiellement sûre si l'on respecte certaines contraintes de longueur de clés et d'usage. Toutefois, personne depuis 2500 ans n'a trouvé de solution rapide au problème de la factorisation, alors il est tout à fait clair, que seule une véritable révolution mathématique ou informatique serait capable de remettre en cause ce cryptosystème.

De même, casser l'algorithme ElGamal est dans la plupart des cas au moins aussi difficile que de calculer le logarithme discret. Cependant, il est possible qu'il existe des moyens de casser l'algorithme sans résoudre le problème du logarithme discret. Et pour assurer une bonne sécurisation de cet algorithme, Zimmermann en 2005 [49] a recommandé l'utilisation des clés d'au moins 1024 bits. Signalons aussi que le désavantage d'ElGamal par rapport à RSA réside dans le fait que le message chiffré est deux fois plus gros que le clair.

- **Comparaison entre les cryptosystèmes symétriques et asymétriques :**

Le tableau ci-dessous présente une comparaison entre les systèmes de chiffrement symétriques et les systèmes de chiffrement asymétriques, en énumérant les principaux avantages et inconvénients de chaque mode de cryptage.

Méthode	Exemples	Avantages	Inconvénients
À clefs Secrètes	DES, AES	▪ Rapidité de calcul en général (dépend de la taille de la clé). ▪ Adaptée au cryptage de flux de données.	▪ Moins sécurisé (DES). ▪ Problème de communication de clefs entre émetteur et récepteur. ▪ Une clé pour chacun des correspondants : n personnes => n(n-1)/2 clés.
À clefs Publiques	RSA, ElGamal	▪ Très sécurisée à cause de l'utilisation de deux clés distinctes, l'une ne permettant pas de retrouver l'autre. ▪ Permet la signature électronique. ▪ Un couple de clés publique/privée suffisant pour 'n' correspondants.	▪ Lente. ▪ Problèmes de gestion de clefs publiques.

Tableau I.1. *Comparaison entre les méthodes de chiffrement symétriques et asymétriques*

- **Clé publique ou clé secrète, un compromis**

La question qui se pose à ce niveau est : Dans quels cas on utilise le chiffrement symétrique ? Et dans quels autres cas le chiffrement asymétrique est conseillé ?

En effet, et d'après la table comparative présentée juste avant, il est clair que les systèmes de chiffrement à clé publique sont très lents par rapport aux systèmes de chiffrement à clé privée. Alors que, l'algorithme de chiffrement ne doit pas être le facteur limitant à notre époque où la vitesse de transmission de l'information constitue un enjeu crucial.

De plus, et à partir des descriptions des systèmes de chiffrement présentées précédemment, on arrive à constater que la taille des clés nécessaire en cryptographie à clé publique pour assurer une sécurité satisfaisante est plus grande que la taille des clés en cryptographie à clé secrète. En fait, la notion et l'importance de la taille de clé pour assurer la sécurité ne sont légitimes que dans le cas de la clé secrète, puisque ces systèmes reposent sur l'hypothèse que les seules attaques possibles sont les attaques exhaustives [50]. Mais, dans le

cas de la clé publique, la taille de clé n'a de pertinence que lorsqu'on considère le même système. Donc, le fait de dire que RSA de 512 bits est bien moins sûr qu'un AES de 128 bits, n'a aucune signification. Cependant, la seule mesure légitime pour évaluer un cryptosystème à clé publique est la complexité de la meilleure attaque connue.

4.3. Algorithmes de chiffrement hybrides

4.3.1. PGP

Philip Zimmermann, qui est un mathématicien passionné par l'informatique, et en croyant à la philosophie qui dise que tout individu a droit à la confidentialité, notamment les organisations des droits de l'homme dans des pays soumis à la dictature, a commencé à travailler en 1984 sur un système cryptographique aussi sûr mais plus souple que le RSA. Ainsi, il a développé le **PGP** (**Pretty Good Privacy**) en 1991, puis il l'a mis à disposition gratuitement sur Internet sans se préoccuper des détails juridiques qui concernent son utilisation de RSA sans l'accord de son propriétaire, ou de son vendeur, ViaCrypt [41]. Après quelques négociations commerciales et trois ans de menaces judiciaires par le gouvernement américain, PGP est à nouveau accessible depuis 1993, mais à 150 $ cette fois-ci et vendu par... ViaCrypt.

Maintenant, les principes et formats de messages utilisés par PGP ont été normalisés à l'IETF sous le nom **OpenPGP** [51].

4.3.1.1. Principe

Lorsqu'un utilisateur chiffre un texte avec le système de chiffrement hybride PGP combinant des fonctionnalités de la cryptographie à clé publique et de la cryptographie symétrique, les données sont d'abord compressées. Cela à pour objectif de réduire le temps de transmission de ces données, et d'économiser l'espace disque et, surtout, le renforcement de la sécurité cryptographique du moment où, les cryptanalystes exploitent les modèles trouvés dans le texte en clair pour casser le chiffrement ; alors que la compression réduit ces modèles dans le texte en clair. Par conséquent, la résistance à la cryptanalyse sera, considérablement, améliorée.

Ensuite, l'opération de chiffrement se fait principalement en deux étapes, qu'on résume comme suit [52]:

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

L'opération de déchiffrement se fait également en deux étapes, qui sont [52]:

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

Remarque :

L'**IDEA** (**International Data Encryption Algorithm**), qui est un cryptosystème symétrique inventé en 1992, effectue des opérations du même genre que celles vues avec l'algorithme DES, et il manipule des blocs de 64 bits avec une clé de 128 bits.

En combinant les deux modes de cryptage symétrique et asymétrique, cette méthode de chiffrement profite des avantages de ces deux modes, à savoir la simplicité et la facilité d'utilisation du cryptage asymétrique et la rapidité de calcul du cryptage symétrique. De plus, le cryptage asymétrique résout le problème de la distribution des clés. Ainsi, les performances seront améliorées sans que la sécurité soit affectée.

4.3.1.2. Cryptanalyse

Depuis 1978, la recherche universitaire civile a intensément attaqué la cryptographie à clé publique, sans pour autant réussir à la remettre en cause. Mais cela ne fournisse aucune garantie totale sur la sécurité de cette manière de chiffrer, car une attaque menée par le gouvernement, par exemple, qui ne se manque pas de ressources très développées ou même en utilisant quelconques nouvelles percées mathématiques classées top-secret, peut tenir à bout ces cryptosystèmes conventionnels utilisés dans PGP.

Tout de même, l'optimisme semble justifié. Les algorithmes de clé publique, les algorithmes de contraction de message, et les chiffres par blocs utilisés dans PGP ont été conçus par les meilleurs cryptographes du monde [53]. Les chiffres de PGP ont subi des analyses de sécurité approfondies et des examens méticuleux de la part des meilleurs cryptographes dans le monde non classé top secret. De plus, et même si les chiffres par blocs utilisés dans PGP possède quelques faiblesses, la compression du texte clair utilisée avant le chiffrement réduit de façon considérable ces faiblesses.

4.3.2. GPG

GPG (**Gnu Privacy Guard**) est dans le principe un clone de PGP, ou plus exactement une implémentation de l'OpenPGP, mais n'utilise aucun code de PGP [51]. Donc, c'est l'équivalent libre de PGP. Il est entièrement écrit par des développeurs bénévoles, et est complètement libre, sous licence GPL (General Public License). Ainsi, il est remis à jour continuellement, aussi bien au niveau des fonctionnalités, qu'au niveau des éventuels problèmes d'implémentation.

4.3.3. Conclusion

Aujourd'hui, de nombreux gouvernements ont restreint l'usage du PGP, qui a été largement diffusé par son développeur, en pensant qu'un cryptage trop fiable ferait le jeu des terroristes et des trafiquants. Ainsi, PGP est inutilisable en France, par exemple. Toutefois, il y'a pas mal d'applications qui utilisent encore ce système de chiffrement, telles que les paiements en ligne qui se font grâce au procédé SSL fonctionnant selon le principe du PGP.
De sa part, GPG, qui est un cryptosystème utilisant des algorithmes de chiffrement à clé publiques (DSA, RSA et ElGamal), est largement utilisé dans les communications par messagerie, c-à-d pour chiffrer des mails, ainsi que pour signer des données.

4.4. Algorithme de chiffrement évolutionniste OTL

C'est un algorithme de chiffrement, développé par *Omary Fouzia* en 2006, en exploitant les algorithmes évolutionnistes. Il vise à apporter le maximum de désordre sur les positions des caractères d'un message à chiffrer. Ces caractères appartiennent à l'ensemble des 256 caractères du code ASCII. L'application de l'algorithme est précédée d'une phase de brouillage du texte initial (M_0) qui peut combiner plusieurs méthodes simples comme les substitutions, les permutations, le chiffrement affiné, etc ; pour obtenir un texte initialement chiffré (M_0').
Pour ce faire, le codage adopté pour représenter les individus, qui sont les différents messages, est résumé à travers la figure ci-dessous :

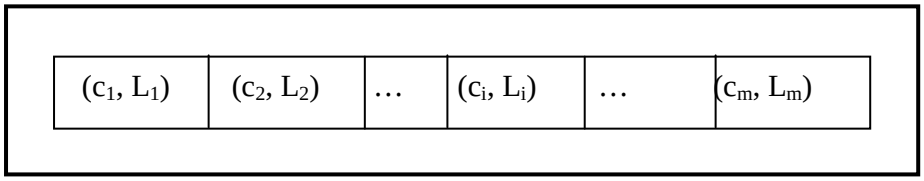


Figure I.20. *Codage des individus*

Avec : ci un caractère appartenant au message.
L_i la liste des positions du caractère c_i.
m le nombre des différents caractères du message.

Remarque :

$L_i \cap L_j = \emptyset$, $\forall i, j \in [1, m]$

L'algorithme OTL cherche à changer itérativement la répartition des listes L_i sur les différents caractères du message (sans modifier le contenu des listes) de telle manière que la différence entre le cardinal de la nouvelle liste affectée à chaque caractère c_i et le cardinal de la liste L_i d'origine soit maximale [54]. Cette variation, en terme de répartition, est assurée grâce aux opérateurs génétiques choisis (MPX pour le croisement avec un taux compris entre 60% et 100%, et une simple permutation pour la mutation avec un taux compris entre 0.1% et 5%). Et pour juger la pertinence des individus construits, la fonction d'évaluation utilisée est celle donnée ci-dessous. Les meilleurs individus sont ensuite sélectionnés par le biais de la méthode classique de la roulette, en vue de se reproduire.

$$F(X_j) = \sum_{i=1}^{m} \left| card(L_{j_i}) - card(L_i) \right|$$

Le processus évolutionnaire est répété jusqu'à la satisfaction d'un critère d'arrêt exprimé à l'aide de la fonction F. Cette dernière est bornée car : $0 \le F(X) \le 2 * l$ (l est la taille du message) pour tout individu X. En fait [54]:

$$\sum_{i=1}^{m} \left| card(L_{j_i}) - card(L_i) \right| \le \sum_{i=1}^{m} (card(L_{j_i}) + card(L_i)) \le 2 * l$$

L'opération de déchiffrement, quand à elle, se fait en deux étapes :

1) Tout d'abord, le texte chiffré est représenté suivant le codage proposé en une suite de listes de positions, et c'est grâce à la clé génétique que les caractères vont retrouver leurs listes de position correspondantes dans le message en clair. En effet, la clé, qui peut être d'un usage symétrique ou asymétrique, est une permutation de {1,2,…,m}. Comme résultat, nous obtenons le message M_0'.

2) La deuxième étape, correspond au déchiffrement du message M_0' pour obtenir M_0.

5. Récapitulation des algorithmes de chiffrement

Depuis des décennies, les algorithmes de chiffrement ont sans cesse été découverts, améliorés et … cassés !

Un historique récapitulatif des plus fameux des algorithmes de chiffrement, symétriques ou asymétriques, ceux qui sont cassés et ceux qui sont encore considérés comme sûrs, est retracé à travers le tableau suivant [55].

Année d'apparition	**Nom**	**Taille**	**Année de cassage**	**Par qui ?**	**Informations**
1974	RSA-100	100 chiffres	1991	Inconnu	/
1974	RSA-110	110 chiffres	1992	Inconnu	/
1974	RSA-120	120 chiffres	1993	Inconnu	/
1974	RSA-129	129 chiffres	1994	Inconnu	100$ ont été donnés par RSA Labs à ceux qui ont découvert le moyen de factoriser ce nombre de 129 chiffres
1974	RSA-130	130 chiffres	1996	Inconnu	/
1974	RSA-140	140 chiffres	1999	Une équipe internationale de chercheurs et la puissance de calcul du SARA Amsterdam Academic Computer Center	/
1974	RSA-155	155 chiffres	1999	Une équipe internationale de chercheurs	/
1974	RSA-160	160 chiffres	2002	Une équipe internationale de chercheurs du BSI	/
1974	RSA-576	560 chiffres	2003	Une équipe internationale de chercheurs (J. Franke, F. Bahr, M. Boehm, T. Kleinjung)	10000$ ont été donnés par RSA Labs à ceux qui ont découvert le moyen de factoriser ce nombre de 576 chiffres

1974	RSA-640	640 chiffres	2005	Une équipe allemande du "Federal Agency for Information Technology Security" (BSI)	20000$ ont été donnés par RSA Labs à ceux qui ont découvert le moyen de factoriser ce nombre de 640 chiffres
1974	RSA-704	704 chiffres	Pas encore	/	RSA Labs offre 30000$ à celui qui le cassera.
1974	RSA-768	768 chiffres	Pas encore	/	RSA Labs offre 50000$ à celui qui le cassera.
1974	RSA-896	896 chiffres	Pas encore	/	RSA Labs offre 75000$ à celui qui le cassera.
1974	RSA-1024	1024 chiffres	Pas encore	/	RSA Labs offre 100000$ à celui qui le cassera.
1974	RSA-1536	1536 chiffres	Pas encore	/	RSA Labs offre 150000$ à celui qui le cassera.
1974	RSA-2048	2048 chiffres	Pas encore	/	RSA Labs offre 200000$ à celui qui le cassera.
1976	DES-56	56 bits	1997	Utilisateurs d'Internet sous la bannière de Distributed.net et Electronic Frontier Foundation	/
1977	Triple-DES	192 bits	Pas encore	/	/
1985	ECCp-79	79 bits	1997	Inconnu	/
1985	ECCp-89	89 bits	1998	Inconnu	/
1985	ECCp-97	97 bits	1998	Inconnu	5000$ ont été attribués par Certicom à ceux qui ont cassé l'algorithme
1985	ECCp-109	109 bits	2002	Utilisateurs d'Internet sous la bannière d'Ecc2.com	10000$ ont été attribués par Certicom à ceux qui ont cassé l'algorithme
1985	ECCp-131	131 bits	Pas encore	/	Certicom offre 20000$ à la première personne qui cassera l'algorithme. Nombre de jours/machine nécessaire au cassage

					: 2.3 x 10^{10} (estimation Certicom)
1985	ECCp-163	163 bits	Pas encore	/	Certicom offre 30000$ à la première personne qui cassera l'algorithme. Nombre de jours/machine nécessaire au cassage : 2.3 x 10^{15} (estimation Certicom)
1985	ECCp-191	191 bits	Pas encore	/	Certicom offre 40000$ à la première personne qui cassera l'algorithme. Nombre de jours/machine nécessaire au cassage : 4.8 x 10^{19} (estimation Certicom)
1985	ECCp-239	239 bits	Pas encore	/	Certicom offre 50000$ à la première personne qui cassera l'algorithme. Nombre de jours/machine nécessaire au cassage : 1.4 x 10^{27} (estimation Certicom)
1985	ECCp-359	359 bits	Pas encore	/	Certicom offre 100000$ à la première personne qui cassera l'algorithme. Nombre de jours/machine nécessaire au cassage : 3.7 x 10^{45} (estimation Certicom)
1987	RC4	40 bits	1995	Adam Back, Eric Young et David Byers	/
1994	RC5-56	56 bits	1997	Utilisateurs d'Internet sous la bannière de Distributed.net	/
1994	RC5-64	64 bits	2002	Utilisateurs d'Internet sous la bannière de Distributed.net	/
1994	RC5-72	72 bits	Pas encore	/	Durée estimée de cassage par Distributed.net au 01/10/2004 : 800 ans. Rejoignez notre équipe de casssage de la clé RC5-72

1997	CS Cipher (CSC-56)	56 bits	2000	Utilisateurs d'Internet sous la bannière de Distributed.net	/
1999	AES-128	128 bits	Pas encore	/	/
1999	AES-192	192 bits	Pas encore	/	/
1999	AES-256	256 bits	Pas encore	/	/
Inconnue	ECC2K-130	131 bits	Pas encore	/	Certicom offre 20000$ à la première personne qui cassera l'algorithme. Nombre de jours/machine nécessaire au cassage : 2.7×10^9 (estimation Certicom)
Inconnue	ECC2-131	131 bits	Pas encore	/	Certicom offre 20000$ à la première personne qui cassera l'algorithme. Nombre de jours/machine nécessaire au cassage : 6.6×10^{10} (estimation Certicom)
Inconnue	ECC2-163	163 bits	Pas encore	/	Certicom offre 30000$ à la première personne qui cassera l'algorithme. Nombre de jours/machine nécessaire au cassage : 2.9×10^{15} (estimation Certicom)
Inconnue	ECC2K-163	163 bits	Pas encore	/	Certicom offre 30000$ à la première personne qui cassera l'algorithme. Nombre de jours/machine nécessaire au cassage : 4.6×10^{14} (estimation Certicom)
Inconnue	ECC2-191	191 bits	Pas encore	/	Certicom offre 40000$ à la première personne qui cassera l'algorithme. Nombre

					de jours/machine nécessaire au cassage : 1.4 x 10^{20} (estimation Certicom)
Inconnue	ECC2-238	239 bits	Pas encore	/	Certicom offre 50000$ à la première personne qui cassera l'algorithme. Nombre de jours/machine nécessaire au cassage : 3.0 x 10^{27} (estimation Certicom)
Inconnue	ECC2K-238	239 bits	Pas encore	/	Certicom offre 50000$ à la première personne qui cassera l'algorithme. Nombre de jours/machine nécessaire au cassage : 1.3 x 10^{26} (estimation Certicom)
Inconnue	ECC2-353	359 bits	Pas encore	/	Certicom offre 100000$ à la première personne qui cassera l'algorithme. Nombre de jours/machine nécessaire au cassage : 1.4 x 10^{45} (estimation Certicom)
Inconnue	ECC2K-358	359 bits	Pas encore	/	Certicom offre 100000$ à la première personne qui cassera l'algorithme. Nombre de jours/machine nécessaire au cassage : 2.8 x 10^{44} (estimation Certicom)

Tableau I.2. *Récapitulatif des algorithmes de chiffrement*

D'après les informations comprises dans ce tableau, et pour assurer une bonne sécurité des applications cryptographiques, il est conseillé actuellement de choisir **AES** et **RSA-704**. Toutefois, triple DES vit certainement ses dernières années de robustesse.

6. Conclusion

Dans ce chapitre un état de l'art aussi riche que possible sur la cryptographie, depuis sa première apparition jusqu'à nos jours, a été présenté. D'après cette étude, un système cryptographique est considéré comme sûr si personne n'a encore mis en défaut sa sécurité. Il a été montré que si l'on est capable de mettre en défaut la sécurité d'un schéma, on peut alors résoudre un problème mathématique réputé difficile. En plus, de la variété des modèles mathématiques exploités dans le domaine de la cryptographie, une nouvelle approche cryptographique, fertile et prometteuse, a été proposée. Il s'agit de la « **cryptographie évolutionniste »**, qui consiste en l'application des algorithmes évolutionnistes en cryptographie. Ceci fera l'objet de notre étude.

Chapitre II
Algorithmes évolutionnaires

1. Introduction

Les phénomènes physiques ou biologiques ont été la source d'inspiration de nombreux algorithmes. Ainsi les réseaux de neurones artificiels s'inspirent du fonctionnement du cerveau humain, l'algorithme de recuit simulé de la thermodynamique, et les **algorithmes évolutionnaires** (**AEs**) de l'évolution darwinienne des populations biologiques qui leur permis d'évoluer au cours du temps en créant des systèmes biologiques très complexes adaptés à de nombreuses conditions. Cette dernière branche a été exposée pour la première fois en 1859 par *Charles Darwin* en publiant « *L'Origine des espèces* » [56]. Disons que la petite partie qui est grossièrement imitée et caricaturée lors de la conception des algorithmes évolutionnaires est basée sur l'idée que l'apparition d'espèces adaptées au milieu est la conséquence de la conjonction de deux phénomènes [57]: d'une part la sélection naturelle imposée par le milieu, qui permette aux individus les plus adaptés de survivre et de se reproduire ; et d'autre part des variations non dirigées du matériel génétique des espèces. Ce sont ces deux principes qui sous-tendent les algorithmes évolutionnaires.

Plus explicitement, le principe de l'évolution darwinienne repose sur les observations suivantes [58]:

- ✓ Il existe au sein de chaque espèce de nombreuses variations, ainsi, chaque individu étant différent ;
- ✓ les ressources naturelles étant finies, d'autre part, il naît rapidement plus d'être vivants que la nature ne peut nourrir ; il en résulte une lutte pour l'existence entre chaque organisme ;
- ✓ les individus survivants possèdent des caractéristiques qui les rendent plus aptes à survivre. *Darwin* baptise ce concept **sélection naturelle** ;
- ✓ les organismes survivants transmettent leurs avantages à leur descendance qui peuvent être encore meilleurs que leurs parents. L'accumulation au cours des générations des

petites différences entre chaque branche généalogique crée de nouvelles espèces de plus en plus aptes à survivre.

Les AEs font partie du champ de l'Intelligence Artificielle (IA), inspirée de « l'intelligence » de la nature. Intelligence que *Fogel* [59] avait défini de la façon suivante: « *The capability of a system to adapt its behavior to meet its goals in a range of environments* ». Ces algorithmes fournissent des solutions aux problèmes difficiles pour lesquels aucune méthode de résolution n'est envisageable et où les solutions sont représentées comme un ensemble de paramètres [60]. Ils constituent une méthode d'exploration automatique d'un espace de recherche potentiellement très vaste. Contrairement à d'autres algorithmes partant d'une solution singulière et cherchant à remonter un gradient de performances, les AEs utilisent un ensemble de solutions dont seule la performance ponctuelle est utilisée et aucune autre propriété mathématique n'est nécessaire. De ce fait, et par rapport aux méthodes habituelles que l'on peut qualifier d'*analytiques*, les algorithmes évolutionnaires peuvent être qualifiés de *synthétiques* [59] puisqu'ils peuvent parfois synthétiser des solutions nouvelles et originales à des problèmes connus car ils expérimentent sans idées préconçues, si ce n'est les paramètres et l'espace de recherche qu'on leur impose.

Par conséquence, le champ des applications potentielles est très vaste à cause de la pauvreté des informations nécessaires et de la généricité des principes exploités. Ainsi, ces algorithmes permettent de résoudre non seulement des problèmes purement théoriques en combinatoire, en économie, en apprentissage, dans la théorie des jeux, mais aussi des problèmes liés à des applications réelles complexes, telles que l'analyse des sondages de sous-sol et la détection des champs pétrolifères, la fabrication des emplois du temps, prévision des cours de la bourse, le contrôle des pipe-lines de gaz, la conception des automobiles, l'optimisation des ailes d'avion, les manœuvres des avions de combat, les allocations de routes aériennes, les allocations dynamiques de fréquences en téléphonie mobile (meilleur résultat actuel), le positionnement d'antennes, le routage dans les réseaux, … De plus, ils peuvent aussi être utilisés pour contrôler un système évoluant dans le temps (chaîne de production, centrale nucléaire…) car la population peut s'adapter à des conditions changeantes.

En fait et suivant la façon de traduire les principes Darwiniens, le terme **algorithmes évolutionnaires** couvre un ensemble de techniques, nommées **algorithmes génétiques**, **stratégies d'évolution**, **programmation évolutionnaire,** et **programmation génétique**.

2. Historique

C'est en 1860 que *Charles Darwin* publie son livre intitulé « *L'origine des espèces au moyen de la sélection naturelle ou la lutte pour l'existence dans la nature* », dans lequel il rejette l'existence «de systèmes naturels figés», adaptés pour toujours à toutes les conditions extérieures, et il expose sa théorie de l'évolution des espèces [61]: « sous l'influence des contraintes extérieurs, les êtres vivants se sont graduellement adaptés à leur milieu naturel au travers de processus de reproductions ».
Presque simultanément, en 1866, *Mendel* publie l'article retraçant dix années d'expériences d'hybridation chez les végétaux en termes de recombinaison de leurs gênes, et l'adresse aux sociétés scientifiques à travers le monde, mais les réactions n'étaient pas du tout encourageantes. Ce n'est qu'en 1900, que des résultats similaires à ceux de *Mendel* ont fait l'objet de trois nouveaux articles signés *Hugo de Vries*, *Carl Correns* et *Erich von Tschermark.*

C'est alors à partir du 20^{ième} siècle que la mutation génétique a été mise en évidence par des chercheurs en informatique qui essayent de développer des méthodes permettant aux systèmes d'évoluer de manière normale et efficace face à de nouvelles conditions d'environnement inconnues, variables ou évolutives. Ainsi, les problèmes de traitement de l'information ne seront plus résolus de manières figées, car il ne sera plus indispensable lors de la phase de conception d'un système, d'énumérer toutes les caractéristiques nécessaires pour les conditions d'exploitations connues au moment de la conception.

Dans les années 1960, *John Holland*, ses collègues et ses étudiants ont mené des recherches, à l'université de Michigan, poussés par deux objectifs principaux [62]: premièrement, mettre en évidence et expliquer rigoureusement les processus d'adaptation des systèmes naturels, et deuxièmement, concevoir des systèmes artificiels qui possèdent les propriétés importantes des systèmes naturels. Toutefois, c'était *Bagley* qui a mentionné, en premier lieu, l'expression « Algorithme Génétique ». C'était en 1967. Et en 1975 *Holland* a introduit, dans son livre *« Adaptation in Natural and Artificial Systems* », le premier modèle formel des algorithmes génétiques : « *the Canonical Genetic Algorithm CGA* ». Ce modèle a servi de base aux recherches ultérieures et a été, plus particulièrement, repris par *David Goldberg* qui a publié, en 1989, un ouvrage de vulgarisation des algorithmes génétiques, tout en ajoutant à la théorie des algorithmes génétiques les idées suivantes [63]:

- ✓ un individu est lié à un environnement par son code d'ADN.
- ✓ une solution est liée à un problème par son indice de qualité.

L'originalité des travaux de *Holland* repose en particulier sur le fait qu'il n'a pas considéré seulement les mutations comme mécanisme d'évolution, mais il exploite surtout les mécanismes de croisement, car c'est en croisant les solutions potentielles existant au sein du pool génétique que l'on peut se rapprocher de l'optimum [64].

Trois types d'algorithmes évolutionnaires ont été développé isolément et à peu près simultanément, par différents scientifiques : la programmation évolutionniste de *L. Fogel* -Californie, USA, 1966-, les Stratégies d'évolution inventée en 1973 par deux étudiants ingénieurs à l'Université de Berlin, *I. Rechenberg* et *H.P.Swhefel* et les Algorithmes Génétiques de *J. Holland* en 1975. En 1985 et grâce à *Cramer* [60], une autre classe d'algorithmes évolutionnaires a vu le jour. C'est la programmation génétique (PG), considérée en premiers temps comme sous groupe des AGs. Mais cette façon de programmer, qui consiste à faire évoluer le code d'un logiciel afin qu'il remplisse au mieux certaines tâches, a pris son indépendance, en 1992, grâce à *J. Koza* (Californie, USA).

Ces différentes classes d'algorithmes évolutionnaires dont les origines diffèrent et qui possèdent chacune leur spécificité, ne diffèrent que sur les détails d'implantation des opérateurs et sur les procédures de sélection et remplacement de la population. De plus, et malgré que leurs buts soient différents à l'origine, ils ont commencé à sortir de leur isolement et sont maintenant surtout utilisés pour résoudre des problèmes d'optimisation en convergeant vers le modèle unique des algorithmes évolutionnistes qui intègre les particularités de chacun des modèles.

3. Les quatre grandes familles des algorithmes évolutionnaires

3.1. Programmation évolutionnaire

La programmation évolutionnaire développée par *L.J. Fogel*, se base sur l'évolution d'une population d'automates à états finis (Figure II.1) pour résoudre des problèmes de prédiction. Ce modèle évolutionniste accentue l'utilisation de la mutation et n'utilise pas dans sa version originale la recombinaison des individus par croisement [65]. La table de transition des automates est modifiée grâce à des mutations aléatoires uniformes dans l'alphabet discret correspondant. Chaque automate de la population parente génère un enfant par mutation, et

les meilleures solutions entre les parents et les enfants sont sélectionnées pour survivre, sachons que l'évaluation de la performance des individus correspond au nombre de symboles prédits correctement.

Par la suite, la programmation évolutionnaire a été développée et son domaine a été élargi par *D.B. Fogel*, afin qu'il puisse travailler dans l'espace réel, où la sélection déterministe est remplacée par un tournoi stochastique (voir section 4.2.3).

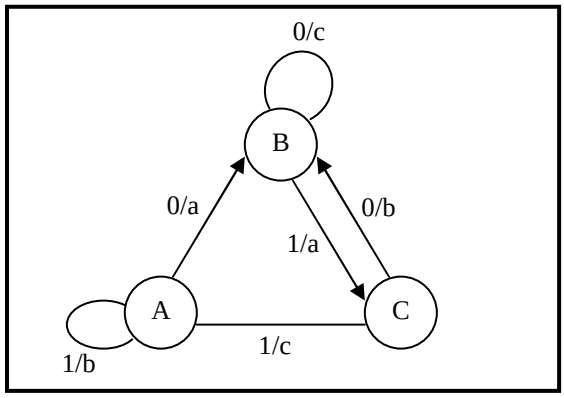


Figure II.1. *Exemple d'un automate à états finis ayant trois états différents S={A,B,C}, un alphabet d'entrée I={0,1}, et un alphabet de sortie O={a,b,c}. Chaque arrête entre deux états indique une transition possible, et la fonction de transition δ : S x I ➔ S x O est spécifiée par les labels au niveau des arrêtes ayant la forme i / o, signifiant que* $\delta(s_i, i) = (s_j, o)$

3.2. Stratégies d'évolution

Les stratégies d'évolution sont dédiées à l'optimisation des problèmes continus dans l'espace de vecteurs des réels [60]. Les premiers efforts pour la mise en place des stratégies d'évolution ont eu lieu en 1973 à l'université de Berlin au cours de la résolution d'un problème aérodynamique. C'est avec ces méthodes évolutionnistes que la notion d'auto-adaptativité pour la mutation permettant de contrôler cette fonction de mutation, a été apparue. Une mise en œuvre de ce principe consiste à augmenter l'intensité de la mutation lorsque la proportion de descendants de bonne qualité, c'est à dire le nombre de mutation à succès, dépasse 20% de la population totale [65]. Elle est diminuée dans le cas opposé. Une interprétation possible de cette règle est la suivante : si la proportion de mutation réussie est élevée, l'espace de recherche exploré est restreint autour d'un optimum local, il faut donc diversifier la population en augmentant le taux de mutation, ce qui revient à ajuster la variance de la mutation au cours du temps par le processus d'évolution. C'est ce que les méthodes évolutionnistes auto-adaptatives visent de le faire : automatiser le réglage des paramètres de l'algorithme évolutionniste pour remédier aux méthodes empiriques du type « essais-erreurs » qui sont employées dans la pratique. De plus, ces approches utilisent un

opérateur de sélection de type déterministe: les solutions dont le *fitness* (la valeur de la fonction d'évaluation des individus) est mauvais sont éliminées de la population. En outre, dans le modèle originel, les populations des parents et de leurs descendants sont généralement de taille différente.

3.3. Algorithmes génétiques

Les algorithmes génétiques (AGs) sont probablement les algorithmes les plus connus et les plus utilisés dans le calcul évolutionnaire. Ils ont été développé dans les années soixante par *Holland* qui les a appliqué à l'optimisation paramétrique pour la première fois en 1975, en posant, ainsi, les fondements de cette technique d'application. Cependant, cette technique n'a pas été appliquée sur des problèmes réels de grande taille, à cause des machines calculatoires, de l'époque, et qui n'ont pas été suffisamment puissantes. Ce n'est que vers la fin des années quatre vingt, précisément, avec l'apparition de l'ouvrage de référence écrit par *Goldberg*, que les algorithmes génétiques ont été connu dans la communauté scientifique. Leur particularité est qu'ils sont fondés sur le *Néo-Darwinisme*, c'est-à-dire l'union de la théorie de l'évolution et de la génétique moderne. Ainsi, les variables sont généralement codées en binaire, par analogie avec les quatre lettres de l'alphabet génétique d'ADN, sous forme de gènes dans un chromosome. Ensuite, des opérateurs génétiques, à savoir le croisement et la mutation, sont appliqués à ces chromosomes.

Les AGs sont utilisés pour retrouver une solution résolvant un problème donné, et ce sans avoir de connaissance à priori sur l'espace de recherche. Seul un critère de qualité est nécessaire pour évaluer les différentes solutions en quantifiant, ainsi, leur capacité à résoudre le problème donné. Donc, le but scientifique et technologique visé par ces algorithmes est de pouvoir traiter des problèmes d'optimisation globaux, grâce à la *généralité* avec laquelle on représente l'espace de recherche qui peut contenir des booléens (système actif ou non), des entiers (nombre de composants à optimiser), des réels (intensités associées aux composants réglables), ou des fonctions discrétisées (optimisation de forme), et grâce aussi à la *robustesse* de la convergence [66].

Il convient de noter, que cette robustesse observée en pratique dans de nombreux domaines applicatifs de l'ingénierie, n'a pas encore selon *Simon Baudot-Roux* [66] une assise théorique très utilisable pour guider les choix algorithmiques, qui sont encore aujourd'hui surtout empiriques. Mais ça n'empêche de noter qu'il existe une école théoricienne qui s'appuie sur

les processus stochastiques, mais dont l'impact méthodologique n'est pas encore tout à fait clair.

3.4. Programmation génétique

L'idée de faire évoluer des programmes date des années cinquante où *Friedberg*, en 1958, a fait plusieurs tentatives pour avoir des ordinateurs auto-programmables en utilisant ce qui est de la mutation actuellement. Donc, et à partir d'une population constituée de programmes aléatoires dont il modifie leurs contenus stochastiquement, il essaye de les améliorer pour aboutir à des résultats satisfaisants. Plutard, *Smith* (1980) travaillant sur les systèmes classifieurs d'apprentissage, a introduit de petits programmes dans les règles qu'il cherche à les faire évoluer [60]. Il convient de noter que, la première utilisation des structures arborescentes dans un algorithme génétique a été suggérée par *Cramer* en 1985, dans le but de faire évoluer des sous-programmes séquentiels d'un langage algorithmique simple.

Toutefois, c'est grâce à *John Koza* (1992) que cette présentation a été adopté pour définir la programmation génétique comme un nouvel algorithme évolutionnaire, en étendant, ainsi, le modèle d'apprentissage des AGs à l'espace des programmes. Donc, son objectif initial était de faire évoluer des sous-programmes du langage LISP (Figure II.2), et c'est d'ailleurs grâce à son ouvrage que l'utilisation de la programmation génétique s'est étendue à la résolution de nombreux types de problèmes où les solutions peuvent être représentées par des structures arborescentes dont les feuilles sont constituées de symboles terminaux (variables, constantes,…); et les nœuds internes de symboles fonctionnels (opérateurs).

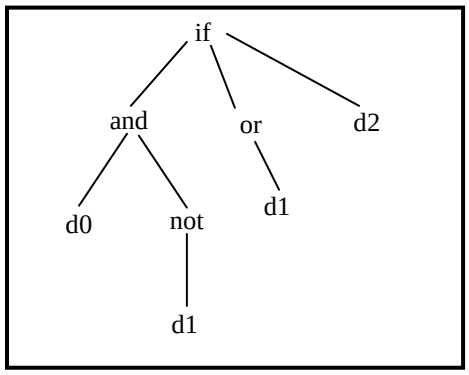


Figure II.2. *Exemple d'une solution Programmation génétique en LISP : {d0,d1,d2} est un ensemble d'instructions constituant les terminaux, et {if, and,or} sont des expressions LISP constituant les nœuds*

Actuellement, la distinction entre ces quatre approches est de plus en plus floue. Le génotype, qui est le codage d'un individu, étant souvent constitué d'un mélange de structures

complexes (arbres, graphes, listes de paramètres, ...) [58]. Donc, la différence entre ces quatre catégories est essentiellement d'ordre historique.

4. Vocabulaire et Principe

4.1. Vocabulaire

Les mécanismes utilisés par les AEs reposent sur le principe de compétition entre les individus, où les mieux adaptés aux conditions survivent et peuvent laisser une descendance qui répandra leurs gènes. De ce fait, le vocabulaire employé est directement calqué sur celui de la théorie de l'évolution et de la génétique. Ainsi, il regroupe les termes suivants :

- **Individu :** Un élément de l'espace de recherche. C'est aussi une solution potentielle du problème.
- **Population :** Un ensemble fini (de taille N) d'individus.
- **Génération :** Correspond à l'itération, c-à-d repère le moment de l'évolution. Mais parfois ce terme signifie la population en une certaine itération.
- **Evolution :** Un processus itératif de recherche d'un, ou de plusieurs, individu optimal.
- **Performance :** La mesure de la qualité des individus basé sur l'objectif de l'optimisation et permettant de comparer les individus entre eux afin d'en déterminer les plus aptes.
- **Evaluation** d'un individu **:** Le calcul de sa performance.
- **Croisement :** L'opérateur de reproduction appliqué avec une probabilité P_c et correspondant à un mélange d'information des individus de la population entre elles. Il consiste à échanger des parties composantes (gènes) entre un ou plusieurs individus.
- **Mutation :** L'opérateur de modification d'un ou plusieurs gènes, appliqué avec une probabilité P_m dans le but de produire une nouvelle diversité dans la population.
- **Sélection :** Processus du choix des individus pour la reproduction en se basant sur leur performance.
- **Remplacement :** Processus de formation d'une nouvelle population à partir des ensembles de parents et d'enfants choisis suivant leur performance.

Ce vocabulaire suffit pour présenter et comprendre le principe de fonctionnement des AEs, que nous allons présenter maintenant.

4.2. Principe

Pour qu'ils puissent surpasser d'autres méthodes plus classiques dans la quête de la robustesse, les AEs doivent être fondamentalement différents. Ils le sont en fait selon quatre axes principaux [62]:

1) Les AEs utilisent un codage des éléments de l'espace de recherche et non pas les éléments eux-mêmes.

2) Les AEs recherchent une solution à partir d'une population de points et non pas à partir d'un seul point.

3) Les AEs n'imposent aucune régularité sur la fonction étudiée (continuité, dérivabilité, convexité…). C'est l'un des gros atouts de ces algorithmes.

4) Les AEs ne sont pas déterministes, ils utilisent des règles de transition probabilistes.

La question qui se pose maintenant, est : Comment crée-t-on cette évolution ? Comment fait-on évoluer ces gènes ? En fait, la recette est assez simple, et tourne autour des huit points fondamentaux suivants :

1) Tout d'abord, une population initiale d'individus est crée en choisissant des valeurs aléatoires pour les gènes constituant ces individus là. Le résultat peut contenir des individus de très mauvaises performances.

2) Ces individus sont ensuite évalués. Cela revient à déterminer leurs performances.

3) Maintenant, c'est là que le principe Darwinien intervient. Il s'agit de sélectionner, parmi les individus formant la population, un certain nombre de « géniteurs » qui sont, par exemple, les individus ayant donné les meilleurs résultats lors de l'évaluation.

4) De nouveaux individus, appelés enfants, sont à ce niveau créés en reproduisant les individus sélectionnés, appelés parents, entre eux en imitant la nature, c'est à dire en recombinant leurs gènes (croisement) et/ou en les mutant ; pour obtenir une nouvelle population constituée de la population initiale plus les enfants qui viennent d'être créés,

5) Evaluer les enfants,

6) Supprimez de la population regroupant les parents et les enfants, certains individus pour revenir à la taille de la population initiale. Par exemple, les moins performants des individus sont à supprimer.

7) Définir un critère d'arrêt de l'algorithme. Il peut être un nombre prédéterminé de générations à atteindre, ou la génération d'un enfant satisfaisant les conditions requises. Dans ce cas, l'algorithme s'arrête et donne comme résultat le meilleur individu,

8) Retourner à la phase 3, et recommencer.

Les applications efficaces à base d'AEs sont évidemment plus complexes. Le problème essentiel étant d'adapter, ou même de créer, les opérateurs répondant aux spécificités du problème. De même qu'une recette d'un livre de cuisine nécessite d'être adaptée avec finesse au matériel et ingrédients disponibles, aux goûts des convives, pour être vraiment appréciée (!) [67]

Ce procédé peut être résumé ou schématisé comme suit :

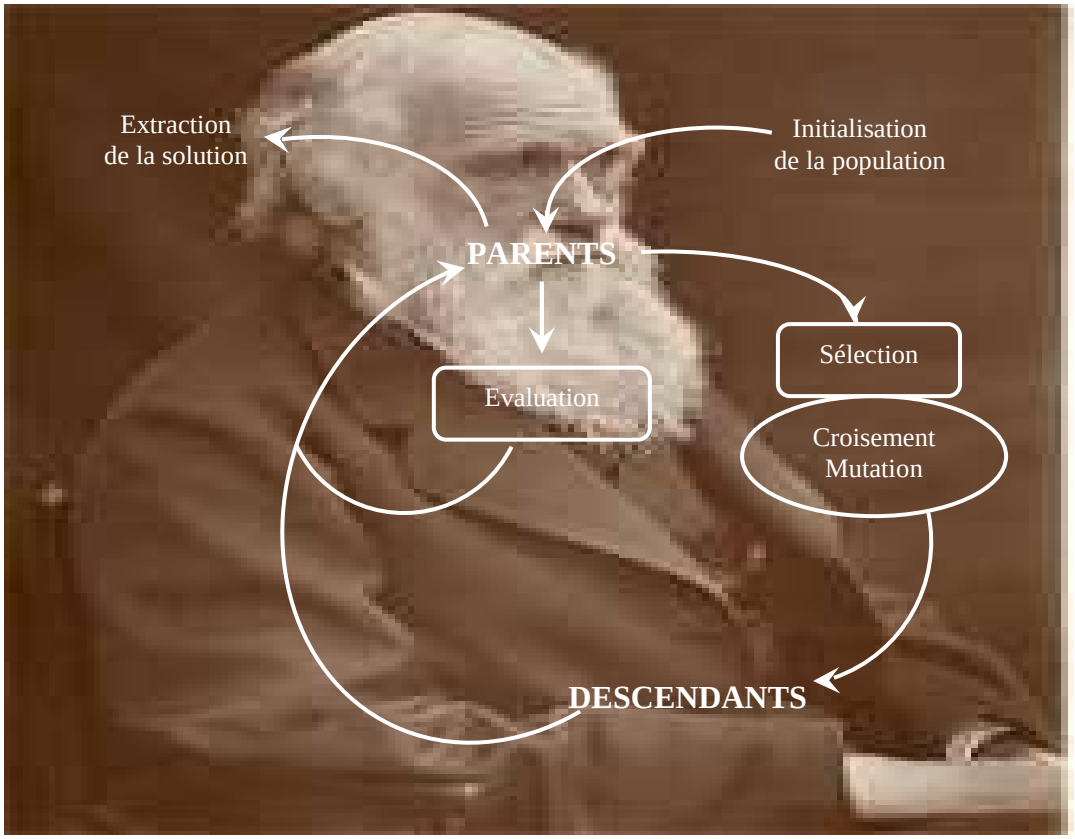


Figure II.3. *Cycle d'un algorithme évolutionnaire*

Comme c'était déjà mentionné, un tel algorithme ne nécessite aucune connaissance du problème, et on peut représenter celui-ci par une boîte noire comportant des entrées, qui sont les variables, et des sorties qui sont les fonctions objectif. L'algorithme ne fait que manipuler les entrées, lire les sorties, manipuler à nouveau les entrées de façon à améliorer les sorties, etc. Il convient aussi de noter que, dans beaucoup de méthodes d'optimisation, le déplacement d'un point à un autre dans l'espace de recherche se fait avec précaution en utilisant une règle de transition pour déterminer le nouveau point. Cette méthode qui fonctionne point par point est dangereuse parce qu'elle a une forte tendance à trouver de faux pics dans des espaces de recherche multi pics. En revanche, les AEs utilisent simultanément un ensemble de points, en

escaladant plusieurs pics en parallèle ; ainsi, la probabilité de trouver un faux sommet est réduite par rapport à d'autres méthodes qui explorent point par point.
Celui-ci était grosso modo le principe de fonctionnement d'un algorithme évolutionnaire. Détaillons maintenant ses principales phases.

4.2.1 Création de la population initiale

Comme les AEs agissent sur une population d'individus, et non pas sur un individu isolé, le premier point dans l'implantation des AEs est la création d'une population initiale d'individus. Par analogie avec la biologie, chaque individu de la population est codé par un *chromosome* ou *génotype*. Une population est donc un ensemble de chromosomes, où chaque chromosome code un point de l'espace de recherche.

Plusieurs types de codage peuvent être utilisés. Le plus connu et le plus utilisé est le *codage binaire* où un gène est un entier long de 32 bits. Par conséquent un chromosome est un tableau de gènes (Figure II.4 et Figure II.5) et un individu est un tableau de chromosomes. La population est un tableau d'individus. Ainsi, on aboutit à une structure présentant cinq niveaux d'organisation (Figure II.4) [59], d'où résulte le comportement complexe des AEs.

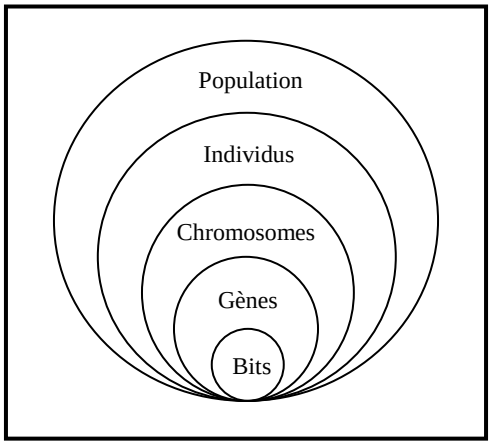


Figure II.4. *Les cinq niveaux d'organisation d'un algorithme évolutionnaire*

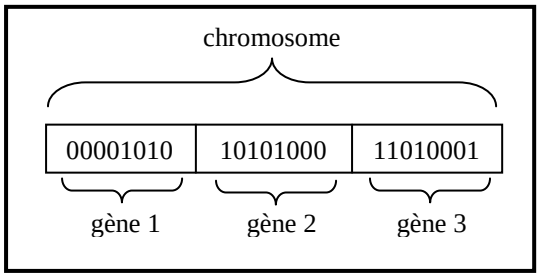


Figure II.5. *Illustration schématique du codage des variables*

Le codage binaire est le cadre général des algorithmes génétiques traditionnels, où chaque individu I est représenté par une chaîne de bits, comme suit :

$$I = \{a_{1,}, \ldots, a_q\} \quad / \ \forall i = \overline{1-q} : a_i \in \{0,1\}$$

Où : q est le nombre de bits dans la chaîne de bits.

Un des avantages du codage binaire est que l'on peut facilement coder toutes sortes d'objets : des réels, des entiers, des valeurs booléennes, des chaînes de caractères… Cela nécessite simplement l'usage de fonctions de codage et décodage pour passer d'une représentation à l'autre.

D'autres formes de codage peuvent être utilisés tels que le codage réel, le codage de Gray,… Toutefois, il sera plus raisonnable de choisir le codage qui convient le mieux avec le problème à résoudre. Imaginant, par exemple, qu'on veut résoudre le problème d'optimisation de la recette des crêpes qui nécessite comme ingrédients essentiels : la farine, le sucre, les œufs et la levure [60]. Dans ce cas, il est clair que le codage le plus adéquat contient quatre gènes qui sont :

(float cpFlour, float tbspSugar, float tspBakingPowder, int nEggs)

Il est à noter que ceci est le génotype tandis que le phénotype est le résultat de la préparation (les crêpes). L'efficacité de l'algorithme va donc dépendre grandement du choix de ce codage ce qui présente deux difficultés: limiter l'espace de recherche et fournir des solutions valides pour le problème.

Après avoir définir le bon codage, l'opération qui suit est celle de la *création de la première population* qui sera amenée à évoluer. C'est là que l'on peut injecter la ou les solutions initiales au problème que l'on a pu obtenir par ailleurs, à l'aide d'autres techniques de résolution, par exemple. On définit ainsi à cette étape la taille des populations N qui ne doit pas être trop élevée pour que le temps de calcul soit raisonnable, ni trop faible pour que la solution trouvée soit intéressante.

Si l'*initialisation* a théoriquement peu d'importance du fait qu'on converge en limite toujours vers l'optimum global, on constate expérimentalement que cette étape influence énormément la variance des résultats et aussi la rapidité d'obtention de ceux-ci [67]. Il est ainsi souvent très avantageux d'injecter le maximum de connaissances a priori sur le problème par le biais de l'initialisation. Cela revient à placer dans la population initiale des individus les plus proches de l'optimum recherché, ce qui va accélérer la convergence de l'algorithme. Malgré ça, les individus formant cette population sont le plus souvent générés *aléatoirement*.

4.2.2 Évaluation

L'objectif des AEs est de maximiser une *fonction d'adaptation* (*fitness* en anglais) ou encore dite *fonction d'évaluation*. Intuitivement, cette fonction peut être envisagée comme

une mesure de profit, d'utilité ou de qualité que l'on souhaite maximiser. Autrement dit, elle est représentative de l'efficacité des solutions générées sur un problème posé. Elle est le principal biais introduit par le concepteur pour guider l'évolution vers les solutions recherchées. La nature n'est pas aussi directive, c'est pourquoi l'analogie ne doit pas être prise au pied de la lettre [68]: l'objectif n'est pas de reproduire, ni même de comprendre le fonctionnement de la sélection naturelle, le but est de disposer d'une méthode efficace de conception et d'optimisation de structures pour répondre à des problèmes particuliers, même si des interactions sont possibles avec des biologistes.

On définit au préalable la fonction d'évaluation qui associe à chaque phénotype, qui est un vecteur de l'espace de recherche, donc c'est la solution correspondante à un génotype représentant le codage d'un individu ; il lui associe une valeur dite d'*évaluation*. C'est la note de l'individu, plus elle est élevée, plus la solution donnée par ce phénotype est meilleure. Donc, cette fonction sert à calculer le coût d'un point de l'espace de recherche, sachons que l'évaluation d'un individu ne dépend pas de celle des autres individus, et le résultat fournit par la fonction d'évaluation va permettre de sélectionner ou de refuser un individu pour ne garder que les individus ayant le meilleur coût en fonction de la population courante : c'est le rôle de la fonction *fitness*. Autrement dit, cette méthode permet de s'assurer que les individus performants seront conservés, alors que les individus peu adaptés seront progressivement éliminés de la population.

La construction de cette fonction doit être particulièrement optimisée car elle sera exécutée un grand nombre de fois, qui peut augmenter jusqu'à N fois à toutes les générations, ce qui fait que la rapidité de l'algorithme dépend essentiellement d'elle. Elle doit aussi pouvoir tenir compte des phénotypes invalides si le problème doit satisfaire des contraintes que les opérateurs de mutation et croisement ne respectent pas.

4.2.3 Sélection

La partie darwinienne de l'algorithme comprend deux étapes totalement indépendantes de l'espace de recherche, qui sont la **sélection** et le **remplacement**. La première, qui est celle de sélection, sert à déterminer les individus d'une population ayant le droit de participer à l'élaboration de la population descendante. Elle ne crée aucune nouveauté, elle se contente de choisir quels individus seront ou ne pas en mesure de contribuer à la création de la population descendante, suivant une stratégie particulière. Cette

opération est bien entendu une version artificielle de la sélection naturelle : la survie darwinienne des chaînes les plus adaptées. Sachons que dans les populations naturelles, l'adaptation est déterminée par la capacité d'une créature à survivre aux prédateurs, aux maladies, et aux autres obstacles à franchir pour atteindre l'age adulte et la période de reproduction [62]. Dans notre environnement indéniablement artificiel, une telle décision, qui permet soit de garder en vie ou de tuer un certain individu, revient à la fonction à optimiser.

L'algorithme de sélection mis en oeuvre doit assurer la convergence vers une solution efficace tout en maintenant la diversité de la population. Ceci est l'aspect caractéristique des AEs par rapport aux autres algorithmes d'apprentissage ou d'optimisation. La diversité dans la population diminue les risques de convergence prématurée vers un extremum local.

Techniques de sélection

Une panoplie de techniques de sélection est présentée à travers les points suivants :

- **Sélection par roulette (wheel)**

Avec cette technique, encore dite **sélection proportionnelle**, les parents sont sélectionnés en fonction de leur performance, où les chromosomes codant les meilleurs résultats, auront plus de chances d'être sélectionnés. Son principe ressemble à celui de la roulette de casino, du fait qu'il revient à imaginer une sorte de roulette de casino sur laquelle sont placés tous les chromosomes de la population suivant leurs valeurs d'adaptation. Un exemple de cette roulette est représenté par la figure II.6.

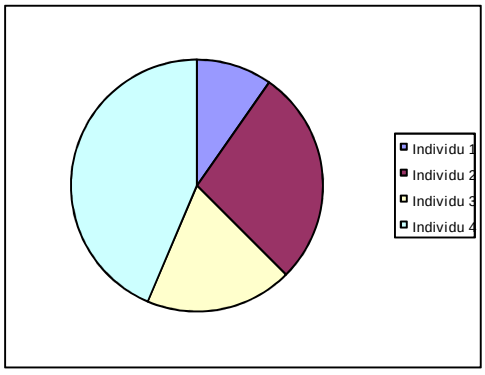


Figure II.6. *Exemple de sélection par roulette*

Ensuite, la bille est lancée et s'arrête sur un chromosome. Cela peut être simulé par la succession des points suivants [61]:

1) On calcule la somme S1 de toutes les valeurs d'évaluation des individus d'une population.

2) On génère un nombre r entre 0 et S1.

3) On calcule ensuite une somme S2 des évaluations en s'arrêtant dès que r est dépassé.

4) Le dernier chromosome dont la fonction d'évaluation vient d'être ajoutée est sélectionné.

Ainsi, le meilleur individu a de grandes chances d'être conservé, mais des individus de performance plus faible peuvent également rester d'une génération à l'autre, ce qui peut aider à conserver la diversité de la population. Cependant, cette technique semble inéquitable lorsque les valeurs d'adaptation des chromosomes varient énormément. Si par exemple, la meilleure valeur d'adaptation d'un chromosome représente 90% de la roulette, les autres chromosomes auront très peu de chance d'être sélectionnés et on arrive à une stagnation de l'évolution.

- **Sélection par rang**

La sélection par rang trie d'abord la population par fitness. Ensuite, elle associé à chaque chromosome un rang suivant sa position. De ce fait, le plus mauvais chromosome aura le rang 1, le suivant 2, et ainsi de suite jusqu'au meilleur chromosome qui aura le plus grand rang et qui égale à la taille de la population. Le principe de cette méthode de sélection est le même que par roulette, mais les proportions sont en relation avec le rang plutôt qu'avec la valeur d'évaluation. Le tableau II.1 présente un exemple de sélection par rang.

Chromosomes	1	2	3	4	5	Total
Probabilités initiales	24 %	16 %	7 %	2 %	51 %	100 %
Rang	4	3	2	1	5	15
Probabilités finales	27 %	20 %	13 %	7 %	33 %	20 %

Tableau II.1. *Exemple de sélection par rang pour 5 chromosomes*

Dans cet exemple et après l'application de la sélection par rang, c'est l'individu ayant comme rang 3 et comme probabilité finale correspondante 20 % qui est sélectionné.

L'avantage majeur de cette méthode de sélection est que, tous les chromosomes ont une chance d'être sélectionnés. Cependant, elle conduit à une convergence plus lente vers la bonne solution, du fait que les meilleurs chromosomes ne diffèrent pas énormément des plus mauvais.

- **Sélection steady-state**

Avec cette méthode, une grande partie de la population peut survivre à la prochaine génération. Tout d'abord, un premier ensemble de chromosomes parents choisis parmi ceux qui ont le meilleur coût, sera utilisé pour créer des chromosomes fils. Ces derniers vont remplacer les chromosomes les plus mauvais pour construire, avec les chromosomes parents, la nouvelle population.

- **Sélection par tournoi**

La sélection par tournoi consiste à choisir aléatoirement un certain nombre d'individus, parmi lesquels celui qui a la plus grande valeur d'adaptation sera sélectionné. Cette étape est répétée autant de fois qu'il y a d'individus à remplacer dans la génération, sachons que, les individus qui participent à un tournoi restent dans la population et sont de nouveau disponibles pour les tournois ultérieurs.

- **Elitisme**

Cette méthode est utilisée pour éviter, lors de la création d'une nouvelle population, la perte de meilleurs chromosomes suite à l'application des opérations d'hybridation et de mutation. Ainsi, un ou plusieurs des meilleurs chromosomes sont copiés dans la nouvelle génération. Ensuite, le reste de la population sera généré selon l'algorithme de reproduction. Cette méthode améliore considérablement les AEs, car elle permet de ne pas perdre les meilleures solutions.

- **Sélection uniforme**

C'est une technique très simple qui consiste à sélectionner un individu aléatoirement de la population. La probabilité P_i pour qu'un individu soit sélectionner est définie par :

$$P_i = 1 / \text{taille-pop}$$

4.2.4. Génération de nouveaux individus

Pour assurer la reproduction de nouveaux individus, on fait appel à des opérateurs génétiques.

a. Croisement (cross-over)

Le croisement est vu comme l'opérateur d'exploitation essentiel des algorithmes évolutionnaires. L'idée générale du croisement est l'échange de matériel génétique entre les parents. Dans ce cas, si les deux parents sont plus performants que la moyenne, on peut espérer que cela est dû à certaines parties de leur génotype, et que certains des enfants recevant les « bonnes » parties de leurs deux parents, n'en seront que plus performants [69]. Tout simplement, cet opérateur consiste à combiner deux génotypes de deux individus pour en obtenir deux nouveaux, en échangeant un ou plusieurs fragments des deux génotypes. On distingue plusieurs croisements possibles, dont les plus utilisés sont :

- **Le croisement en un point:** Pour obtenir les deux nouveaux génotypes, on choisit au hasard un même point de coupure sur ces deux génotypes et on échange les fragments situés après le point de coupure.
- **Le croisement en 2 points:** Dans ce cas, on choisit au hasard deux points de croisement et on échange les fragments situés entre ces deux points.
- **Le croisement en k points :** Cette façon d'hybrider est la généralisation à k points de coupure des méthodes précédentes. Elle est aussi connue par le *croisement multi points*.
- **Le croisement uniforme:** Cette méthode peut être considérée comme un cas particulier du croisement multi points, puisqu'elle consiste à échanger les bits à chaque position indépendamment avec une probabilité de 0.5 ; donc, le nombre de coupures n'est pas connu à priori mais il est déterminé aléatoirement au cours de l'opération. En pratique, pour chaque couple de génotypes, un masque binaire généré aléatoirement et de même longueur que ces génotypes est utilisé. Il permet de conserver les gènes dans les positions correspondantes aux positions contenant des zéros dans le masque. Les autres gènes seront échangés.

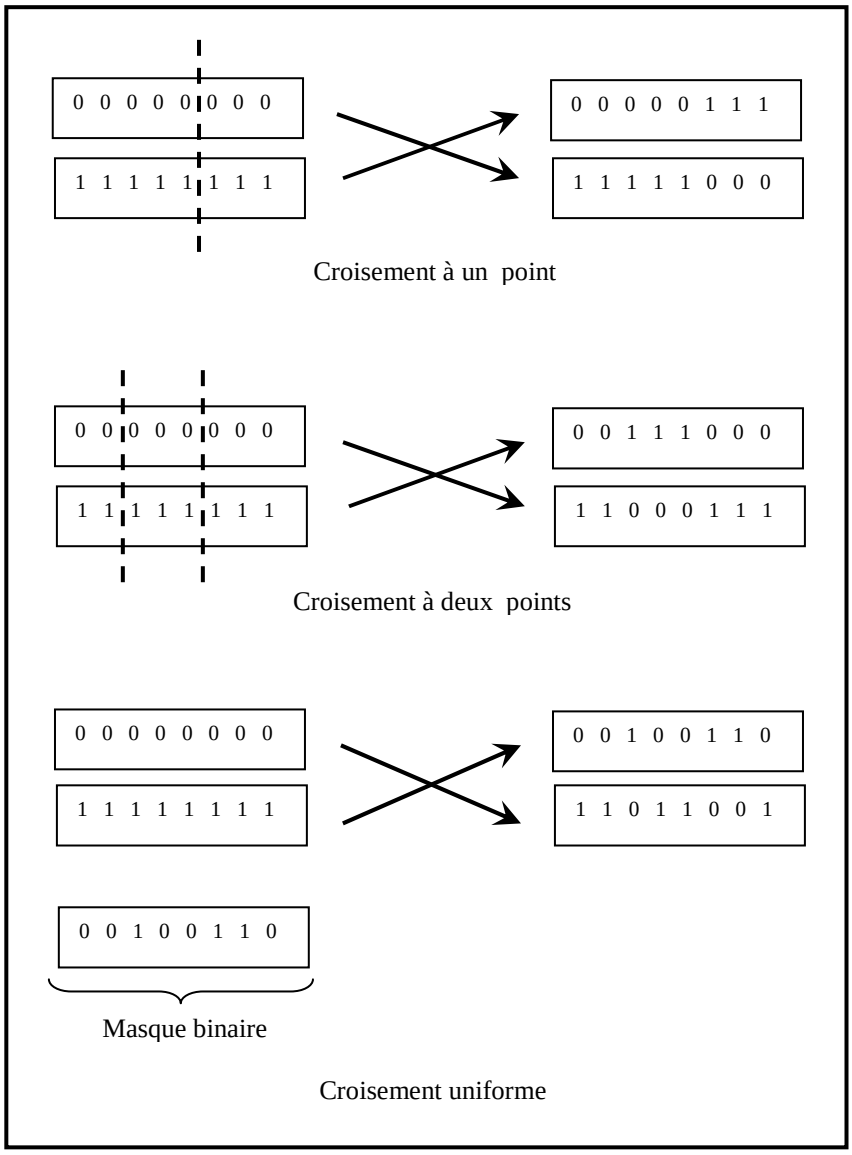


Figure II.7. *Les divers croisements binaires*

b. Mutation

Il s'agit, comme son nom l'indique, d'une modification arbitraire des gènes. Elle est souvent paramétrée par un facteur de mutation qui indique la probabilité pour qu'un gène donné soit muté. Dans le cas des génotypes binaires, une mutation correspond à la transformation d'un 0 en 1 ou vice versa.

L'opérateur de mutation apporte aux algorithmes évolutionnaires l'aléa nécessaire à une exploration efficace de l'espace de recherche en leur dotant de la possibilité d'atteindre tous les points de l'espace d'états, sans pour autant les parcourir tous dans le processus de

résolution. Ainsi, cet opérateur est le responsable directe de la diversité génétique d'une certaine population, et en son absence, aucune caractéristique génétique nouvelle ne peut apparaître ; ce qui revient à dire que les algorithmes évolutionnaires peuvent converger sans croisement, et certaines implantations fonctionnent de cette manière.
Les propriétés de convergence des AEs sont donc fortement dépendantes de cet opérateur qui doit vérifier la condition d'*ergodicité* [69], c'est-à-dire le fait que tout point de l'espace de recherche peut être atteint en un nombre fini de mutations.

Les mutations les plus utilisées sont :

- **La mutation stochastique (Bit Flip) :** C'est la mutation la plus employée dans la représentation binaire. Elle consiste à inverser chaque bit indépendamment avec une probabilité (c / n), tel que : $0 < c \leq n$ et n est la taille du vecteur.
- **La mutation 1 bit :** Inverse le symbole d'un bit choisi au hasard avec une probabilité P_m.

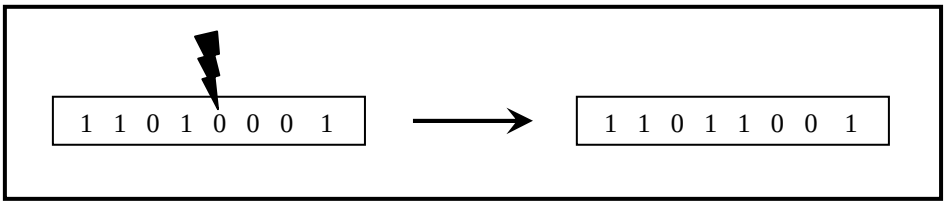


Figure II.8. *La mutation 1 bit : le 5ème bit a été inversé*

Remarque :
Les opérateurs génétiques fonctionnent au niveau génotypique tandis que le mécanisme de sélection opère au niveau phénotypique.

4.2.4 Critère d'arrêt du processus

Arrêter le processus au bon moment est essentiel du point de vue pratique. Si l'on a la valeur exacte de l'optimum recherché, aucun problème ne se pose puisque on peut arrêter le processus dès que cette valeur est atteinte par le meilleur individu de la population courante. Tandis que, en l'absence ou l'insuffisance d'informations sur cette valeur là, il sera délicat de savoir quand arrêter l'évolution. Dans ce cas, une stratégie couramment employée consiste à arrêter l'algorithme dès qu'un nombre maximal d'itérations est atteint, ou qu'un niveau de stagnation est identifié. Donc, il est évident qu'une bonne gestion de l'arrêt de l'évolution contribue de façon importante à l'efficacité de la méthode, et intervient au même titre que le réglage des principaux paramètres de l'algorithme, à savoir la taille de population et les probabilités de croisement et de mutation.

5. Les points clés

5.1. La diversité génétique

Le terme *diversité génétique* désigne la variété des génotypes codant les différents individus formant une population. C'est cette notion qui détermine la *convergence* de l'algorithme au moment où tous les individus deviennent identiques, c-à-d lorsque la diversité génétique devienne nulle. Dans ce cas, ou même au cas où la diversité génétique devient très faible, il sera très difficile voir même impossible qu'elle augmente à nouveau. Et si cela se produit trop tôt, la convergence a lieu vers un optimum local et on parle de *convergence prématurée*. Il faut donc préserver la diversité génétique, sans pour autant empêcher la convergence.

5.2. Le dilemme exploration – exploitation

Le compromis *exploration – exploitation* doit être vérifier à chaque étape de l'algorithme. Cela se donne par une bonne exploration de l'espace de recherche, afin d'éviter le blocage autour d'un optimum local, et par une exploitation des meilleurs individus obtenus, afin d'atteindre de meilleures valeurs aux alentours. Toutefois, trop d'exploitation entraîne une convergence vers un optimum local, alors que trop d'exploration entraîne la non-convergence de l'algorithme.

Typiquement, les opérations de sélection et de croisement sont des étapes d'exploitation, alors que l'initialisation et la mutation sont des étapes d'exploration [69]. On peut ainsi régler les parts respectives d'exploration et d'exploitation en jouant sur les divers paramètres de l'algorithme, à savoir les probabilités d'application des opérateurs génétiques, la pression de sélection, ... Malheureusement, il n'existe pas de règles universelles de réglages et seuls des résultats expérimentaux donnent une idée du comportement des divers composantes des algorithmes.

6. Avantages et limites

6.1. Avantages

Les algorithmes d'optimisation classiques, tel que le recuit simulé, ont du mal à contourner le problème des optima locaux qui leur piègent. Mais les AEs et grâce à leur très large exploration non systématique de l'ensemble des solutions possibles, appelé espace de recherche, dépassent ce problème et trouvent plus rapidement de meilleures solutions. Ceci représente leur principal avantage. Ainsi, ces algorithmes sont un peu les algorithmes de la dernière chance pour trouver de très bonnes solutions à des problèmes dont on a montré

mathématiquement qu'il n'existe aucune autre méthode pour les résoudre que d'explorer combinatoirement toutes les possibilités. Ces derniers sont les problèmes dits NP-complets ou NP-difficiles.

Leur autre avantage est de pouvoir résoudre des problèmes difficiles à exprimer mathématiquement, ce qui est habituellement le cas des problèmes inverses [70]. En effet, les algorithmes classiques optimisent les paramètres de problèmes mis préalablement en équations mathématiques. Dans le cas des AEs, ceci n'est pas nécessaire puisque l'évaluation des individus créés peut se faire par comparaison avec un résultat recherché, ou, cas extrême, en demandant son avis à l'utilisateur sur la qualité de la solution trouvée.

6.2. Limites

Malgré l'attirante simplicité des notions et du principe, de façon générale, des AEs, la conception et la réalisation de tels algorithmes efficaces restent des tâches difficiles vu leur sensibilité aux choix algorithmiques et paramétriques. De plus, elles nécessitent une très bonne connaissance du problème à traiter, beaucoup de créativité, et une compréhension fine des mécanismes évolutionnaires.

Les principales difficultés et limites des AEs peuvent être regroupées dans les points suivants :

- ✓ Il faut adapter les données du problème à l'algorithme, ce qui nécessite l'ajout de deux étapes supplémentaires. La première est l'étape de pré-traitement dans laquelle une fonction de codage est crée et la deuxième étape est celle de post-traitement qui consiste à créer une fonction de décodage.
- ✓ Difficulté de réutilisation des programmes déjà créés puisque chaque algorithme est très dépendant du problème. De plus, les concepteurs de tels systèmes ne sont souvent pas les utilisateurs, ce qui limite leur bonne utilisation.
- ✓ Temps de calcul pouvant être important, ce qui rend l'essai de plusieurs valeurs de paramètres plus dur.
- ✓ Les nombreux paramètres contrôlant ces algorithmes, tels que les probabilités de croisement et de mutation, sont délicats à régler. De même, le codage des chromosomes peut faire varier radicalement la vitesse de convergence.
- ✓ Afin de garantir la robustesse des AEs, le calcul d'un très grand nombre de fitness est généralement nécessaire avant l'obtention d'une bonne solution. Ce nombre de calcul important peut devenir, lui-même, un problème lorsque le coût de calcul, en termes de

ressources systèmes ou temporelles, de la fitness est important. Cela se produit, par exemple, lorsqu'on travaille en grande dimension sur des fonctions à complexité importante.

7. Parallélisassion des algorithmes évolutionnaires

Malgré que le grand temps de calcul représente un obstacle lors de l'utilisation des AEs, il est, toutefois, possible de l'améliorer puisque l'étape la plus coûteuse qui est l'évaluation de la performance de toute une population, n'est à la fin du compte qu'un ensemble de calculs totalement indépendants entre eux, qui peuvent donc être exécuter en parallèle. Ceci n'est pas l'unique manière d'envisager la parallélisassion des AEs, mais il existe dans la littérature une gamme complète de manières de paralléliser ces algorithmes que nous présenterons dans la section qui suit.

Une quantité clé pour le choix de la méthode de parallélisassion comme pour le calcul du gain que celle-ci apporte est le *ratio évaluation-évolution*, qui est le rapport entre le temps moyen pour un calcul de performance et le temps moyen nécessaire pour la sélection, l'application des opérateurs génétiques et le remplacement d'un individu [69]. On parle de temps moyen, car ces opérations sont généralement effectuées sur des groupes d'individus.

7.1. Parallélisassion du calcul de performance

Pour exécuter cette technique, il suffit de se disposer d'un algorithme parallèle de calcul de la fonction d'évaluation qui remplacera le calcul séquentiel. De ce fait, le gain espéré dépend totalement du problème.

7.2. Distribution du calcul de performance

7.2.1. Distribution synchrone

a. Principe

Cette technique consiste à exécuter le calcul de la fonction d'évaluation de façon parallèle. Il suffit, donc, de lancer en parallèle sur des stations esclaves le calcul de la fonction d'évaluation de plusieurs individus. Ensuite, les résultats seront récupérés par une station maître qui gère l'algorithme évolutionnaire standard. Le comportement de l'algorithme parallèle est alors exactement celui de l'algorithme séquentiel, et le facteur d'accélération devient rapidement très proche du nombre de processeurs lorsque le ratio évaluation-évolution

augmente, pour des tailles de population ne dépassant pas quelques multiples du nombre de processeurs disponibles [69].

b. Avantage

L'unique avantage de cette tentative de parallélisassion est qu'elle est simple à mettre en œuvre.

c. Inconvénients

Le gain apporté par ce schéma de parallélisassion peut se dégrader lorsque les durées des calculs de performance ne sont pas les mêmes. Ceci est la conséquence soit des conditions différentes de calcul ou de l'hétérogénéité des processeurs utilisés. Donc, les processeurs doivent attendre le plus lent d'entre eux. Pour limiter ce problème, une gestion plus fine de la distribution des calculs est nécessaire, où une liste de calculs à effectuer est affectée à chacun des processeurs. Ainsi, un processeur, et dès qu'il termine un calcul, reçoive un autre. Il est à signaler aussi que la tolérance aux pannes, avec cette technique, est nulle puisque si un processeur s'arrête pour une raison quelconque, l'ensemble de l'algorithme se met en attente du résultat manquant.

7.2.2. Distribution asynchrone

a. Principe

Cette technique représente une solution aux problèmes de la technique précédente. Elle consiste à utiliser un schéma d'évolution asynchrone où chaque processeur esclave effectue son calcul de performance, renvoie le résultat au maître et reçoit immédiatement un nouveau calcul. Toutefois, cela suppose que les étapes de sélection / remplacement peuvent être effectuée individu par individu.

b. Inconvénients

Lorsque la durée du calcul de performance de certains individus sera un peu grande à cause, par exemple, des processeurs moins rapides que les autres processeurs, les individus en question risquent d'être démodés et en particulier d'être éliminés rapidement, avant d'avoir été sélectionnés pour reproduction. Toutefois, l'inconvénient majeur de ce modèle est l'accumulation de communications qui se produit au noeud maître dés que le nombre de processeurs augmente, et si le temps ratio évaluation-évolution n'est pas suffisant, les esclaves doivent alors attendre leur tour pour recevoir le calcul suivant.

7.2.3. Ni Dieu ni maître

a. Principe

Ce modèle représente une version tolérante aux pannes, malgré qu'ici aussi la population soit toujours centralisée. L'originalité de ce modèle réside dans le fait que la population n'est plus maintenue physiquement sur un processeur, mais elle est stockée dans un fichier auquel tous les processeurs peuvent accéder pour accomplir les opérations du processus évolutionniste, à savoir la sélection, l'application des opérateurs génétiques, l'évaluation et le remplacement. Ceci revient soit à lire ou à écrire dans le fichier de la population. Donc, la pertinence d'application de ce modèle dépend du temps de calcul de la performance, où il donne de bons résultats lorsque ce dernier sera important.

b. Inconvénients

Le premier inconvénient de ce modèle est le temps important nécessaire aux : ouvertures /lectures /fermetures du fichier. L'autre inconvénient se résume dans le fait qu'un seul processeur peut écrire en même temps.

7.3. Modèle en îlots

a. Principe

Avec ce modèle la population est divisée en petites sous-population, évoluant chacune sur un processeur suivant un schéma traditionnel auquel vient s'ajouter une étape de **migration**. Ainsi, chaque sous-population envoie ses meilleurs individus soit vers les populations voisines soit dans un « pool » commun [69]. Ce choix dépend des coûts relatifs des communications entre processeurs. Ensuite, chaque sous-population reçoit des individus soit envoyés par ses voisins soit péchés dans le « pool » central.

b. Points forts

Une des possibilités offertes par ce modèle en îlots est que chaque sous-population évolue en utilisant des paramètres différents [69] (voir Figure II.9). D'autre part, les bons paramètres peuvent être différents à différentes étapes de l'évolution, et on peut espérer qu'une des sous-populations au moins aura des paramètres bien réglés à tout moment. En envoyant ses meilleurs individus aux autres sous-populations, elle tirera l'ensemble des individus dans le bon sens.

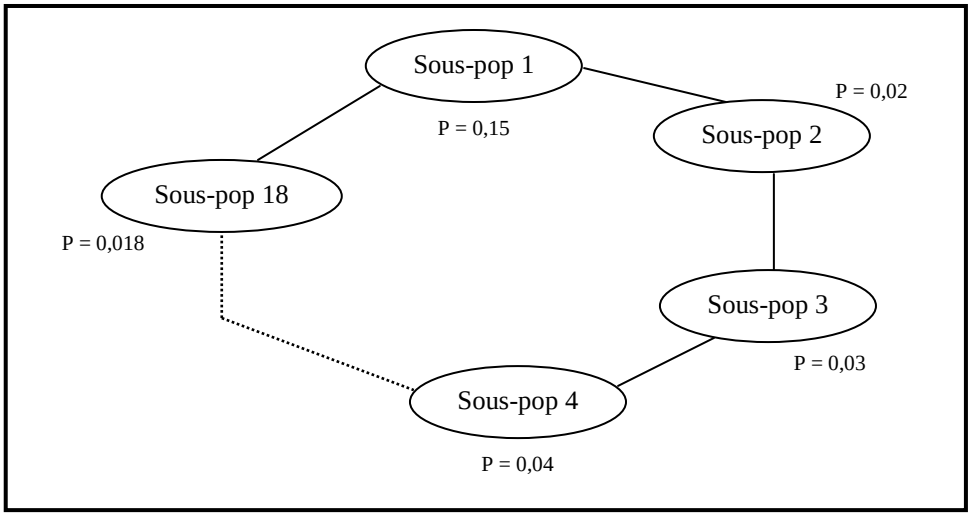


Figure II.9. *Exemple de modèle en îlots*
(Le paramètre de mutation est différent à chaque nœud)

c. Difficultés

Ce modèle ne représente pas une simple parallélisation de l'algorithme séquentiel, mais c'est un algorithme avec une dynamique totalement différente. Ainsi, les comparaisons avec les versions séquentielles seront difficiles, et les tentatives d'estimation du gain seront biaisées même si les sous-populations utilisent exactement les mêmes paramètres et le même mode de migration (c'est-à-dire entre voisins uniquement, ou avec un échange généralisé).

7.4. Population distribuée

Ce modèle génère lui aussi une nouvelle dynamique de l'algorithme, différente des précédentes, du fait qu'il consiste à distribuer une unique population sur l'ensemble des processeurs et en général sur des machines massivement parallèles. Ainsi, les opérations de sélection, de remplacement et de croisement se font entre individus voisins pour la topologie du réseau de processeurs. Sachons, toutefois qu'il y a dans ce modèle un processeur superviseur qui reçoit des informations de l'ensemble des processeurs lui permettant, ainsi, de suivre précisément l'évolution de l'ensemble et de décider du moment de l'arrêt de l'algorithme.

7.5. Quel est le modèle à choisir?

Les modèles de parallélisassion présentés ci-dessus ne sont certainement pas les seuls et d'autres modèles peuvent être inventés. Toutefois, les deux modèles en îlots et le modèle de

la population distribuée se sont montrés souvent plus performants que l'algorithme séquentiel puisqu'ils assurent un gain en temps de calcul dû à l'accroissement de ressources.

Le choix d'un modèle particulier dépend, essentiellement, du problème, de la configuration matérielle et du temps dont on dispose pour l'implantation. Ainsi [69] :

- ✓ Si la fonction d'évaluation ne peut être parallèle, il est évident que le modèle le plus simple à implanter est celui consistant à distribuer le calcul de cette fonction là, puisqu'il n'y a pas de paramètres supplémentaires à ajuster.
- ✓ Si l'on dispose d'un réseau de processeurs homogènes ayant, de plus, un temps de calcul équivalent, alors le temps de calcul sera plus ou mois amélioré selon le nombre de processeurs disponibles. Mais le problème qui peut se poser, est la saturation du nœud maître lorsque le nombre de processeurs devient grand en fonction du temps de calcul de performance.
- ✓ Le modèle en îlots sera le plus adéquat lorsque le nombre de processeurs disponibles augmente malgré qu'il nécessite un effort supplémentaire puisqu'il faut ajouter les étapes de migration à la boucle de génération classique, en plus du temps de prise en main non négligeable dû au fait que l'algorithme est totalement différent ; sans compter les choix de paramètres (stratégie et fréquence de migration). Malgré tout ça, les gains espérés peuvent être plus importants à cause de l'utilisation de différents jeux de paramètres d'évolution aux différents nœuds en plus de la dynamique différente qui permettent de préserver plus longtemps une plus grande diversité génétique.

8. Conclusion

Un oiseau ne connaît pas les équations de l'aérodynamique, pourtant il est capable de voler. Son corps, particulièrement adapté au vol, et son comportement ont été conçus au fil des générations par sélection successives des individus les plus performants. La nature a exploité une stratégie basée sur l'apprentissage par essais et erreurs qui peut être exploitée par les ingénieurs pour résoudre les problèmes auxquels ils sont confrontés [68].

Et si la connaissance précise des équations d'un système demeure fondamentale pour maîtriser pleinement un processus, les algorithmes évolutionnaires dont nous avons présenté ici les principes, peuvent fournir un outil d'exploration intéressant dans des domaines partiellement ou totalement inconnus. Ils peuvent aider à étudier le système en trouvant des solutions astucieuses exploitant des caractéristiques spécifiques dont les justifications théoriques peuvent n'apparaître que bien plus tard.

Dans ce chapitre, nous avons présenté en détail les techniques de base pour réaliser un algorithme évolutionnaire simple mais efficace, très robuste et général. Ses opérateurs sont la simplicité même, ne mettent en jeu des procédures aussi peu complexes que la génération de nombres aléatoires, la copie de chaînes, et les échanges de morceaux de chaînes ; cependant, malgré leur simplicité, la performance de l'optimisation qui en résulte est impressionnante. Mais n'oublions pas qu'il existe un compromis généralité / efficacité. Et pour obtenir des performances optimales, il peut être souhaitable d'adapter l'algorithme au problème traité, en introduisant des méthodes spécifiques, et si possible des connaissances sur le problème.

Les règles de transition des AEs sont stochastiques, alors que la plupart des autres méthodes ont des règles de transition déterministes. Il y a cependant une différence entre les opérateurs pseudo-aléatoires des AEs et des autres méthodes pour lesquelles seul le hasard intervient pour guider l'exploration. Les AEs utilisent le hasard pour guider une exploration qui tire grandement parti de l'information disponible. Il peut paraître étrange d'utiliser le hasard pour parvenir aux résultats souhaités, qui sont les meilleurs points, mais la nature regorge d'exemples de tels systèmes. Et c'est d'ailleurs l'une des raisons qui nous a motivée pour exploiter ces algorithmes dans la phase principale de la cryptographie soit le chiffrement.

Enfin, les AEs peuvent être utilisés à tous les stades : recherche, développement, production... En effet, il est tout aussi nécessaire de concevoir un dispositif ayant des caractéristiques optimales, qu'un dispositif ayant une faisabilité optimale, un coût optimal... Ils constituent un domaine de recherche très actif, d'une part de par leur intérêt propre, d'autre part parce qu'ils rejoignent les enjeux industriel et économique actuels.

Chapitre III

Algorithme de Chiffrement Evolutionnaire basé Occurrences (ACEO)

1. Introduction

Notre intérêt pour les algorithmes évolutionnaires comme approche de chiffrement a été motivé, en premier lieu, par le grand besoin en sécurisation de données, vue que les vrais algorithmes de chiffrement sont peu nombreux ; et par l'adaptabilité et l'efficacité d'application de ces algorithmes dans un tel domaine à cause de leur large utilisation de l'aléatoire.

2. Mise en œuvre de l'Algorithme de Chiffrement Evolutionnaire basé Occurrences (ACEO)

À ce niveau, nous allons présenter les deux phases essentielles de l'**ACEO**: le chiffrement et le déchiffrement.

2.1. Chiffrement

Les algorithmes évolutionnistes se présentent en plusieurs versions. Celle que nous avons choisie pour résoudre notre problème regroupe les étapes suivantes:

1) Définition d'un codage du problème,
2) création de la population initiale,
3) sélection parmi les parents (individus de la population courante) ceux qui vont avoir des enfants,
4) application des opérateurs de variation (croisement / mutation) aux parents sélectionnés (génération des enfants),
5) évaluation des performances des individus,

6) sélection, parmi les parents et les enfants, de ceux qui vont survivre à la génération suivante.

Ce processus est réitéré jusqu'à la satisfaction d'un critère d'arrêt.

Pour appliquer l'approche évolutionnaire à notre problème, nous présenterons les choix utilisés en terme de : représentation chromosomale des solutions du problème, méthode de création de la population initiale des solutions, fonction d'évaluation, opérateurs génétiques et procédure de sélection. Pour mieux approcher le problème, nous présentons une formalisation de ce dernier.

Formalisation du problème

Soit *Mess* le message à chiffrer qui est constitué d'une suite de caractères appartenant à la liste des 149 caractères possibles et regroupés dans la table TCAR (Tableau III.1). Ainsi, *Mess* peut être structuré comme suit :

$$Mess = c_1, c_2, \ldots\ldots, c_n$$

Où n désigne la longueur du message *Mess*.

Ce mode de représentation (structure) permet de distinguer les différents caractères formant ce message, et par conséquent de déterminer le nombre d'occurrences de chacun d'eux. L'opération de chiffrement consiste à perturber le message en clair de telle manière à avoir le maximum de désordre dans le message chiffré. Pour se faire, nous avons choisi de jouer sur les nombres d'occurrences des caractères, pour avoir la plus grande différence entre les nombres d'occurrences des 149 caractères admissibles dans le message en clair et leur nombre d'occurrence dans le message chiffré, grâce au codage que nous présenterons dans le paragraphe suivant. Ce choix est dû au fait que cette unique donnée (nombres d'occurrences) ne représente pas une information pertinente pour les cryptanalystes. Donc, le problème ainsi vue ou posé correspond à un problème d'optimisation.

Rang	1	2	3	4	5	6	7	8	9	10	11	12	13
Caractères	a	b	c	d	e	f	g	h	i	j	k	l	m

14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
n	o	p	q	r	s	t	u	v	w	x	y	z	0	1	2

30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
3	4	5	6	7	8	9	!	«	#	$	©	«	%	&	¢

46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61
£	¤	¥	¦	§	¬	®	±	»	¸	¶	´	µ	²	³	¹

62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77
¨	،	؛	(	)	*	+	,	-	.	/	:	;	<	=	>

78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93
?	@		[	\	]	^	_	`	{	|	}	~	▯	€	ƒ

94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109
÷	ً	ي	و	ه	ن	م	ل	ك	ق	ف	غ	ع	ظ	ط	ض

110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125
ص	ش	س	ز	ر	ذ	د	خ	ح	ج	ث	ت	ب	ئ	إ	ؤ

126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141
ة	أ	ا	ء	ى	؟	ù	ú	û	ü	ô	î	ï	ç	è	é

142	143	144	145	146	147	148	149
ê	ë	á	â	¼	½	¾	˚

Tableau III.1. *La table TCAR*

Dans ce qui suit, nous présentons les différentes étapes de l'algorithme illustrées d'exemples. Dans tous les exemples qui seront donnés, le message clair (M) représente un extrait d'un article écrit dans le journal quotidien algérien « El khabar» :

كشف الرئيس الإيراني محمود أحمدي نجاد أمس الإثنين 10 أفريل 2007 عن المفاجأة التي كان قد وعد بها, حيث أكد أن إيران دخلت مرحلة الإنتاج الصناعي للوقود النووي. و في كلمة ألقاها في مصنع نطنز لتخصيب اليورانيوم بمناسبة يوم الطاقة النووية الإيراني، قال نجاد إن إيران دخلت النادي النووي العالمي. و هو إنجاز نقدمه إلى قائد الثورة الإسلامية والشعب الإيراني و كل الشعوب الحرة في العالم. و أوضح أن هذا الإنجاز سيساهم في توفير 20 ألف ميغاواط سنويا من الطاقة التي تساعد في المجالات الزراعية و إنتاج الكهرباء و كذلك الطب. مبرزا حق بلاده في إمتلاك الطاقة النووية حتى النهاية، و أنها لن تسمح للدول الكبرى بوقف برنامجها النووي..

2.1.1. Le codage

Le codage dépend étroitement du problème à résoudre. En effet sa définition permet de cerner l'espace des solutions possibles. Ce codage doit, de plus, être aussi compact que possible pour permettre une évolution rapide. Et si le codage binaire semble tout à fait adapté pour certains problèmes d'optimisation, tel le problème des N-dames, cette représentation semble peu appropriée car non intuitive. La représentation qui nous semble la plus naturelle est le codage entier, puisque toutes les opérations de l'algorithme vont manipuler des nombres d'occurrences qui sont des nombres entiers.

Dans le cas de notre problème, l'opération de codage consiste à transformer le message en code particulier en calculant le nombre d'occurrence dans le message des 149 caractères admissibles et l'attribuer à la case qui correspond à son rang dans une table, désignée par TCARDESORDON, regroupant les 149 caractères de la table TCAR rangés aléatoirement. Le contenu de TCARDESORDON change à chaque nouvelle instance du problème afin d'augmenter la confusion de l'algorithme. La pertinence de cette opération va se démontrer au fur et à mesure de l'avancement dans la présentation des autres étapes de l'algorithme. Un exemple du contenu de TCARDESORDON est donné par le tableau III.2.

Rang	1	2	3	4	5	6	7	8	9	10	11	12	13
Caractères	,	ب	>	؛	z	-	:	€	c	ù	h	o	*

14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
"	e	r	¦	á	f	²	{	ذ	»	ç	ل	ِ	n	8	`

30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
ك	ض	¨	[	أ	q	س	â	x	a	ر	ï	>	m	ز	ء

46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61
&	é	ئ	u	k	û		(	5	^	ص	ّ	b	2	j	~

62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77
ق	ق	¤	I	£	¬	ط	_	¾	§	ة	»	¹	4	[	ش

78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93
v	3	)	©	غ	ه	s		è	#	7	ê	د	؟	p	ف

94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109
=	ج	¥	آ	ى	÷	I	¢	،	ا	ë	.	+	½	³	ع

110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125
6	0	!	±	}	ي	م	%	;	w	¦	t	®		¼	\

126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141
ث	ح	و	ت	$	@	|	l	µ	´	g	y	d	ن	/	l

142	143	144	145	146	147	148	149
خ	ü	?	ظ	9	ٔ	ؤ	¶

Tableau III.2. *Exemple du contenu de TCARDESORDON*

Dans ce qui suit, on désigne par individu le message à chiffrer ou tout autre message, et par chromosome tout individu ayant subi une telle opération de codage. (Figure III.1 [71])

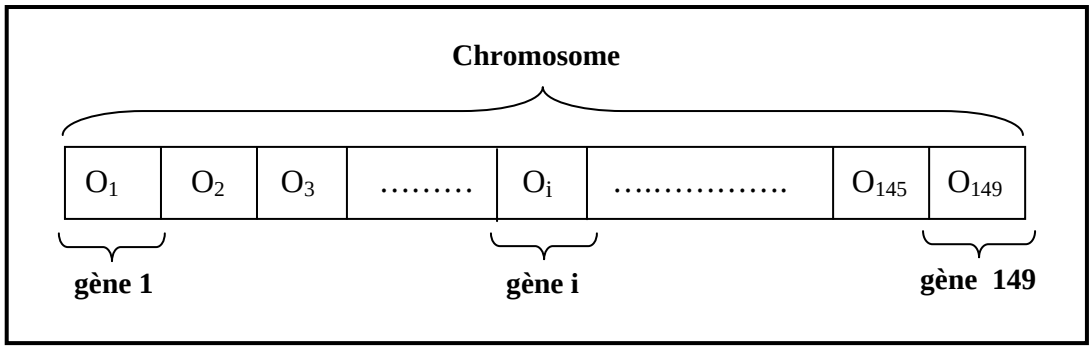


Figure III.1. *Codage des individus*

Où $i = \overline{1-149}$ est le rang des caractères dans la table TCARDESORDON et O_i le nombre d'occurrences dans le massage du caractère ayant comme rang i.

Remarques :

1) $\sum_{i=1}^{149} O_i = n$ Où $n \in N$ représente la longueur du message.

2) Les caractères qui ne figurent pas dans le message auront une occurrence nulle.

2.1.2. Création de la population initiale

Dans un algorithme évolutionnaire, la première génération d'individus peut être construite de trois façons :

- ✓ soit elle est créée aléatoirement ;
- ✓ soit elle est donnée par l'utilisateur ;
- ✓ ou bien, une partie est créée et l'autre est donnée (pour introduire dès le début de bonnes solutions construites, par exemple, à partir de certaines heuristiques).

Dans notre cas, les m individus (I_1, I_2,…, I_m) formant la population initiale sont obtenus par application de m perturbations aléatoires sur le chromosome initial I_0. Ce dernier est le codage du message en clair suivant l'ordre des caractères dans TCARDESORDON. Cette table (TCARDESORDON) est gardée secrète afin d'augmenter la complexité de la tâche des cryptanalystes. Ainsi, s'ils réussissent à trouver l'ordre des caractères dans TCARDESORDON, et par la suite de dégager la façon de construire le message chiffré en cours et finalement de trouver la bonne combinaison des nombres d'occurrences qui forme son codage ; ces informations ne seront plus utiles car la répartition des caractères dans le codage du message initial se diffère d'une instanciation du problème à une autre.

Exemple :

I_0 Chromosome initial :

Le tableau ci-dessous représente le chromosome initial correspondant au message M. Il regroupe 149 cases qui correspondent aux 149 gènes. Ainsi, il se lit de gauche à droite, ligne après ligne.

1	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	2	0	0	0	0	58	0	0	0	0	11	0	0
1	10	0	9	0	0	0	17	0	0	0	1	5	0	0	2
0	0	0	0	0	0	3	0	0	2	0	0	11	0	0	0
0	14	3	0	0	0	111	0	0	0	0	0	6	0	11	1
0	0	0	0	0	0	0	1	0	0	16	0	3	0	0	9
0	12	0	0	0	79	2	0	6	0	0	23	40	0	0	0
0	4	10	0	0	0	0	14	0	0	0	0	0	13	34	9
3	0	0	0	0	0	0	0	0	0	3	1	0	37	0	0
0	0	0	0	0											

Tableau III.3. *Chromosome initial I_0*

Population initiale :

Nous présentons seulement un exemple d'un seul individu parmi ceux qui forment la population initiale d'une taille quelconque. C'est celui donné à travers le tableau III.4. Cet individu est le résultat d'une perturbation du chromosome I_0 (Tableau III.3), qui se traduit par une simple permutation entre deux gènes de I_0 ($4^{ème}$ gène et le gène numéro 31 de I_0). Ce mode de permutation permet de garantir une évolution progressive vers la solution finale dans le but d'éviter toute convergence prématurée.

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	2	0	0	0	0	58	0	0	0	0	11	13	0
1	10	0	9	0	0	0	17	0	0	0	1	5	0	0	2
0	0	0	0	0	0	3	0	0	2	0	0	11	0	0	0
0	14	3	0	0	0	111	0	0	0	0	0	6	0	11	1
0	0	0	0	0	0	0	1	0	0	16	0	3	0	0	9
0	12	0	0	0	79	2	0	6	0	0	23	40	0	0	0
0	4	10	0	0	0	0	14	0	0	0	0	0	13	34	9
3	0	0	0	0	0	0	0	0	0	3	1	0	37	0	0
0	0	0	0	0											

Tableau III.4. *Exemple d'un individu de la population initiale*

2.1.3. Application des opérateurs de variation : Croisement et mutation

Le choix de ces opérateurs est crucial vu leur sensibilité.

- **Le croisement :**

L'opérateur de croisement a pour but d'enrichir la diversité de la population, en manipulant la structure des chromosomes. Classiquement, les croisements sont réalisés à partir de deux parents et génèrent deux enfants. Une description plus détaillée a été exposée dans le deuxième chapitre qui a été consacré aux algorithmes évolutionnaires.

Comme opérateur de croisement, notre choix s'est porté sur l'opérateur OX « Order Cross-over » proposé par Davis [72]. C'est un opérateur à deux points de croisement. Il consiste à générer des descendants en trois phases :

1**)** Choisir dans les deux parents une sous-séquence interne, comprise entre deux points de coupure tirés aléatoirement.

$Parent_1$: 1 3 5 7 9 | **2 4 6** | 8 10

$Parent_2$: 10 1 9 2 8 | **7 3 4** | 6 5

2) Recopier la sous-séquence interne du Parent1 dans le descendant Enfant1 aux mêmes positions et retirer du chromosome Parent2 les allèles compris dans cette sous-séquence.

$Enfant_1$: • • • • • | **2 4 6** | • •

$Parent_2$: 10 1 9 • 8 | **7 3** • | • 5

Le chromosome Parent2 permet alors de former une séquence d'allèles résiduelles en partant du deuxième point de coupure et en considérant le chromosome comme une chaîne circulaire.

3) compléter les gènes disponibles du descendant Enfant1 en lui transmettant dans l'ordre les allèles issus de la séquence résiduelle précédente.

$Enfant_1$:	1	9	8	7	3	|	**2**	**4**	**6**	|	5	10
$Parent_2$:	10	1	9	•	8	|	**7**	**3**	•	|	•	5

Les deux points de croisement sont choisis aléatoirement. Le meilleur taux de croisement varie entre 60% et 100% [73].

Il est important aussi de noter que nous avons essayé d'utiliser un croisement simple en un point, mais dans ce cas l'algorithme converge après quelques générations seulement (de l'ordre de 10). Cela va aider les cryptanalystes de l'attaquer facilement, puisqu'il sera très facile de retrouver les transitions chromosomales menant à la solution finale en appliquant une attaque exhaustive dans le pire des cas (même celle-ci ne va pas être suffisamment gourmande en espace et surtout en temps de calcul à cause du petit nombre de générations menant à la solution).

- **La mutation :**

Pratiquement, le rôle de la mutation consiste à faire apparaître de nouveaux gènes. Cet opérateur introduit une diversité nécessaire à l'exploration de l'espace de recherche. Les mutations jouent, alors, le rôle de bruit et empêchent l'évolution de se figer.

Pour notre problème, nous avons opté pour une simple mutation, celle qui consiste à permuter aléatoirement deux gènes d'un chromosome. Le meilleur taux varie entre 0.1% et 5% [73].

Les individus résultants de ces deux opérations (croisement et mutation) seront rajoutés à la population des parents pour les acheminer vers l'étape suivante qui est celle d'évaluation.

2.1.4. Evaluation des individus

La fonction d'évaluation (ou d'adaptation) quantifie la qualité de chaque chromosome par rapport au problème. Les chromosomes ayant une bonne qualité ont plus de chance d'être sélectionnés pour la reproduction, et donc plus de chance pour que la population suivante hérite de leur matériel génétique. La fonction d'adaptation produit la pression qui permet de faire évoluer la population de l'algorithme évolutionnaire vers les individus de meilleure

qualité. En clair, le choix de la fonction d'évaluation va fortement influer sur le succès de l'algorithme.

La fonction d'évaluation que nous avons définie pour associer des valeurs d'adaptation à chaque individu I_i de la population Pop, est :

$$F(I_i) = \sum_{j=1}^{149} \left| O_{j_i} - O_{j_0} \right|$$

Avec : $I_i = [O_1, O_2, \ldots, O_{149}]$, Pop = $\{I_1, I_2, \ldots, I_m\}$ où m un paramètre de réglage et O_{j_i} est le $j^{ème}$ gène du $i^{ème}$ individu.

2.1.5. Sélection des individus les plus adaptés

Le rôle de la sélection est de distinguer entre les individus sur la base de leur qualité, en particulier, pour permettre aux meilleurs individus de devenir parents dans la génération suivante. Ainsi, elle est responsable sur le fait de pousser l'amélioration de la qualité.
Dans notre cas, nous avons utilisé la méthode de la sélection par roulette, qui a été décrite dans la section 4.2.3 du chapitre II.

2.1.6. Critère d'arrêt

Les étapes de reproduction, d'évaluation et de sélection sont répétées jusqu'à ce que l'optimum recherché soit trouvé. Ce dernier représente l'individu ayant la plus grande valeur de convergence durant tout le processus évolutionnaire. Donc, c'est le message le plus différent du message clair et qui a été retrouvé durant une instance du problème. Toutefois, il est important de rappeler qu'aucune garantie n'est donnée sur la qualité de cette solution, donc elle peut ne pas être l'optimale, mais c'est la meilleure solution retrouvée.

La condition d'arrêt assurant la convergence de notre algorithme évolutionnaire est exprimée à l'aide de la fonction F comme suit :

$$F(I_i) = \sum_{j=1}^{149} \left| O_{j_i} - O_{j_0} \right| \leq 2 * \sum_{j=1}^{149} O_{j_i} \qquad \ldots\ldots. \text{ (I)}$$

pour un : $I_i = [O_1, O_2, \ldots, O_{149}]$ et $I_i \in$ Pop = $\{I_1, I_2, \ldots, I_m\}$

Preuve :

$$F(I_i) = \sum_{j=1}^{149} \left| O_{j_i} - O_{j_0} \right| \leq \sum_{j=1}^{149} (O_{j_i} + O_{j_0}) \quad \text{..... (1)}$$

$$\leq \sum_{j=1}^{149} O_{j_i} + \sum_{j=1}^{149} O_{j_0}$$

Et comme :

$$\forall I_i, I_k \in Pop : \sum_{j=1}^{149} O_{j_i} = \sum_{j=1}^{149} O_{j_k}$$

Et :

$$\forall I_i \in Pop : \sum_{j=1}^{149} O_{j_i} = \sum_{j=1}^{149} O_{j_0}$$

Donc :

$$(1) \Rightarrow \sum_{j=1}^{149} \left| O_{j_i} - O_{j_0} \right| \leq \sum_{j=1}^{149} O_{j_i} + \sum_{j=1}^{149} O_{j_i}$$

$$\leq 2 * \sum_{j=1}^{149} O_{j_i}$$

Exemple :

Nous essayons d'expliciter cette condition à travers un exemple explicatif en utilisant trois individus I_0, I_1 et I_2. Nous vérifions tout d'abord en (1) la condition pour I_0 et I_1, puis en (2) pour I_0 et I_2.

1)

I_0

10	0	5	0	0	3

I_1

0	10	0	3	5	0

$$\Rightarrow \sum_{j=1}^{6} O_{j_1} = \sum_{j=1}^{6} O_{j_0} = 18$$

$$F(I_1) = \sum_{j=1}^{6} \left| O_{j_1} - O_{j_0} \right|$$

$$= |10-0| + |0-10| + |5-0| + |0-3| + |0-5| + |3-0|$$

$$= 10 + 10 + 5 + 3 + 5 + 3$$

$$= 36$$

$$\leq 2 * \sum_{j=1}^{6} O_{j_1}$$

2)

I_0

10	0	5	0	0	3

I_2

0	5	10	3	0	0

$$\Rightarrow \sum_{j=1}^{6} O_{j_2} = \sum_{j=1}^{6} O_{j_0} = 18$$

$$F(I_2) = \sum_{j=1}^{6} \left| O_{j_2} - O_{j_0} \right|$$

$$= |10-0| + |0-5| + |5-10| + |0-3| + |0-0| + |3-0|$$

$$= 10 + 5 + 5 + 3 + 0 + 3$$

$$= 26$$

$$\leq 2 * \sum_{j=1}^{6} O_{j_2}$$

Dans les deux cas (1) et (2), nous avons pris à titre d'exemple un individu constitué de six (06) gènes.

Après quelques tests d'exécution, nous avons constaté que le fait de prendre cette condition uniquement comme condition d'arrêt peut poser le problème de boucle infinie du moment où la valeur de convergence maximale devient inchangée d'une itération à une autre tout en vérifiant la condition (I). Pour remédier à ce problème, nous avons pensé à fixer expérimentalement un nombre maximum d'itérations (de générations) à ne pas dépasser. Ainsi et suite à plusieurs exécutions, nous avons arrivé à déterminer ce nombre :

MaxGen = 100

La condition d'arrêt finalement adoptée englobe les deux conditions suivantes :

1) $F(I_i) = \sum_{j=1}^{149} \left| O_{j_i} - O_{j_0} \right| \leq 2 * \sum_{j=1}^{149} O_{j_i}$

2) MaxGen = 100

2.2. Déchiffrement

Le déchiffrement est l'opération inverse du chiffrement. Elle permet d'obtenir la version originale d'un message qui a été précédemment chiffré.

Dans la deuxième phase de l'algorithme correspondant à cette opération, nous exploitant deux informations qui sont le message en clair et le message chiffré pour générer une **clé de session**

qui peut être d'un usage symétrique ou asymétrique. Cette clé représente la permutation des positions des nombres d'occurrences des caractères du message chiffré pour obtenir ceux des caractères du message en clair. De ce fait, elle varie d'un message à l'autre puisqu'elle dépend du message. Et ce n'est que par l'introduction de la clé appropriée que les caractères du message chiffré rejoignent leurs positions initiales pour retrouver le message en clair.

Remarque :

La notion de *clé de session* est un compromis entre le chiffrement symétrique et asymétrique permettant de combiner les deux techniques. Son principe est simple : il consiste à générer aléatoirement une clé de session de taille raisonnable, et de chiffrer celle-ci à l'aide d'un algorithme de chiffrement à clef publique (plus exactement à l'aide de la clé publique du destinataire). Le destinataire est en mesure de déchiffrer la clé de session à l'aide de sa clé privée. Ainsi, expéditeur et destinataires sont en possession d'une clé commune dont ils sont seuls connaisseurs. Il leur est alors possible de s'envoyer des documents chiffrés à l'aide d'un algorithme de chiffrement symétrique.

3. Conclusion

Dans ce chapitre nous avons présenté un nouvel algorithme de chiffrement en utilisant les algorithmes évolutionnistes, où nous avons présenté en détailles les différentes étapes du processus crypto-évolutionniste et nous avons justifié nos choix en termes d'opérateurs génétiques et leurs taux d'application. Cet algorithme a été testé sur différents messages et les résultats obtenus seront présentés dans le suivant chapitre.

Chapitre IV

Expérimentation et Résultats

1. Présentation

Nous avons développé un nouvel algorithme de chiffrement évolutionniste, qu'on a appelé ACEO pour Algorithme de Chiffrement Evolutionniste basé Occurrences, qui reçoit en entrée un message qui peut être formé uniquement de nombres ou un mélange de nombres et de phrases littéraires, arabes ou française, y compris la ponctuation, etc.

Dans cette partie, nous présentons les résultats de chiffrement obtenus pour différents messages. Et pour mettre en valeur cette nouvelle contribution, l'algorithme développé sera comparé avec les plus connus des algorithmes de chiffrement.

2. Structure du programme

Nous avons utilisé le langage MATLAB pour implémenter notre algorithme de chiffrement. Le code source présente environ 540 lignes réparties entre des fonctions, du fait que nous avons conçu le programme de façon modulaire où chaque fonction se charge d'accomplir une étape du processus crypto-évolutionnaire. Les principales fonctions sont les suivantes :

- La fonction **ChInit** : Permet de construire le chromosome initial à partir du message en clair.
- La fonction **Initialisation** : Génère la population initiale à partir du chromosome initial.
- La fonction **CroisOX** : Réalise un croisement de type OX entre deux individus.
- La fonction **Permutation** : Effectue la mutation sur un individu.
- La fonction **Evaluation** : Evalue une population donnée.
- La fonction **SelProportionnelle** : Détermine les individus retenus pour construire la génération suivante en utilisant la méthode de sélection proportionnelle.

- La fonction **CrytpEvol** : Se charge de récupérer les résultats des différentes étapes du processus évolutionnaire qui serviront à l'analyse de la convergence de l'algorithme. Elle se charge, aussi, de générer la clé de chiffrement.
- La fonction **ChFin** : Construit le message chiffré à partir du chromosome solution.

Ces fonctions implémentent le schéma général d'un algorithme évolutionnaire, [74]**,** donné ci-dessous :

BEGIN
INITIALISE population ;
EVALUATE each candidate ;
REPEAT UNTIL (termination condition is satisfied) **DO**
1. *SELECT* parents ;
2. *RECOMBINE* the pairs of parents ;
3. *MUTATE* the resulting offspring ;
4. *EVALUATE* new candidates ;
5. *SELECT* individuals for the next generation ;

OD
END

Figure IV.1. *Schéma général d'un algorithme évolutionnaire en pseudo-code*

3. Choix du taux de croisement et de mutation

Pour pouvoir choisir les bons paramètres relatifs aux taux de croisement et de mutation, il a fallu appliquer l'algorithme plusieurs fois. Les différentes valeurs de test appartiennent aux intervalles [0.6,1] et [0.001,0.05] pour la probabilité de croisement et celle de mutation respectivement (voir tableau IV.1). Ainsi, la probabilité de croisement a été choisie de 0.8 (taux = 80%) et celle de mutation correspond à 0.008 (0.8% comme taux).

Le tableau suivant présente les résultats de chiffrement du message exemple mentionné dans le chapitre précédent, en termes de nombres de générations, de valeurs de convergence et de temps d'exécution obtenus pour différentes valeurs de probabilités de croisement et de mutation.

Pc	Pm	NG	VC	T(s)
0.6	0.001	86	1208	40.7
0.65	0.001	72	1166	32.2
0.7	0.001	95	1174	37.7
0.75	0.001	19	1198	37.3
0.8	0.001	79	1158	38.5
0.85	0.001	88	1188	30.4
0.9	0.001	31	1198	114.3
0.95	0.001	72	1202	34.9
1	0.001	66	1190	31.2
0.6	0.005	65	1176	25.4
0.65	0.005	78	1192	34.6
0.7	0.005	61	1168	32.7
0.75	0.005	67	1188	35.5
0.8	0.005	42	1210	46.8
0.85	0.005	93	1202	28.2
0.9	0.005	51	1178	30.8
0.95	0.005	46	1172	32.3
1	0.005	87	1188	33.2
0.6	0.007	89	1134	35.4
0.65	0.007	93	1146	39.8
0.7	0.007	60	1192	28.7
0.75	0.007	93	1202	33.1
0.8	0.007	86	1200	33.1
0.85	0.007	61	1190	30.8
0.9	0.007	65	1184	30.5
0.95	0.007	41	1174	28.3
1	0.007	86	1194	32.3
0.6	0.008	75	1182	33
0.65	0.008	98	1202	29.5
0.7	0.008	70	1200	47.6
0.75	0.008	62	1206	32.3
0.8	0.008	58	1212	27.7
0.85	0.008	70	1172	35.8
0.9	0.008	77	1198	34.1
0.95	0.008	84	1190	34.5
1	0.008	98	1202	41.5
0.6	0.01	63	1190	36.4
0.65	0.01	65	1180	26.9
0.7	0.01	73	1200	33.1
0.75	0.01	32	1184	45.8
0.8	0.01	42	1194	31.7
0.85	0.01	94	1184	31.2
0.9	0.01	67	1158	30.7
0.95	0.01	32	1190	62.9
1	0.01	50	1184	35.7
0.6	0.03	98	1182	41.5
0.65	0.03	80	1168	25
0.7	0.03	76	1186	26.1
0.75	0.03	41	1180	28

0.8	0.03	69	1204	40.6
0.85	0.03	30	1182	42
0.9	0.03	95	1148	50.2
0.95	0.03	96	1198	39.9
1	0.03	87	1196	45.6
0.6	0.05	80	1186	25.4
0.65	0.05	49	1188	25.7
0.7	0.05	95	1196	37.2
0.75	0.05	91	1200	36.6
0.8	0.05	88	1212	91.4
0.85	0.05	59	1168	35.1
0.9	0.05	42	1196	41.6
0.95	0.05	96	1182	30
1	0.05	38	1192	54.3

Pc : probabilité de croisement
Pm : probabilité de mutation
NG : nombre de génération
VC : valeur de convergence
T : temps d'exécution

Tableau IV.1. *Résultats obtenus suivant les différentes valeurs de probabilité de croisement et de mutation choisies*

4. Résultats

Notre algorithme (ACEO) a été testé sur des messages de différentes tailles, représentant soit des passages écrits en langue française, arabes ou formés de nombres seulement dont nous présentons, dans cette section, les résultats obtenus. Et pour pouvoir choisir le bon paramètre relatif à la taille de la population, il a fallu appliquer l'algorithme sur des populations ayant différentes tailles pour pouvoir retenir le meilleur.

4.1. Message 1 (278 caractères)

يلعب المنتخب الوطني لكرة القدم هذا المساء مباراة ودية دولية أمام المنتخب الأرجنتيني بملعب نوكامب الشهير بمدينة برشلونة الإسبانية . و تعد المباراة الأولى من نوعها في تاريخ المنتخبين منذ مونديال الأواسط باليابان عام 1979. و ستكون العودة إلى مرسيليا مباشرة بعد ذلك لمواصلة التربص.

Figure IV.2. *Le message 1*

؟؟@ì***؟؟*d[c[@*¤?¤î=*ìd[[*[*[*▯ [d¤*˝@£˝؟u¤£é]*@**]]~˝*@@˝*~؟¤ìì[{`àì؟@@½*@؟؟uì¤˝؟*ء¤¤u
؟½ì**˝8ق؟u؟]]ق~==؟@k*˝؟+؟*u؟@؟ì*ìؤ=£*¬~]]ق[ق=]¤*؟k=[=[؟**ء@فق@d*@[و¤؟~خ@]؟+d~ì=~]@@ì=*e*
¨d£؟خ¨¨¨*وàì[*é[*▯ ~`ì½£8**£ì؟؟¤*=`k*~؟*[¤à8u؟[*àu*˝¨*¤*؟dîu˝[¤d@¤˝*ق**¤u˝؟~؟=]؟@¤؟à=؟=éì~خ*£=£`

Figure IV.3. *La forme cryptée du message 1*

Clé de session :

18 61 2 37 15 111 25 24 50 4 5 6 7 8 9 10 11 12 13 14 16 17
19 20 22 23 27 28 29 31 32 33 34 35 36 38 39 71 41 42 43 92 44 45
47 48 49 51 96 52 53 54 55 56 46 57 59 60 62 64 65 66 67 68 40 69
72 73 74 75 76 77 78 79 80 81 58 82 83 84 85 86 87 88 89 90 91 93
94 95 97 98 99 100 101 115 103 104 105 118 106 107 113 108 120 3 109 110
112 114 116 117 119 121 122 123 124 125 126 127 128 129 130 131 132 133 134
135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 63 1 102 26
30 70 21

Le tableau suivant représente un récapitulatif des résultats de chiffrement du message 1 obtenus avec notre algorithme de chiffrement pour des tailles de population différentes.

TaillPop	NG	VC	T (s)
3	94	516	3.5
6	61	552	16
8	69	536	17.6
10	61	544	15.7
13	60	544	19.2
15	55	546	21.1
18	82	548	36.6
20	94	554	46.3
22	46	554	33
25	69	552	57.8
28	55	550	64
30	52	544	93.5
33	59	552	53.7
35	85	552	55
38	74	550	58.7
40	86	552	61.8
43	83	544	70.7
45	97	548	70.4
48	64	550	86.4
50	99	548	73.2

TaillPop : taille de la population
NG : nombre de générations au bout duquel la solution est atteinte
VC : valeur de convergence

Tableau IV.2. *Résultats du chiffrement du message 1*

D'après ce tableau, nous constatons que la valeur de convergence maximale (554) a été obtenue pour des populations de tailles 20 et 22 après 46.3 secondes et 33 secondes respectivement, sachons que la valeur de convergence optimale vaut 556 (correspond à la taille du message*2 = 278*2). Pour cela, nous optons pour une population de taille 22

puisqu'elle assure une convergence plus rapide que celle assurée par la population formée de 20 individus. De plus, et si nous pensons à l'opération de cryptanalyse, le fait de deviner et de suivre le changement d'états de 22 individus est plus compliqué que de suivre le changement d'états de 20 individus. Autrement dit, le nombre des différentes combinaisons possibles avec 22 individus, pour des positions de croisement aléatoires, est considérablement plus grand que le nombre de combinaisons possibles avec 20 individus ; ce qui revient à dire que la cryptanalyse dans le premier cas (taille de population = 22) est plus difficile.

Il est important de noter, aussi, que les populations de tailles 6, 25, 33, 35 et 40 ont donné de bonnes valeurs de convergence (552), mais le problème qui se pose avec la première (population de taille 6) est l'insuffisance du nombre d'individus formant cette population ce qui facilite l'opération de cryptanalyse. Les autres cas (populations de 25, 33, 35 et 40 individus) sont aussi à éloigner puisque leurs temps d'exécution correspondants (57.8 s, 53.7s, 55 s, 61.8 s respectivement) sont plus grands que le temps d'exécution correspondant à la population choisie (de 22 individus).

4.2. Message 2 (816 caractères)

أعلن الجيش الأمريكي يوم أمس الاثنين 26 مارس 2007 عن مقتل خمسة من جنوده، أربعة منهم قضوا في حادث وحد، كما جرح في ذات العملية جنديان آخران. و في بيان آخر أعلنت قيادة قوات الاحتلال إصابة أربعة جنود آخرين في عمليتين عسكريتين في محافظتي ديالي و بغداد. و جاء في بيان آخر أن عبوة ناسفة انفجرت أثناء مرور دورية أمريكية في بغداد، و خلفت مقتل جندي أمريكي و جرح اثنين آخرين. و بمقتل هؤلاء الجنود الخمسة ترتفع خسائر القوات الأمريكية في العراق إلى 3240 قتيل، و هذا منذ احتلال العراق في مارس من عام ألفين و ثلاثة حتى يوم أمس، و هذا حسب البيانات الرسمية الصادرة عن قوات الاحتلال. بموازاة هذا أصيب أربعة أشخاص بجراح بينهم شرطي في انفجار عبوة ناسفة بحي الزعفرانية جنوبي شرق بغداد ، و في البصرة حيث تتواجد القوات البريطانية، أصيب جندي بريطاني في انفجار عبوة ناسفة أثناء مرور دورية بريطانية وسط المدينة، مما تسبب في إعطاب إحدى آلياتها.

Figure IV.4. *Le message 2*

o¶è*ˆى-
èèz¶pzبèùzzzè"€ظى<è€ف€èzèù®ى¶؛ؤz¶¬ùèzpèêظ!ˆ؛؛ؤ#èyى®zz¶<ozpzى¶ىê€eôyyèô9¶فzzىe#op÷pùz
³>ؤzp<ى<z®÷prؤèzى€>؛o€zèpˆzظ؛ù®€*ىyhىyù³9:ى:hzpèى€y¶:p'è€*è¶®&oèzèùè¶o<¶ظzyê¶ؤoزىùyىëù#
ىزىˆزyzèzzىzzz:èyùè®*zz9®oyzëcyèىزz÷1è€yz*ف®¶z*oىèz؛hzèظزy:zyzèù€z#€z:÷èzz¶,ˆى¶h:èpؤ¶بpˆؤyè
y&ocz:ozzyzèؤoèė zˆp|ؼzبz9ؤoèè9ىzzpyz؛zyىٹo-#ˆ؛&®oèù®=èèyz-zz|÷:z9®¶ˆz'èù¶o,9y|rozzo¶z-
"y|zzˆ ىغzyzüh÷€zى¶¶ؼp¶<#<yˆ|€ؤ¶ô®èˆىˆؤyz؛zzè®<ùz¶hèzبzˆىzؼ<€è®ؤpˆcèè÷zy¶ùى<èى<èëىz9ىe¶z9zىo
ôpèc€ى¬¶zzˆىè÷zeèèzz:ؤzىpozzèzىzzzù<èىèى*ˆىù9ؤzzىy**ze®÷<èz¶zoo³€ˆê<z®€¶zظzؤ÷èzy9ˆ®<c¬
ىى¶ؤyyè:ؤy<ز€zؤp":èèoyèظèè®zyˆ9ىp|®|:ؤ÷çzp®z÷opz÷ˆo<ˆèz¶ypèˆèp<ىٹypz¶ى*®#ùz|ˆظ÷èozoظzى<zz*zp
-ظyz¶ظcèè€9ظ:zoy¶ظىpz÷ؤ؛y¶®zèô®ˆبùèyë-
èèzzz<®ˆzzzzp÷z÷÷èىزىzؤغ€ؤoˆèzzzىzz*p€èzˆ9ù³ؤ¬بzىz<zèىزôz|è€9è¶ˆ<*ôèˆèp¶è|y<€zؤzèêôzظè¶<èى؛op
ى::ˆؤy9r*yىz؛èz9غىèzˆزùz¶zzo#nغىo9p®®z÷yظp¶ؤ<è

Figure IV.5. *La forme cryptée du message 2*

Clé de session :

45 61 66 34 71 55 96 98 77 91 94 127 128 58 20 80 1 2 3 5 6 7 8 33 9 10 48 11 12 13 14 15 16 17 18 19 21 22 23 24 26 27 28 29 72 93 31 32 35 37 38 39 41 42 43 46 47 49 50 51 52 79 53 54 56 57 59 60 62 44 63 64 65 68 69 70 73 74 75 103 76 78 81 82 84 102 129 85 105 86 87 108 109 89 90 92 83 114 115 95 97 99 100 67 101 104 106 107 88 110 111 113 112 116 117 120 118 119 121 122 123 126 124 125 130 131 132 133 134 135 136 139 137 138 140 141 142 143 144 145 146 147 148 149 4 25 30 36 40

Le tableau ci-dessous représente un récapitulatif des résultats de chiffrement du message 2 obtenus avec notre algorithme de chiffrement pour des tailles de population différentes.

TaillPop	NG	VC	T (s)
3	99	1522	4.2
6	69	1512	13
8	37	1536	11.4
10	60	1556	15.7
13	98	1552	17.5
15	66	1532	19.8
18	96	1566	18.3
20	99	1602	30.3
22	69	1614	29.6
25	98	1560	32.6
28	97	1602	41.7
30	87	1574	51.6
33	97	1588	52.5
35	71	1592	50.6
38	22	1604	57.9
40	94	1584	58.6
43	57	1586	119.2
45	91	1584	90.7
48	83	1582	73
50	79	1588	90.1

TaillPop : taille de la population
NG : nombre de générations au bout duquel la solution est atteinte
VC : valeur de convergence

Tableau IV.3. *Résultats du chiffrement du message 2*

D'après les résultats présentés dans le tableau ci-dessus, nous constatons que la meilleure valeur de convergence (1614) a été obtenue pour une population de taille 22 après 69 générations, avec un temps d'exécution égale à 29.6 secondes.

D’autre part, les populations de tailles 20, 28 et 38 ont donné des résultats assez satisfaisants du fait que leurs valeurs de convergence correspondantes (1602, 1602, 1604) sont assez petites de la valeur de convergence optimale qui vaut 1632 (taille du message * 2), ce qui signifie que, dans ces cas, la confusion apportée par l’algorithme en terme de brouillage du texte initial est assez satisfaisante et assez significative. En plus de ça, les temps d’exécution correspondants à ces instances du problème (30.3 s, 41.7 s, 57.9 s) sont plus grands que celui correspondant à la population de 22 individus.

4.3. Message 3 (803 caractères)

By encryption, we mean a process of converting information to a disguised form in order to send it across a potentially unsafe channel. The reverse process is called decryption. Using strong encryption techniques, sensitive, valuable information can be protected against organized criminals, malicious hackers, or spies from a foreign military power, for example. Indeed, cryptography used to be almost exclusively a tool for the military. However, in moving into an information society, the value of cryptography in everyday life in such areas is privacy, trust, electronic payments, and access control has become evident. In this way, the field of cryptography has broadened from classical encryption techniques into areas such as authentification, data integrity, and non-repudiation of data transfer.

Figure IV.6. *Le message 3*

§#06++؟¨à{+1£+0´1à1‹à+1آ¨à+#é0¨£1#§£î¨îغàî+6+¨1و{à0é6¨à+ض¨éغ+‹ààéî1+à09ْ+#¨غ#é0++q1ْë¨00c¨î{
آî8à‹9¨+01{î1£#aْ0î1à+0î£01¨´+àî^ëîîî+ààîéë¨+#¨1´éà0ààà+§8¨£§¨¨0´£è#§é1ë+++£11¨{é1؟£ëà¨++§+à+¨++9
§´£09£و+ë؟é#éà06غà1+1#18و£+++ééà#ë£¨+08££+¨00‹؟و+9}1¨ْ+8؟à+11++§#‹î0؟¨غ‹1ء´î#é0#§£éè+1+غ8و£
+ëà9{¨éو#¨و£à¨é+++00éْ#î1+î#à‹0#0#éàà+0‹î1‹+80#{é0ْ¨é++#àé11+؟‹é#0#+غ££++à1غéخ£¨¨¨à#8}ûé+¨î90+و+
ë¨î‹§1++ëé؟+î£àé#ë11§#à#î£+1£é+1ëc¨خ+ؤîéë#î؟#؟î1#ً¨+8#+1+و£éûé¨غ8؟1£و#é10èà#+08¨1#+£é0é£+88
+#9‹^آî¨î+¨{ْ£§£+´6§6+¨à{6+##ûë؟+îà+16î1¨11£1£0#9+غà§+é£+ë+#11؟£ëî‹î+£‹1+´£00£+غ¨+ûé#£19§î+c01
àî¨8#8و+à§¨{î¨+60¨§608é#1#à{+1£‹à9#1؟à6+¨1£ْ6à+1î1ë9£++؟îî+80£6é#¨+£¨1¨{ëc0و{+00#ûè{è8î#£¨£#ûàà
£#+1£+¨+à؟à+++++àْ1161؟+éîà£c{´+#c´¨8´{éîو¨é110£#غخ

Figure IV.7. *La forme cryptée du message 3*

Clé de session :

2 26 29 40 42 51 99 58 64 66 8 74 76 81 3 4 5 6 7 9 10 11 12
13 14 15 16 17 18 19 20 21 22 23 24 25 27 28 30 31 33 34 35 73 36
37 38 39 41 43 44 45 46 47 48 50 53 54 55 56 57 59 61 62 63 65 60
67 68 49 69 70 71 72 75 77 78 79 80 52 82 83 84 85 86 87 89 88 90
92 91 93 94 95 96 97 32 98 100 102 101 103 104 105 106 107 108 109 110
111 112 113 114 115 117 118 119 116 121 120 122 123 124 125 126 128 127 129
130 131 132 134 133 135 137 136 138 139 140 141 142 143 144 145 146 147 148
149 1

Le tableau ci-dessous représente un récapitulatif des résultats de chiffrement du message 3 obtenus avec notre algorithme de chiffrement pour des tailles de population différentes.

TaillPop	NG	VC	T (s)
3	85	1540	3.9
6	61	1550	10.6
8	48	1504	14.6
10	86	1516	13.8
13	67	1564	18.9
15	93	1594	22.3
18	98	1566	28.3
20	85	1602	30.5
22	50	1606	30
25	75	1604	43.5
28	36	1604	103.5
30	99	1604	63.6
33	93	1606	101.7
35	88	1602	52.4
38	75	1568	55.8
40	92	1602	65.7
43	97	1596	68
45	73	1600	74.2
48	58	1586	78.5
50	94	1596	106.4

TaillPop : taille de la population
NG : nombre de générations au bout duquel la solution est atteinte
VC : valeur de convergence

Tableau IV.4. *Résultats obtenus du chiffrement du message 3*

D'après ce tableau, la meilleure valeur de convergence (1606) a été obtenue pour des populations de tailles 22 et 33 et elle représente en même temps la valeur de convergence optimale. Toutefois, la population de 22 individus a occupé 30 secondes comme temps d'exécution, tandis que la population de 33 individus a passé 101.7 secondes en exécution ce qui est peu important.

D'autres instances du problème correspondantes aux populations de 20, 25, 28, 30, 35, 40 et 45 individus ont aussi donné de bonnes valeurs de convergence (1602, 1064, 1604, 1604, 1602, 1602, 1600). Cependant, les temps d'exécution de certaines de ces instances (43.5 s, 103.5 s, 63.6 s, 52.4 s, 65.7 s, 74.2 s) sont plus grands que le temps d'exécution de la population de 22 individus, et malgré que le temps d'exécution de la population de 20 individus (30.5 s) est presque égale à celui de la population de 22 individus (30 s), cette

instance (population de 20 individus) est moins résistible aux tentatives de cryptanalyse que l'autre instance (population de 22 individus) puisqu'elle englobe moins d'individus ; donc, il sera plus facile de déterminer les combinaisons d'états menant à la solution.

4.4. Message 4 (1149 caractères)

Abraham Lincoln, the 16th president of the United States, guided his country through the most devastating experience in its national history--the CIVIL WAR. He is considered by many historians to have been the greatest American president. Abraham Lincoln was born Sunday, February 12, 1809, in a log cabin near Hodgenville, Kentucky. He was the son of Thomas and Nancy Hanks Lincoln. Thomas Lincoln was a Abraham had gone to school briefly in Kentucky and did so again in Indiana. He attended school with his older sister...Both of Abraham's parents were members of a Baptist congregation which had separated from another church due to opposition to slavery. When Abraham was 7, the family moved to southern Indiana. As Abraham grew up, he loved to read and preferred learning to working in the fields. This led to a difficult relationship with his father who was just the opposite. Abraham was constantly borrowing books from the neighbors. Lincoln's declining interest in politics was renewed by the passage of the Kansas-Nebraska Act in 1854. In 1860 he furthered his national reputation with a successful speech at the cooper Institute in New York.

Figure IV.8. *Le message 4*

à2أ¢à¦¦200آ^â22آ÷q¢ ز÷:â¦2{آqââ÷÷22¢222¦ز}¢j^jë زj2q¢&±¦22¦(}÷}j¢à`j0 زââ&j¢q÷ëjj+÷(2آ÷أj(&g2}¦` ز÷j^«2¦
2j(ë÷÷jv`2&`~2÷ث±آj÷آkâ&¦¦ زj÷2_&2÷222^){&±آ¢^آj&¦¢ زq÷`!ë}¦(أ÷¦ز ز÷÷qj2&#j2ë2â}÷2#2â:}jj2(&2آj2q¢±
22ë(أàj}:2¢¢jjأj¢2÷2` زq&0& ز÷2 ز&j÷2âآ}&}¢}!jآ زâq¦¦#`2آj&2¢â¦±¢jq2}ثq¢`¦ëq±ث&÷j¢2!2آq÷}jj}¦¢¦&¦2jqثj
¦&±j2±÷÷}¦÷2q2&¦j¦5 ز2{)&آ^&qq¢jjqâ}¦¢22jث÷â÷^â2!¢¦2¢2¢±¢¢{ ز:2÷¢ زâ+_¦&2{÷`& زjj(&â2&22أ}÷÷^±q2
}02&¦ث¢2¦&{¢ث±jâ±2à}ëh&(&}q¦âj¦22¢&â2÷(q÷÷2}¦â0q¦2&â&(÷àj¦(¦ëâ_2 زjj&22&22ث&àj(¦±+!}÷ زتز زأ2
q^2â زqثj0 زââ2!2:¢ث22÷ ز22¦÷2&&÷ زâ÷}÷!*÷}&&} زë:2÷2!¢}2±2 ز2{2&2*!ث:{ث±&â^jë¦*&j±âqëâ زq زj0}
¢j÷&g¢qآ¦}qj2÷âj22 زg0¢¢¦â?2÷ث÷أ&j+2¢^±ث(`j^ ز(àثq:&2±¢` زتq22آq22:2}±}2 ز`22#آ¦&j¦22: زq2¢¦}2& زj&
2(¢& ز(j2ث÷ف¦â¦â&}ث ز÷¢}ëq`آ}j2¦أ` زj2j÷¢±÷0 ز¦!¦¢¢â2 زتآ}ë÷&2÷&¢22 ز± زh&&÷2¦q&¦22¢2!{¢)¢&2÷ ز±±}^
!âj«!â¦à2&&2{آ&!2 زj2j±¦¦j2à÷:22 ز±qj&q2±آj}j÷÷+i#j¦(q¦آq زà}q!±÷â2q&â0¦âq`0& زتë2qشàq2^&2آâ^q}&â¦
أأآ ز^j¦â222â}÷&}&22÷ ز¢ ز2 ز*: زjq¢}j¢2 ز&j ز`jj}2¢¢2 زâ&أ+q¢2¢ث:âj زثأ2{+آ`ââ÷#2¦`} ز÷â&j:`jà÷2qj÷÷j`â÷}÷
*jq02÷2 زà*2¢آ÷ ز±÷q÷¢2«2±آ&j2¦2â÷22¢j2 ز&22±2 زq ز¦¢ëآ÷ زâq22à زj¦¦0ث÷jâآ}÷«qâ2}22&2âq*¦¢â¦¦â÷¦22¦*2آ
±¦à÷^¦222ë¦jâ&2*02#«22j#¢âq÷!2ث2&آ&{{* زjآ÷¦àà:2&52أ2 زت÷2أ2ش{!02¦2 ز}j÷¦0 ز}j&}}j زâ زآأ¦¢¦÷ث}q¢2ث÷±
jâ!÷q2j}¦÷^ثز¢â22¦ز÷

Figure IV.9. *La forme cryptée du message 4*

Clé de session :

33 36 47 145 71 72 12 21 60 37 19 1 3 4 5 6 7 9 10 11 13 14
43 15 16 18 75 20 22 23 31 24 25 26 27 29 30 32 68 34 35 38 45 39
40 41 42 44 46 48 49 50 51 52 53 54 73 55 56 57 58 59 79 61 62 63
64 65 66 102 87 67 69 90 70 92 74 76 77 96 78 80 99 81 101 82 83
84 85 86 107 88 109 89 91 93 94 95 97 98 100 103 104 105 106 108 111
110 112 114 113 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129
130 131 132 134 135 133 137 138 136 139 141 142 140 143 146 147 144 148 149
2 8 17 28

Le tableau suivant représente un récapitulatif des résultats de chiffrement du message 4 obtenus avec notre algorithme de chiffrement pour des tailles de population différentes.

TaillPop	NG	VC	T (s)
3	65	2094	4.7
6	96	2250	9.9
8	25	2276	11.2
10	70	2260	18.9
13	78	2294	38.9
15	81	2266	23.2
18	52	2262	26.9
20	68	2292	30.8
22	46	2294	28.7
25	71	2288	44.5
28	26	2296	49.4
30	84	2282	54.3
33	92	2278	48.7
35	80	2276	50.5
38	41	2292	68.2
40	69	2286	61.1
43	76	2272	65.4
45	78	2280	71.6
48	95	2272	76.3
50	76	2268	81.9

TaillPop : taille de la population
NG : nombre de générations au bout duquel la solution est atteinte
VC : valeur de convergence

Tableau IV.5. *Résultats du chiffrement du message 4*

Selon le récapitulatif des résultats présenté dans le tableau ci-dessus, la meilleure valeur de convergence (2296) est celle obtenue par une population de 28 individus après 49.4 secondes d'exécution.

La deuxième bonne valeur de convergence égale à 2294 obtenue par des populations de 13 et de 22 individus avec des temps d'exécution de 38.9 s et 28.7 s respectivement. Donc, il est clair que la deuxième instance des deux dernières instances (population de 22 individus) est préférable que l'autre, puisqu'elle consomme moins de temps d'exécution et elle englobe plus d'individus lui assurant une bonne résistibilité face aux attaques possibles. De même, et malgré que cette instance présente une légère dégradation en valeur de convergence par rapport à la meilleure des valeurs de convergence, mais le temps de calcul est réduit presque à la moitié, ce qui favorise d'opter pour elle (taille de population égale à 22).

Nous constatons aussi que les valeurs de convergence obtenues par les deux populations de tailles 20 et 38 (2292) sont aussi bonnes, mais la première présente une légère augmentation

en temps de calcul par rapport au cas choisi (population de 22 individus) plus une diminution de la taille de deux individus. L'autre instance (population de 38 individus) souffre d'une considérable augmentation en temps de calcul presque de trois fois.

4.5. Message 5 (1026 caractères)

Traveling salesman problem (TSP) is an NP-complete problem, implying that there is no know algorithm to solve it exactly in better than O(an) time (for n cities). In fact, there are (n-1)!/2 possible combinations, so to try every one requires factorial time with the problem size. Just to remind, there are cities and given distances between them. Traveling salesman has to visit all of them, but he does not want to travel very much. The task is to find a sequence of cities to minimize traveled distance. In other words, find a minimal Hamiltonian tour in a complete graph of n nodes. A method was implemented for a 6 city problem. The optimal tour found for the 6 cities was: 1-2-5-4-6-3-1 which has distance 9.242640 units. The processor times for the 5 and 6 city runs were below the timer resolution. The time for each combination can be estimated as 2s-07 seconds. The number of combinations for the 29 cities problem is 8.84s30. This gives a time estimate of 1.77e24 seconds, or 5.6e16 years.

Figure IV.10. *Le message 5*

?3 ²ي2=+¢ صاپر¾ي?¢د¾˚²³¼¦د
ي¾€ ييؤ¾/د¾¾¾¾3zدgص¾=¾©üي¾¾ ؤ²ي/²³¼ي²ي~ؤؤ?يي* ؤ3 دؤ*¾ي¾?دي¾¾ ؤâيا¾?د?ي?üي ؤü¢
²³¼¾¾¢ي² ؤيدبيد+ي²³¼يي²² ژdé¢صgي~+ي¾*¾ د/ي ژ?ي¾، ؤؤ¸gت*²ي¹ي©=²3²?ي+دي3¦3|*ي¾يü ؤدü¹¾ت¾يüي¸?ي?*د¾ ؤ
¢*¢g¢د¾ü²³¼يدgü¦ ررgد+¾2€د¾©¾ رü¨3*ي*+*ي?ي€?3 ژé ؤييب+يي¾/+¢ي²ييي¾ü¾ت¢¦? ؤ2 ؤيدي²¹د¾¦ا¸üü3¦3دييü¢²دؤéïط
¾+©g²*² ؤ¾¾ص¦ü²²¹ü*3?_²¢ ؤيبت:ي¸ي¾صü*¾+?â+ رديدي? ؤر*دي²دؤü¢¾يé¾² ژ+¦3ي?ي¢ي¾/ي¾3+€دü3â²²ي¾®?+=*¦ ؤ*ï
+¾3¾*+¾üي¨دي~¾gييي¢?يg€ي? ر²ي²=_¢دüيي¾ي¦ؤ ددي¸¢+ ؤ¦ي ژé+ ؤ*üييي²ي©/+¾¨د¦*ü/²¦²²¢ ؤ€ي3+ ؤé3صي¢€ ؤ*«ي3 ؤg
?_²gي~د+ص©*©دي²³¼يâ_دي+ ؤé¾¦¾3دييي¾ ؤ¸*+ ؤدي*¦ ؤيي+ييي¦ ؤ+د3+ي¾ي*يي3 ؤدييي¦ي3© ؤ*دد ؤ+?®*ت ؤgيديdي¦¢â¸ ؤ?¾ü
3¾*ييي©يي3ي*3¸رد*ت¢ü= ؤص?¢+©ي+/üصدد*¾*ü3
د²³¼دديي¾ط¢¾¾¨ي¦3ي¦¾3!_يزي)%g²¢ي¾دد*د+ ؤدزيي²ييg+¾3² ري¢¢ش¦ü*â3د² ؤ¾ü²?*
ي€ي++²³¼+3_©+ص¾*ي ؤ=يد¢23©ت*z²¦ ؤ€يü¾¾²3يüدüâ=يييي¸ي،*¦?3¾ي²³دي¾.صي33©€¸=ي€ي¦دييي ؤéص¾يي¦د+¾ي¾د
üي+د?g²ي3ص/üي ؤ*ديgر ؤ!¢دي ؤ¾32² ؤ¾¾دي¾دي ؤيg¹¢* ؤ€¦¸¦ ؤ:ü3**€رد¾د¾د33د+*يدgي/g32®¾*¸*3ي_يي¾g33*
¢¢د²ي¦ديي?ي*3ي ؤ3?يي¾ggييدي*¦? ؤيؤ*+¾3رü+ي¢=د2 ؤ©+ييي€*ي=+¢+د© ؤييé¸+*²يü3+ طد
طد©¢¢ ؤي ؤ3ص¾+¦د¸يصي3ي ؤييي¦*üégيرü ؤييü*ييي صü²ت+ ؤي¾¢¾د*¾+*gي* ژüص¦¦¸*+ر¾ü3¸ي€¾ü¾~²+دgي¾
3 ؤ²ي3+=¾ي*يg¦ ؤ¦²د

Figure IV.11. *La forme cryptée du message 5*

Clé de session :

39 22 16 59 1 2 6 64 7 8 9 10 11 12 13 14 15 17 19 20 21 23 24 25 26 27 47 28 29 30 32 33 34 31 58 35 36 37 40 41 66 42 5 43 44 45 46 50 51 18 76 75 78 52 81 3 53 54 55 57 60 89 90 61 62 63 65 67 97 69 70 98 71 72 101 73 74 79 82 83 84 86 87 110 88 112 113 114 91 92 93 94 95 96 99 100 123 102 125 103 104 128 129 105 106 107 108 109 38 111 115 116 117 118 49 119 80 85 120 121 136 4 122 131 124 126 127 130 132 133 134 135 137 138 139 140 142 141 144 145 143 146 148 149 48 56 68 77 147

4.6. Message 6 (684 caractères)

Astronauts In August 1983 Guion Bluford became the first African American to go into space, while serving on a mission aboard the Challenger space shuttle. Bluford said that the blastoff of the shuttle was like riding in a high-speed elevator through a bonfire. He also recognized that, "From a black perspective, my flight on the shuttle represented another step forward." Astronaut Mae Jemison became the first African American woman to travel in space when she flew on the space shuttle Endeavor in a September 1992 mission. After her space flight, Jemison resigned from NASA and established the Jemison Group, a company that researches, develops, and markets advanced technologies.

Figure IV.12. *Le message 6*

)6حج76^ح¥ظ(;ح?»ح^(ح²رحج2??»^ظ؟î}2ح^û¥jظ/¥/))j«ح¥ل7حjح(ح/´î²²²?66(²²6²ح¦²ظ(?ح¥î~7}^²))6حj^77/¥6
حححج÷¦²ح¥7ح~¥¥¬6¥¦j¥7)?¥¦)?j6j;ح¥حج;7¥((²^~¥»»¥حj/^حj}7¥(6¥6ح(لj²7ح¢6^??ح}«j¢î)¥¬6î¥;6ح?¥ل¬(ح)j¥2¬}
26¬فج»¥¦q)«¥ح)/î;66ل7!&¥~ح7¥6حîح¥j?jح^//¥ح»»^?»؟ح&¢¥¬¥«j¥²ح²حj77/j)6¥²6¬(&ر¥?حîح«»ح»¥¥حر?j²²
66~¥»رq¬^62ظححط¥ح?¬(?7;ح««î^حححج6/(»/¦/?6حج6حj)²?¥6حîر(¥²7¬ح~~»{ح;?¦«î¥«حلج)«î67!!?ر»ح(;6;;^¥/حج?¥ل)j
¥7ل(¥7»¥7ح7/6؟»ح¢6ح^~¥ح)jî7^^¢²لف¥/حj²~6حj6j?¥²jî¢¬?7«««²»زحj)67;¢ح»;²ح?j67²»;^;ح7ح؟;¦²ح6¦²6(ظjf¥7
7/حج6فê¥})^;¥¥î²´ح6ح÷676((»ظ~^q?66ح²?ح؟ح¥ل)j«¥?7¥(2¬¥حظ¦6ح^»^ح^î¢^¥²ظ¥?6¥^ح2¥ظ¬»(ح»¥ظ¥¢¢ل(»¥حî
«^ح(²^»ح¥î²«7ظû/7²¬ح7²?»?²6ظ¦«î¥¥;¥j~²^¥6حبح¥ح¦¦¥îحلjل(زحj6~66¥676²ر»ح^îi6ح6?ح7?6ح~غحج?¥^çj7ح~¥««j
ح¥¥/ل/jj

Figure IV.13. *La forme cryptée du message 6*

Clé de session :

76 28 8 29 1 2 3 6 7 10 11 12 13 14 15 17 18 19 20 21 22 23
24 4 5 9 25 26 27 31 32 69 33 34 35 37 36 38 39 78 40 41 42 43
46 47 48 49 51 54 89 55 56 57 52 58 60 62 63 64 65 66 67 68 70 73
71 72 74 75 77 79 80 81 82 83 84 85 86 87 88 90 91 92 94 104 16
95 96 97 98 99 100 101 102 103 106 116 107 108 109 110 121 122 111 112
113 114 127 115 117 130 118 119 120 105 123 124 125 126 128 129 131 132 133
134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 30 44 50
93 53 59 61 45

5. Récapitulation des résultats

Les deux figures suivantes représentent une récapitulation des résultats de chiffrement des quatre premiers messages présentés précédemment en vue de les comparer et de les interpréter pour pouvoir choisir la valeur adéquate au paramètre relatif à la taille de population.

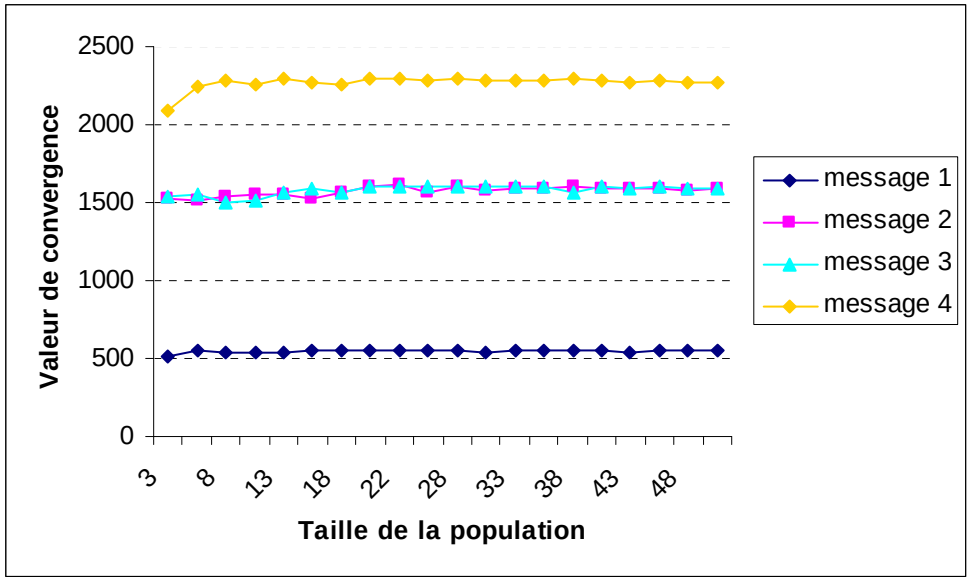


Figure IV.14. *Evolution des valeurs de convergence en fonction de la taille de population*

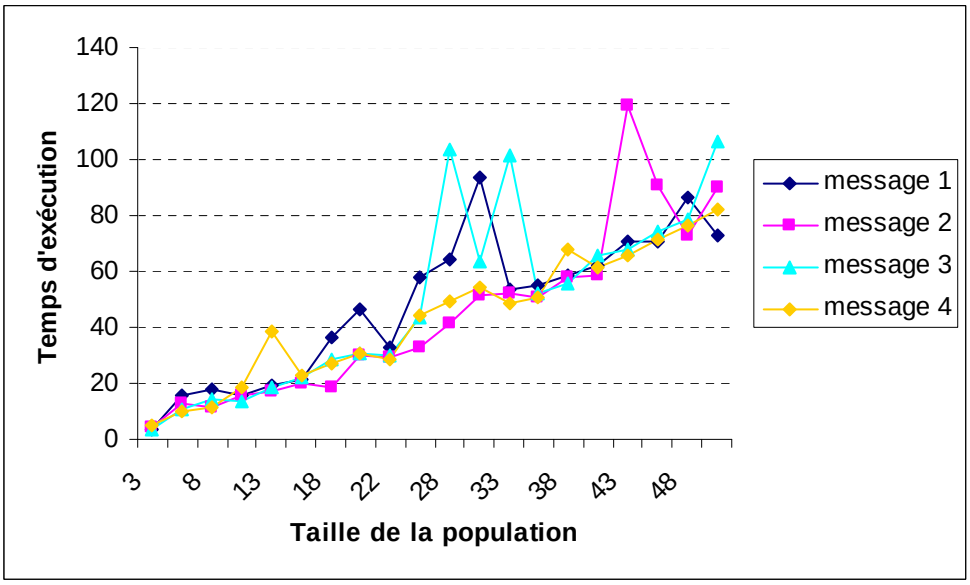


Figure IV.15. *Evolution du temps d'exécution en fonction de la taille de population*

D'après la figure IV.14 présentant l'étude d'influence de la taille de population sur les valeurs de convergence, nous constatons que, pour les quatre messages, les meilleures valeurs de convergence correspondent à une taille de population égale à 22. Ceci se voit clairement sur la courbe correspondante au message 2, tandis qu'il est un peu difficile de le remarquer pour les autres courbes puisque les valeurs de convergence, pour les tailles de populations

voisines de la taille de population qui égale à 22, sont très proches de la valeur de convergence correspondante à cette taille-là (22).

De même, et d'après la figure IV.15 résumant l'étude d'influence de la taille de population sur le temps d'exécution, nous constatons que les temps d'exécution des quatre messages se rapprochent entre eux, pour des tailles de populations égalent à 3, 13, 18, 22, 38, 43. Toutefois, dans les cas de tailles égalent à 3, 13 ou 18, la cryptanalyse est assez facile puisque le nombre d'individus est insuffisant (cas traité et argumenté dans l'interprétation du résultat de chiffrement du message 1). Pour les tailles égalent à 38 et 43, elles présentent l'inconvénient du temps d'exécution qui est assez grand ; tandis que pour la taille de population égale à 22, le nombre d'individus et le temps d'exécution sont satisfaisants par rapport aux autres valeurs.

Donc, et d'après l'interprétation des deux figures, nous optons pour régler le paramètre relatif à la taille de population en lui attribuant la valeur 22.

6. Discussion

Cette section présente une comparaison de l'ACEO avec les plus fameux des algorithmes de chiffrement. Nous commençons par l'étude comparative de l'ACEO avec l'important algorithme de chiffrement évolutionniste OTL (figure IV.16) sur la base des valeurs de convergence obtenues suite à l'application des deux algorithmes sur les messages 3, 4, 5 et 6, pour une même fonction d'évaluation. Les résultats obtenus sont résumés à travers le tableau IV.6 [75].

	ACEO	**OTL**
Message 3	1606	1096
Message 4	2294	1622
Message 5	1986	1394
Message 6	1366	1100

Tableau IV.6. *Valeurs de convergence d'OTL et d'ACEO*

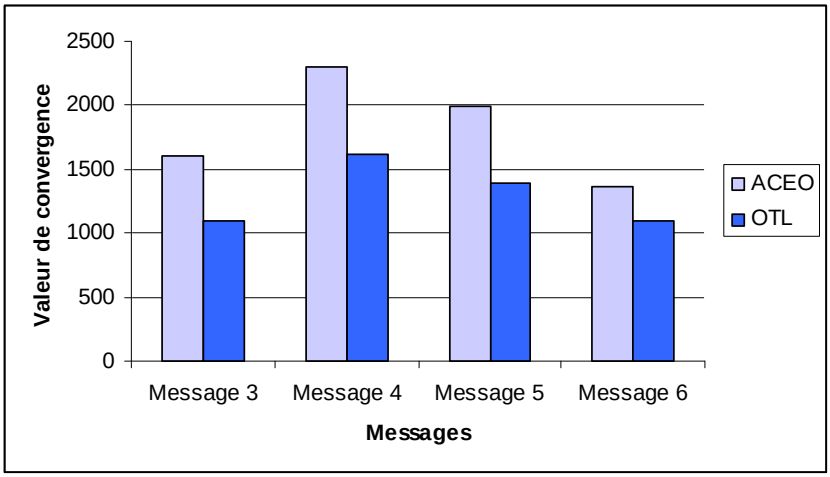


Figure IV.16. *Comparaison de l'ACEO avec OTL*

D'après la figure IV.16 comparative entre ACEO et OTL, nous constatons un considérable écart entre les valeurs de convergence de l'optimum obtenu par ACEO et celles de l'optimum obtenu par OTL de telle manière que les premières sont plus grandes que les deuxièmes, pour une même fonction d'évaluation. Cela revient à la grande puissance du codage d'ACEO grâce à l'introduction dans le message chiffré des caractères ne figurant pas dans le message à chiffrer ; ce qui signifie que la confusion de l'ACEO et plus grande que celle d'OTL. Ainsi, l'ACEO ne nécessite pas une phase de brouillage comme c'est le cas avec OTL.

Et si nous voulons situer ACEO parmi d'autres crypto-systèmes, il sera utile de noter que la majorité des algorithmes s'inscrivant sous le mode de chiffrement symétrique opèrent par blocs (DES, 3DES, AES), ce qui permette à la cryptanalyse différentielle d'être appliquée en parallèle sur les différents blocs. Et comme ACEO opère sur le message entier pour obtenir un message chiffré, ce dernier sera difficilement cassé, voir même hors porté de ce type d'attaque. Même si l'on est confronté à une attaque de type fréquentielle, l'algorithme proposé est fortement sécurisé parce que le texte chiffré contient des caractères qui ne figurent pas dans le texte en clair, en plus du fait que cette attaque étudiée les fréquences d'apparition des caractères d'un texte écrit suivant une langue naturelle, tandis que nos messages peuvent contenir des caractères du code ASCII autres que ceux-ci.

L'autre attaque qui fait peur au cryptologues est l'attaque exhaustive réussissant toujours à remettre en cause des crypto-systèmes utilisant des clés de petites tailles (DES). Ce point est bien pris en considération dans la conception de l'ACEO : premièrement, par l'utilisation d'une clé de session générée à partir du message en clair et du message chiffré (Voir

section 2.2 du précédent chapitre). Donc, elle change d'une instanciation du problème à une autre. Deuxièmement, et vu que la clé utilisée soit la concaténation de 149 nombres représentant des permutations de positions des nombres d'occurrences, donc elle est de taille égale à 1192 bits. Ceci offre à notre algorithme de chiffrement ACEO une résistibilité aux attaques exhaustives. Le tableau ci-dessous donne les tailles de clés des algorithmes DES, 3DES, AES, OTL et ACEO, ainsi que le nombres d'opérations nécessaires pour retrouver toutes les clés possibles. Il est à noter que dans le cas de l'OTL, il est supposé que le message est constitué de 30 caractères différents [76], donc la taille de la clé est de 240 bits. Dans le cas contraire, le message est complété par d'autres caractères dans le but d'atteindre la taille 240 bits pour la clé.

	Taille de clé (bits)	Nombre d'opérations
DES	56	2^{56}
3DES	112	2^{112}
AES	128, 192 ou 256	2^{128}, 2^{192} ou 2^{256}
OTL	$\geq$ 240	$\geq 2^{240}$
ACEO	1192	2^{1192}

Tableau IV.7. *Complexité de l'attaque exhaustive*

D'après les informations comprises dans ce tableau, il est clair que la tâche de l'attaque exhaustive est plus complexe dans le cas d'ACEO par rapport aux autres algorithmes (DES, 3DES, AES et OTL) du moment où le nombres d'opérations nécessaires pour accomplir cette attaque est de 2^{952} fois plus grand que celui d'OTL, de 2^{1064}, 2^{1000} ou de 2^{936} fois plus grand que celui de l'AES, de 2^{1080} fois plus grand que le nombre d'opérations requises dans le cas du 3DES, et de 2^{1136} fois plus grand que celui de DES.

L'unique algorithme admet incassable est le masque jetable puisqu'il utilise une clé aléatoire une fois. Cette dernière doit être aussi longue que le texte clair, ce qui représente l'unique défaut de cet algorithme. L'ACEO préserve l'avantage d'utilisation de clé générée pseudo-aléatoirement pour se bénéficier de sa robustesse, tout en essayant de pallier le défaut du masque jetable vue que, la clé utilisée est moins longue que le texte en clair (sauf en cas où le texte clair est constitué de caractères totalement différents apparu chacun une seule fois,

c'est à dire tous les nombres d'occurrences égalent à 1, dans ce cas la taille de la clé sera soit égale à celle du message ou plus grande).

Il est important de noter, aussi, que l'autre point clé participant à l'augmentation de la confusion de l'ACEO est le caractère aléatoire dominant dans la majorité des étapes de l'algorithme et pendant la génération de la clé de session. Ce caractère aléatoire renforce sa sécurisation et rend toute tentative de cryptanalyse très difficile, voir même impossible.

7. Conclusion

Dans ce chapitre, nous avons présenté, interprété et discuté les résultats expérimentaux obtenus par application de notre algorithme de chiffrement évolutionniste, ACEO, sur des messages représentant des passages français ou arabes de différentes tailles.

ACEO a montré une bonne confusion meilleure que celle d'OTL, et il a donné de très bons résultats grâce au caractère aléatoire présent dans toutes les étapes de l'algorithme, et grâce, aussi, au codage proposé et utilisé qui renforce la sécurisation de l'ACEO face aux différents types d'attaques (différentielle, fréquentielle, exhaustive).

Conclusion générale

De nos jours, la cryptographie s'est imposée dans la vie courante notamment pour la protection des transactions. Malgré son antiquité (plus de 3000 ans) elle est toujours à l'état embryonnaire où les vrais algorithmes de chiffrement sont comptés sur les bouts des doigts.

En effet, le but de notre travail présenté à travers ce mémoire était de concevoir et de développer un nouvel algorithme de chiffrement. Ce dernier exploite, dans sa phase principale qui est le chiffrement, les algorithmes évolutionnaires choisis à cause de leur large utilisation du caractère aléatoire très prometteur et très intéressant à exploiter dans un tel domaine.

Et avant de lancer le processus crypto-évolutionnaire, il a fallu définir le codage des individus le plus adéquat vis-à-vis du problème étudié. L'idée était de considérer les nombres d'occurrences de 149 caractères regroupant l'alphabet français, arabe, les chiffres, les symboles de ponctuation et les caractères spéciaux du code ASCII, pour construire les gènes formant les chromosomes qui seront ramené à évoluer d'une génération à une autre.

Grâce au codage proposé, l'algorithme bénéficie d'une forte sécurisation contre les attaques les plus dures, à savoir l'attaque différentielle, fréquentielle et surtout exhaustive. Cela revient au fait que ce codage n'exploite pas d'informations utiles pour la cryptanalyse. De son tour, le caractère aléatoire caractérisant les algorithmes évolutionnaires renforce cette sécurisation en ne laissant aucune trace des différents états intermédiaires menant à la solution.

L'algorithme conçu a été implémenté et testé sur la base de différents messages représentant des passages littéraires arabes ou français de différentes tailles. Les résultats obtenus sont intéressants où ils ont montré un degré confusionnel important meilleur que celui assuré par l'OTL qui est aussi un algorithme de chiffrement évolutionnaire. De même, la clé

de chiffrement utilisée représente l'un des atouts offerts par l'ACEO puisqu'elle a été soigneusement définie. C'est une clé de session qui dépend seulement du message en clair et du chiffré correspondant, ainsi elle ne sera valable qu'une seule fois du fait qu'elle change d'une instanciation du problème à une autre. Et vu que la taille de la clé représente soit un point constructif ou destructif d'un algorithme de chiffrement, ce point est bien pris en considération du fait que la clé générée et utilisée par l'ACEO soit la concaténation de 149 nombres équivalents à une taille égale à 1192 bits. Donc, la seule manière de la retrouver sera d'utiliser une attaque exhaustive exécutant 2^{1192} opérations, ce qui est très coûteux surtout pour n'utiliser cette clé qu'une seule fois.

En conclusion nous pouvons dire que l'algorithme de chiffrement proposé, qui est d'un usage simple ne nécessitant que l'introduction du message à chiffrer pour retenir le chiffré correspondant, présente des résultats très satisfaisants, que ça soit sur le plan cryptographique ou cryptanalytique ; malgré qu'un crypto-système n'a pas besoin d'être absolument incassable pour être utile. Il lui suffit plutôt d'être assez fort pour résister à des attaques pendant la durée de validité des données qu'il protège.

Toutefois, il est important de pouvoir traiter des messages écrits dans des langues qui utilisent un alphabet autre que l'alphabet français ou arabe (l'alphabet chinois ou hébreux par exemple). Pour atteindre cet objectif, nous suggérons faire une extension du codage pour augmenter, aussi, d'avantage la confusion de l'algorithme et compliquer considérablement la tâche des cryptanalystes. Mais dans ce cas le temps de calcul va augmenter et la solution la plus prometteuse sera de faire une parallélisation de l'algorithme en choisissant le modèle en îlots qui semble le plus adéquat. Nous envisageons, donc, de concevoir une version extensible-parallèle de l'ACEO.

Bibliographie

[1] Pierre Jarraud, « Grands principes de la cryptographie », Paris 6 2001. http://ourworld.compuserve.com/homepages/hlifchitz/Renaud/fr/crypto/index.htm

[2] Charles D, Cédric M, « La cryptographie », Cyberworld Awareness Security Enhancement Structure, Avril 2005.

[3] http://www.mines.inpl-nancy.fr/~tisseran/I33_Reseaux/cryptage/cryptage.htm

[4] Brigitte Collard, « La cryptographie dans l'Antiquité gréco-romaine », Folia Electronica Classica, Numéro 7, janvier-juin 2004.

[5] http://www.gpcservices.com/dictionnaire/C/cryptographie.html

[6] Philippe Guillot, « Introduction à la cryptographie », 3ème Ecole Informatique de Printemps (EIP'06), Alger, Algérie, 17-19 juin 2006.

[7] http://www.commentcamarche.net/crypto/crypto.php3

[8] Sébastien Varrette, « Cryptographie: Principe et évolution au cours des âges », Cours de Cryptographie CUT I3-011, Université du Luxembourg, septembre 2004.

[9] Fabrice Theoleyre, « La sécurité des réseaux, Partie 2 : Cryptographie », CITI / INSA Lyon, 2005 / 2006.

[10] Serge Haddad, « Cryptographie », Cours d'IUP-MIAGE 3ème année, Université Paris-Dauphine, 08 mars 2005.

[11] http://www.commentcamarche.net/secu/secuintro.php3

[12] Auguste Kerckhoffs, « La cryptographie militaire », Journal des sciences militaires, Janvier 1883.

[13] http://fr.wikipedia.org/wiki/Principe_de_Kerckhoffs

[14] Douglas Stinson, « Cryptographie, théorie et pratique », International Thomson Publishing France, 1996.

[15] Jacques Stern, Louis Granboulan, Phong Nguyen, David Pointcheval, « Conception et preuves d'algorithmes Cryptographiques », Cours de magistère M.M.F.A.I, École normale supérieure, Edition 2004.

[16] Christophe Bidan, « Introduction à la cryptographie », Supélec : Ecole Supérieure d'électricité, France.

http://www.supelec-rennes.fr/ren/perso/cbidan/cours/crypto.pdf

[17] Bart Prenneel, Vincent Rijmen, « State of the Art in Applied Cryptography », Springer edition, 1998.

[18] http://cryptotpe.free.fr/index.php?rubri=cesar

[19] http://www.apprendre-en-ligne.net/crypto/subst/affine.html

[20] Andreas Enge, « La méthode RSA et la cryptographie fondée sur les courbes elliptiques », XXIe colloque interdisciplinaire du Centre d'Alembert, Université PARIS-SUD 11, mars 2006.

[21] http://fr.wikipedia.org/wiki/Cryptologie

[22] Anne Canteaut, Françoise Lévy, « La cryptologie moderne », L'Armement, 73:76_83, mars 2001.
http://www-rocq.inria.fr/codes/Anne.Canteaut/crypto_moderne.pdf

[23] Anne Canteaut, « Analyse et conception de chiffrements à clé secrète », Mémoire d'HDR, Université Pierre et Marie Curie -Paris 6, 15 septembre 2006.

[24] http://www.techno-science.net/?onglet=glossaire&definition=6155

[25] Alfred J. Menezes, Paul C. Van Oorschot, Scott A. Vanstone, « Handbook of Applied Cryptography », CRC Press, 1996.

[26] Nino Silverio, « Concepts de base des technologies de l'information et de la communication », Cours 1ère année du BTS de l'Ecole de Commerce et de Gestion.
http://homepages.internet.lu/silverio/cours.html

[27] Jacqueline Dousson, « 2021, l'Odyssée quantique », FI-3-99 du 13 avril 1999.

[28] Mélanie Langlois, « Cryptographie quantique - solution au problème de distribution de clefs secrètes », Université d'Ottawa, Décembre 1999.
http://www.apprendre-en-ligne.net/crypto/bibliotheque/

[29] http://www.iforum.umontreal.ca/Forum/ArchivesForum/2004-2005/050214/article4384.htm

[30] http://fr.wikipedia.org/wiki/Cryptographie_quantique

[31] P. Navez, G. Van Assche, « A method for secure transmission: Quantum Cryptography», Technopol IT-Scan, 2002.

[32] http://www.apprendre-en-ligne.net/crypto/quantique/index.html

[33] http://ibsuisse.ch/pages/archives/04.11/04_11_actu_inforum.htm

[34] « Un saut quantique en cryptographie », id Quantique SA, septembre 2004. http://www.swissquantum.ch/technology/article_cryptographie_quantique.pdf

[35] Julien Bourdet, « Le secret des communications assuré par la lumière », 24 avril 2006. http://www.lefigaro.fr/sciences/20060424.FIG000000012_le_secret_des_communications_assure_par_la_lumiere.html

[36] http://physique.quantique.free.fr/cryptographiequantique.htm

[37] Munsch Alexandre, Truong Mai-Hân, « Informatique quantique », Rapport de mini-projet, ENSICAEN, 2005-2006.

[38] Ghislaine Labouret, « Introduction à la cryptographie », 5 novembre 1998. http://www.labouret.net/crypto/

[39] Franck Leprévost, « Les standards cryptographiques du XXIe siècle : AES et IEEE-P1363 », Gazette des Mathématiciens - n°85, Juillet 2000.

[40] François Arnault, « théorie des nombres et cryptographie », Université de Limoges, Cours de D.E.A, mai 2003.

[41] Helyette Damany, Sonia Chaabouni, « Correspondants de guerre dans le cyberespace : Sécurité, espionnage et cryptographie sur Internet », Projet de première année, Ecole des Mines de Nancy, octobre 1996.

[42] Emonet jean-Bruno, « Algorithmes de chiffrement : Mesures de performances réseaux », Centre de Ressources Informatiques, France, 22 juin 2005.

[43] « L'AES en question », Article technique, SCRYPTO. http://www.scryptosystems.com/whitepapers_securite.htm

[44] http://www.gel.ulaval.ca/~klein/maitrise/aes/

[45] http://fr.wikipedia.org/wiki/Standard_de_chiffrement_avanc%C3%A9

[46] SECUNEWS, Sécurité Informatique http://www.secunews.org/Dico.html&lettre=R

[47] Paul Zimmermann, « Cryptographie asymétrique : signature et RSA en détail », 6ème cours du module : Introduction à la cryptologie, Master M1 Informatique, 2005. http://www.loria.fr/~zimmerma/cours/id12-2005-6.pdf

[48] Steve Gury, Nicolas Rémond, « Cryptologie : ElGamal d'après Diffie-Hellman », décembre 2004. http://nire.free.fr/data/ElGamal/elgamal.pdf

[49] Paul Zimmermann, « Chiffrement asymétrique : Panorama et signature », 5ème cours du module : Introduction à la cryptologie, Master M1 Informatique, 2005. http://www.loria.fr/~zimmerma/cours/id12-2005-5.pdf

[50] Pierre Loidreau, « Introduction à la cryptographie », LinuxFocus, Article № 243, 14 janvier 2005.

[51] Christophe Premel, « Compréhension et utilisation d'outils de chiffrement de fichiers (PGP et/ou GPG) », 30 octobre 2000.
http://www.hsc.fr/ressources/breves/gpg.html.fr

[52] http://www.commentcamarche.net/crypto/pgp.php3

[53] « Une introduction à la cryptographie », traduction française: news:fr.misc.cryptologie, 1998. Copyright © 1990-1998 Network Associates, Inc. and its Affiliated Companies.

[54] Fouzia Omary, Abderrahim Tragha, Abdelghani Bellaachia, Aboubakr Lbekkouri, Abdelaziz Mouloudi, « Chiffrement évolutionniste », e-TI la revue électronique des technologies d'information, Numéro 2, 17 avril 2006.

[55] Arnaud Jacques, « les algorithmes de chiffrement cassés ».
http://www.securiteinfo.com/cryptographie/cracked.shtml

[56] Charles Darwin, « The origin of species by means of natural selection, or the preservation of favored races in the struggle for life », New York: D. Appleton and Company, 443 & 445 Broadway.

[57] Grégoire Allaire, Marc Schoenauer, « Conception optimale de structures », Edition Springer, 2006.

[58] Jean-Baptiste Mouret, « Concepts fondamentaux des algorithmes évolutionnistes », Linux Magazine France numéro 76, pp 34-41, 15, novembre 2005.

[59] Vincent Magnin, « Optimisation et algorithmes génétiques », Cyber-cour gratuit de l'EUDIL, 2001.
https://iris.univ-lille1.fr/dspace/bitstream/1908/499/2/

[60] Pierre Collet, Jean-Philippe Rennard, « Handbook of Research on Nature Inspired Computing for Economics and Management: Stochastic Optimization Algorithms », Edition Rennard, J.P, Hershey, IGR, 2006.

[61] Yves Coueque, Julien Ohler, Tollari Sabrina, « Algorithmes génétiques pour résoudre le problème du commis voyageur », Avril 2002.
http://sis.univ-tln.fr/~tollari/TER/AlgoGen1/node5.html

[62] David E. Goldberg, « Algorithmes génétiques : exploration, optimisation et apprentissage automatique », Editions Addison-Wesley France, SA.

[63] Souquet Amédée, Radet Francois-Gérard, « Algorithmes génétiques », TE de fin d'année, Université Nice Sophia Antipolis, juin 2004.

[64] Jean-Philippe Rennard, « Introduction aux algorithmes génétiques », Avril 2000.
http://www.rennard.org/alife/french/gavintr.html

[65] Rémy Dupas, « Amélioration de performance des systèmes de production : apport des algorithmes évolutionnistes aux problèmes d'ordonnancement cycliques et flexibles », HDR, Université d'Artois, 10 décembre 2004.

[66] Simon Baudot-Roux, «OPALE : OPtimisation et contrôle, ALgorithmes numériques et intégration de systèmes complexes multidisciplinaires régis par des E.d.p », Projet commun CNRS-INRIA-UNSA bilocalisé Sophia-Antipolis / Rhône-Alpes.

[67] Evelyne Lutton, « Darwinisme artificiel : une vue d'ensemble», Revue Technique et Science Informatique (TSI), numéro spécial "Méthodologie de la gestion intelligente des senseurs", 2005.

[68] Stéphane Doncieux, « Algorithmes évolutionnistes: de l'optimisation de paramètres à la conception complète d'un système de contrôle», Journées MicroDrones, Toulouse, 2002.

[69] Éric Goubault, Frédéric Nataf, Marc Schoenauer, « calcul parallèle : algorithmes évolutionnaires », École Polytechnique.
http://www.enseignement.polytechnique.fr/profs/informatique/Eric.Goubault/poly/cours009.html

[70] Pierre Collet, « Evolution artificielle et algorithmes évolutionnaires », interview vu par Automatesintelligent, Déclaration cnil n° 1166774, 2005.

[71] Souici Ismahane, Seridi Hamid, Akdag Herman, « Algorithmes évolutionnistes pour chiffrement », Journées Ecole Doctorale & Equipes en réseaux (JED'2007), Annaba, Algérie, 27-28 mai 2007.

[72] Davis L, « Applying Adaptive Algorithms to Epistatic Domains », Proceedings of the International Joint Conference on Artificial Intelligence, pp162-164, 1985.

[73] Grenfenslette J.J, « Optimization of control parameters for genetic algorithms », IEEE translation on system Man and cybernetics, Vol 16 №1, pp122-128, 1986.

[74] A.E.Eiben, J.E.Smith, « Introduction to Evolutionary Computing », Edition Springer, 2003.

[75] Souici Ismahane, Seridi Hamid, Aissaoui Zine, « Nouvel algorithme de chiffrement évolutionniste ACEO (Algorithme de Chiffrement Evolutionniste basé Occurrences) », 3èmes Journées internationales sur l'Informatique Graphique (JIG'07), Constantine, Algérie, 29-30 octobre 2007.

[76] F. Omary, A. Tragha, A. Bellaachia, A. Mouloudi, « Design and Evaluation of Two Symmetrical Evolutionist-Based Ciphering Algorithms », International Journal of Computer Science and Network Security (IJCSNS), VOL.7 No.2, February 2007.