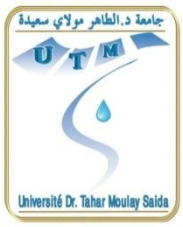
الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التعليم العالي و البحث العلمي
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université Dr. Tahar Moulay SAIDA
Faculté : Technologie
Département : Informatique

جامعة د. الطاهر مولاي سعيدة
كلية التكنولوجيا
قسم : الإعلام الآلي



MEMOIRE DE MASTER

Option : Sécurité informatique et cryptographie

THEME

Contribution à un cryptosystème inspiré de l'ADN « Appliqué sur un texte »

Présenté par

- Mustapha MEFTAH
- Fatima LAKHACHE

Encadré par :

M. Kadda BENYAHIA

2015/2016

RESUME

La cryptographie ADN est un domaine nouveau et prometteur pour la sécurité de l'information. C'est une combinaison des solutions classiques de cryptographie avec les avantages du matériel génétique. En effet, il est possible de bénéficier des avantages des systèmes cryptographiques classiques et de les rendre plus efficaces sur certaines méthodes grâce à l'utilisation de l'ADN. Il y a différentes façons d'utiliser l'ADN pour sécuriser le contenu de l'information. Cette thèse propose deux solutions différentes pour utiliser l'ADN dans la cryptographie sous forme numérique.

D'une part, l'ADN numérique peut être utilisé pour le stockage et pour cacher des données à l'intérieur de celui-ci. L'information secrète est placée dans une molécule de l'ADN. Cette méthode est possible grâce à l'indexation des bases nucléotides dans une séquence génétique.

D'autre part, les nombres aléatoires peuvent être générés à partir de séquences numériques d'ADN. En effet, les bases informatiques de données génétiques contiennent des séquences d'ADN sous forme numérique. Ils représentent une solution pour la génération et la transmission des clés OTP (One-Time-Pad) symétriques. La transmission d'une très longue clé de cryptage n'est pas nécessaire, car chaque séquence possède un numéro d'identification unique dans la base de données. Ce numéro, ou une combinaison de ces numéros, peut alors être transmis.

ملخص

التشفير عن طريق الحمض النووي هو مجال جديد وواعد لأمن المعلومات. بل هو مزيج من حلول التشفير الكلاسيكية مع مزايا المادة الوراثية. في الواقع، من الممكن أن نستفيد من أنظمة التشفير التقليدية وجعلها أكثر فعالية في بعض الطرق من خلال استخدام الحمض النووي.

هناك طرق مختلفة لاستخدام الحمض النووي لتأمين محتويات المعلومات. ونقترح من خلال هذه الأطروحة اثنين من الحلول لاستخدامها للتشفير باستعمال حمض النووي رقمي.

من ناحية، الحمض النووي الرقمي يمكن استخدامه لتخزين و و إخفاء البيانات بداخله. يتم وضع معلومات سرية في جزيء الحمض النووي. هذا الأسلوب هو ممكن بفضل فهرسة قواعد النكليوتيدات في تسلسل الحمض النووي.

من ناحية أخرى، فإن أرقام عشوائية يمكن أن تتولد من تسلسل الحمض النووي الرقمي. في الواقع، قواعد البيانات الجينية تحتوي على تسلسل الحمض النووي في شكل رقمي. أنها تمثل حلا لتوليد ونقل الشيفرة ذات الاستعمال الوحيد المتناظرة. نقل مفتاح التشفير طويل ليس من الضروري، لأن كل تسلسل له رقم تعريف فريد في قاعدة البيانات. ويمكن بعد ذلك نقل هذا الرقم، أو مزيج من هذه الأرقام.

ABSTRACT

DNA cryptographyis is a new and promising field in information security. It is a combination of classical cryptography solutions with the advantages of genetic material. Indeed, it is possible to enjoy the benefits of conventional cryptographic systems and make them more effective on some methods through the use of DNA. There are different ways of using DNA to secure the contents of the information. This thesis proposes two differents solutions for use in DNA cryptography.

On the one hand, the digital DNA can be used for storage and for data hiding inside there of. The secret information is placed in a DNA molecule. This method is possible thanks to the indexing of nucleotide bases in a DNA sequence.

On the other hand, the random numbers can be generated from digital DNA sequences. Indeed, genetic data of computer databases contain DNA sequences in digital form. They represent a solution for the generation and transmission of symmetrical key OTP (One-Time-Pad). Transmitting a long key is not necessary, since each sequence has a unique identification number in the database. This number, or a combination of these numbers may be transmitted.

DEDICACE

Je dédie ce modeste travail spécialement à mes chère parents que j'aurai tant aimé les avoir à mes cotés en ce moment.

Aussi je le dédie à ma femme bien-aimée qui m'a tant soutenue pendant mon parcours, et qui n'a jamais cesser de supporter mes caprices de petit enfant.

A mes petits poussins : Amine, Mayar, Ritadj et Hiba qui ont renoncé à leurs part de temps à mes cotés pour me laisser étudier.

A ceux qui dans nos veines coule le même sang, Mon frère Tayeb et mes sœurs adorées.

A mes oncles, mes tantes, bref à toute ma famille, Je dédie ce mémoire.

Mustapha MEFTAH

DEDICACE

Je dédie ce modeste travail à ma chère mère

Aussi je le dédie à mes frères et mes sœurs.

A mon neveu Mohamed Seddik

A toute ma famille.

A Mon chef de service M. Krim Abdelkader qui ma beaucoup aider pour accomplir ce projet.

Fatima LAKHACHE

REMERCIEMENT

Nous tenons tout d'abords à adresser nos remerciements à notre encadreur M. Kadda Benyahia qui n'a ménagé aucun effort afin de contribuer à l'aboutissement de ce modeste travail.

Nos remerciements vont également à tout l'encadrement du département de l'informatique de l'université Dr. Taher Moulay pour leurs précieux conseils et leurs aides dans notre cursus.

Enfin nous tenons également à remercier toutes les personnes qui ont contribué de prêt ou de loin à la réalisation de ce modeste projet.

Merci du fond du cœur.

TABLE DES MATIERES

INTRODUCTION GENERALE

Motivation et objectifs de la thèse

L'intérêt pour la sécurité de l'information existait depuis l'antiquité et il est présent dans notre vie quotidienne. Les techniques pour protéger l'information évoluent avec le progrès de la technologie de l'information.

Certains des premiers chiffres fondés sur la substitution de lettres étaient Polybe et Cesar. Il ya deux directions de la protection de l'information: la cryptographie et de stéganographie. Ces deux sciences manipulent l'information afin de changer sa signification ou de cacher son existence. L'apparition des l'ordinateurs a donné une interprétation différente de l'information et de nouvelles orientations dans le développement des algorithmes de chiffrement et des protocoles cryptographiques sont apparus. La puissance de calcul offre la possibilité de construire des algorithmes nouveaux et robustes, mais aussi un outil fort utilisé par les cryptanalystes pour briser les systèmes cryptographiques. Cela signifie que le sujet de trouver des chiffres nouveaux et puissants est toujours d'intérêt et de nouvelles orientations en cryptographie sont explorées.

La cryptographie offre une gamme de fonctionnalités pour la sécurité de l'information. Les principaux aspects traités par la cryptographie sont : la confidentialité, l'intégrité des données, l'authentification et la non-répudiation. Les objectifs de cette thèse se sont concentré sur la confidentialité et de trouver de nouvelles méthodes (algorithmes de chiffrement) pour assurer la confidentialité grâce à l'utilisation de l'ADN.

La cryptographie ADN consiste en l'utilisation de la génétique et du calcul biomoléculaire et il est l'un des nouvelles orientations en cryptographie. Le matériel génétique tel que l'ADN peut être utilisé comme un vaste espace de stockage. Cette idée est inspirée du fait que l'ADN est un support naturel de l'information qui est codée par un alphabet à 4 lettres: A, C, G et T. Cet alphabet peut être facilement transposé dans l'alphabet binaire (A - 00, C - 01 G - 10, T - 11). Par conséquent, l'ADN peut être utilisé comme un support de stockage pour chaque type d'information. L'informatique ADN a commencé avec la recherche de Adleman **[Adl94]**.

Compte tenu du fait que la cryptographie ADN est un nouveau domaine, l'un des objectifs de la thèse est de trouver et de définir différentes façons dont il peut être appliqué dans la sécurité de l'information. Trois orientations principales de l'utilisation de l'ADN dans la cryptographie ont été trouvées : espace de stockage, puissance de calcul, génération de clés cryptographiques de ses longues séquences. Il peut y avoir deux environnements de travail avec de l'ADN : au niveau moléculaire, dans un laboratoire, avec l'ADN biologique et avec l'ADN numérique à l'aide de bases de données génétiques disponibles. Différentes techniques pour manipuler l'ADN au niveau moléculaire ont été analysées. Une conversion de son alphabet binaire a été établie.

Le séquençage du génome et leur apparition sous la forme de bases de données électroniques était une étape importante pour la croissance dans le domaine de la recherche en génomique. Les avantages des bases de données génomiques numériques peut aussi être étendue au domaine de la sécurité de l'information. Par exemple, ces bases de données peuvent être utilisées pour l'application pratique du schéma de chiffrement One-time pad (OTP). Les propriétés OTP correspondent aux caractéristiques du système de chiffrement incassable définies par Claude Shannon **[Sha49]**.

La variété des gènes et chromosomes de différentes espèces sont de bons matériaux pour la création d'une seule plaquette d'utilisation aléatoires. Aujourd'hui, il existe des bases de données électroniques de génomes séquencés entiers de différentes espèces, y compris l'homme, chien, souris, grenouille, mouche des fruits, amibe sociale et bien d'autres. Ces séquences peuvent être accessibles à partir de bases de données génétiques publiques **[wncbi]** dans différents formats..

Cette thèse contient deux contributions. La 1ère est une amélioration d'une approche qui existe déjà nommée « Indexation ADN ». La 2ème aborde le problème du schéma de chiffrement OTP. Il offre une sécurité forte, mais ne sont pas utilisées dans des applications pratiques en raison des difficultés pour générer des clés purement aléatoire.

Structure de la thèse

Cette thèse est organisée en 4 chapitres.

Une introduction générale.

Le chapitre 1 est une introduction au sujet de la thèse et les objectifs.

Le chapitre 2 offre un bref aperçu et l'état de l'art pour les domaines scientifiques impliqués dans ce travail: la cryptographie, la biologie moléculaire et la bioinformatique. En raison de la diversité des domaines concernés, un brève contexte théorique est prévu pour eux avec les références importantes de livres et les documents pertinents pour les domaines décrits.

Le chapitre 3 présente deux contributions à un crypto-système à base d'ADN : Stegano-ADN & VernADN. Se sont deux algorithmes de chiffrement symétrique qui utilisent les séquences ADN à partir de bases de données génétiques comme clé secrète. Ils ont en commun le codage en bases nucléotides et le brouillage inspiré de DES et ENIGMA. Stegano-ADN utilise le principe de l'indexation ADN et VernADN utilise le principe de l'OTP.

Le chapitre 4 décrit la mise en œuvre des algorithmes proposés, ainsi que les résultats des différents tests effectués dans le but d'évaluer leurs performances.

Une conclusion générale conclut le travail de thèse. Après, une bibliographie est donnée.

Contenu en bref

I.1 Introduction

I.2 Généralités sur la cryptographie

I.3 Historique

I.4 Les éléments de la cryptographie

I.5 Compression des données

I.6 Conclusion du chapitre

I.1 Introduction :

La cryptologie, étymologiquement la science du secret, englobe la cryptographie, l'art des écritures cachées, et la cryptanalyse dont le but n'est autre que d'attaquer les méthodes cryptographiques. **[DUO05]**

La cryptographie à son tour se divise en deux grandes branches, des systèmes de chiffrement symétriques et asymétriques. Le chiffrement symétrique est caractérise par des clés de chiffrement et de déchiffrement identiques. La confidentialité d'un message repose alors uniquement sur le secret de la clé partagée par les deux interlocuteurs. Quand-t-au chiffrement asymétrique, la clé de chiffrement et de déchiffrement sont différents. La clé de chiffrement (clé publique) est connue de tous et la sécurité du système repose sur le secret de la clé de déchiffrement (clé privée) et sur l'impossibilité de la déduire à partir de la clé de chiffrement. **[BOU13]**

La cryptographie, est devenue aujourd'hui une science à part entière. Au croisement des **mathématiques**, de l'**informatique**, et parfois même de la **physique**, elle permet ce dont les civilisations ont besoin depuis qu'elles existent : « le maintien du secret ». Pour éviter une guerre, protéger un peuple, il est parfois nécessaire de cacher des choses... **[MEL01]**

I.2 Généralités sur la cryptographie :

Avant d'entamer cette thèse, il est impérativement important de définir certain notions très utilisé en la matière.

I.2.1. Cryptologie :

Le préfixe « crypto » provient du grec « kryptos » qui signifié «caché» ou «secret» donc la cryptologie signifié la science du caché ou la science du secrets. Se décompose en deux sciences distinctes, **cryptographie** et **cryptanalyse**.

I.2.2. Cryptographie :

Le mot vient du grec « crypto » qui signifie «caché» ou «secret», et «graphie» qui signifie « écriture », donc c'est une manière de masquer l'écriture tout en préservant un moyen de la retrouver**. [MEL01]**

I.2.3. Cryptanalyse :

Est la science qui consiste à tenter de déchiffrer un message ayant été chiffré sans posséder la clé de chiffrement. Le processus par lequel on tente de comprendre un message en particulier est appelé une *attaque*. **[MEL01]**

I.2.4. Chiffrement et déchiffrement :

Le chiffrement consiste à transformer une donnée afin de la rendre incompréhensible par une personne autre que celle autorisée. La fonction permettant de retrouver le texte clair à partir du texte chiffré porte le nom de déchiffrement. **[REN07]**.

I.2.5. Texte clair

Texte original intelligible tel qu'il se présentait avant tout chiffrement.

I.2.6. Texte chiffré (cryptogramme)

C'est le texte obtenu après avoir appliqué le chiffrement sur le texte clair. **[NAS14]**.

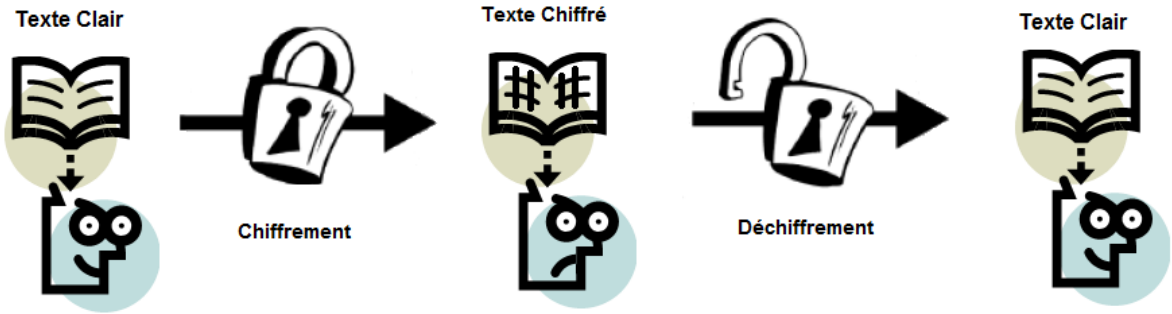


Figure 1.1 : Principe Chiffrement / Déchiffrement

I.2.7. La clef :

Dans un système de chiffrement, elle correspond à un nombre, un mot, une phrase, etc. qui permet, grâce au processus de chiffrement, de chiffrer ou de déchiffrer un message.

- **Les clés symétriques**: il s'agit de clés utilisées pour le chiffrement ainsi que pour le déchiffrement. On parle alors de chiffrement symétrique.
- **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. **[REN07]**

I.2.8. Stéganographie :

La stéganographie est l'art de la dissimulation : son objet est de faire passer inaperçu un message dans un autre message. Pour prendre une métaphore, la stéganographie consisterait à enterrer son argent dans son jardin là où la cryptographie consisterait à l'enfermer dans un coffre-fort — cela dit, rien n'empêche de combiner les deux techniques, de même que l'on peut enterrer un coffre dans son jardin. **[AD58]**

I.2.9. La signature numérique :

La **signature numérique** (parfois appelée **signature électronique**) est un mécanisme permettant de garantir l'intégrité d'un document électronique et d'en authentifier l'auteur, par analogie avec la signature manuscrite d'un document papier.

I.2.10. Fonction de hachage :

On nomme **fonction de hachage**, de l'anglais *hash function* (*hash* : pagaille, désordre, recouper et mélanger), une fonction particulière à sens unique qui, à partir d'une donnée fournie en entrée, calcule une *empreinte* servant à identifier rapidement la donnée initiale. Ex : MD5, SH-1, SH-2 …

I.3. Historique :

I.3.1 La scytale :

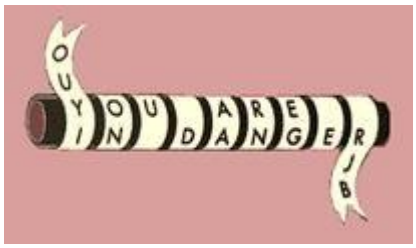


Figure 1.2 : La Scytale

La scytale est un support cylindrique souvent en bois et utilisé pour crypter des messages.

- La clé de cryptage est dans ce cas le diamètre de la scytale.
- La méthode assyrienne est un système de chiffrement par transposition utilisant une scytale.
- Ce procédé était utilisé dans les environs de 600 avant J.C et son fonctionnement consistait à enrouler une bande de papyrus autour de la scytale, et d'écrire un message à sa surface de façon à rendre le message illisible une fois déroulé. **[AD58]**

I.3.2. Le Chiffre de César :

Jules César était un général, homme politique et écrivain romain, né à Rome 100 av. Il utilisait une méthode de chiffrement qui porte aujourd'hui son nom.

Le chiffre de César est la méthode de cryptographie la plus ancienne communément admise par l'histoire. Il consiste en une **substitution mono-alphabétique** : chaque lettre est remplacée ("substitution") par une *seule* autre ("mono-alphabétique"), selon un certain décalage dans l'alphabet ou de façon arbitraire. César avait coutume d'utiliser un décalage de 3 lettres. **A** devient **D**, **B** devient **E**, **C** devient **F**, etc. Il écrivait donc son message normalement, puis remplaçait chaque lettre par celle qui lui correspondait : **[AD58]**

CLAIR	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
décalage = 3																										
CODE	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

exemple : d'après cette méthode, "VIVE LES MATHS" devient donc "YLYH OHV PDWKV"

I.3.3. Le Chiffre de Vigenère :

Correspondance lettre ⇔ nombre. A = 0; B = 1… Z = 25

Addition sur les lettres. J + W = F (9 + 22 mod 26 = 5)

Exemple :

```
NOUS ATTAQUERONS AU MATIN PAR LE NORD
VIGE NEREVIGENER EV IGENE REV IG ENER
-----------------------------------
KYCY PZMGNEMXDTL GR WIZXT IGO VM TDXW
```

[AD58]

I.3.4. Le chiffre des Templiers :

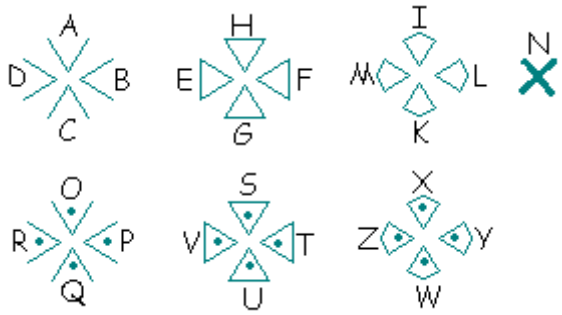


Figure 1.3 : Dictionnaire de substitution du chiffre des templiers

Le Temple était un ordre de moines fondé au XIIe siècle, dont la mission était d'assurer la sécurité des pèlerins en Terre Sainte. Par la suite, tellement enrichis, les Templiers devinrent les trésoriers du Roi et du Pape et prirent l'habitude de chiffrer les lettres de crédit qu'ils mettaient en circulation. Leur alphabet de chiffrement était déduit de la croix dite "des huit béatitudes" qui constituait l'emblème de l'ordre. **[AD58]**

I.3.5. La machine Enigma

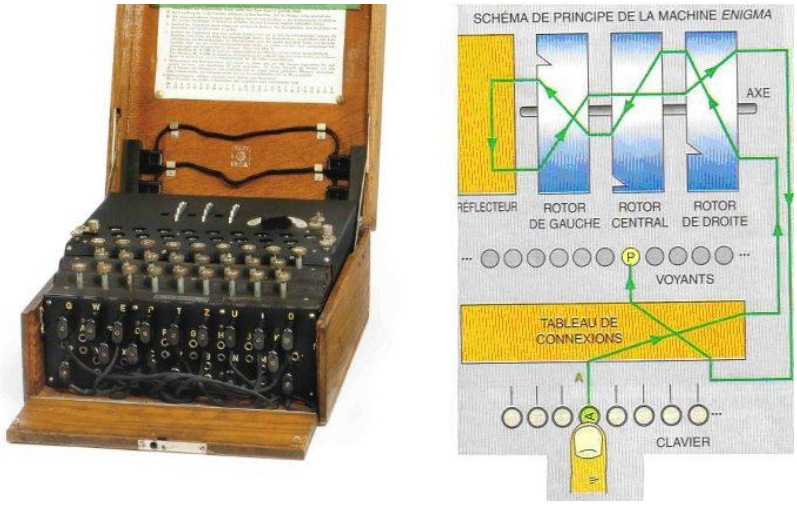


Figure 1.4 : La machine ENIGMA

La machine allemande Enigma a joué un grand rôle pendant la guerre de l'Atlantique, et son décryptage par les Alliés leur a assuré bon nombre de victoires (notamment parce que les Allemands ne se doutaient pas que leurs messages étaient déchiffrés).

Enigma ressemble à une machine à écrire : on frappe le texte clair sur un clavier, et des petites lampes s'allument pour éclairer les lettres résultant du chiffrement.

Le principe de chiffrement qu'utilise Enigma est à la fois simple et astucieux. Simple, car il ne s'agit ni plus ni moins d'une substitution de lettres, par exemple, A devient Q, P devient N, etc. Et astucieux, parce que **la substitution change d'une lettre à une autre** : si la lettre A correspond à Q la première fois qu'on la saisit, elle pourrait correspondre à M, K, H, ou tout autre lettre différente de Q à la fois suivante (ce principe est possible grâce à un système de rotors).

De plus, un autre avantage non négligeable que possède Enigma est la réversibilité : si on tape le message clair, on obtient le message code, et avec le message codé, on obtient le message clair. **[MOV96]**

I.4. Principes de Kerckhoffs :

En 1883 dans un article paru dans le Journal des sciences militaires, **[AUG83]**, Auguste Kerckhoffs (1835-1903) posa les principes de la cryptographie moderne.

Ces principes et en particulier le second stipulent entre autre que la sécurité d'un crypto-système ne doit pas reposer sur le secret de l'algorithme mais qu'elle doit uniquement reposer sur la clef secrète du crypto-système qui est un paramètre facile à changer, de taille réduite, et donc assez facile à transmettre secrètement. Ce principe a été très exactement respecté pour le choix du dernier standard de chiffrement, l'algorithme symétrique AES, par le NIST. Ce dernier a été choisi à la suite d'un appel d'offre international et tous les détails de conception sont publics. Ce principe n'est que la transposition des remarques de bon sens suivantes:

- Un crypto-système sera d'autant plus résistant et sur qu'il aura été conçu, choisi et implémenté avec la plus grande transparence et soumis ainsi à l'analyse de l'ensemble de la communauté cryptographique.

- Si un algorithme est supposé être secret, il se trouvera toujours quelqu'un soit pour vendre l'algorithme, soit pour le percer à jour, soit pour en découvrir une faiblesse ignorée de ses concepteurs. A ce moment l`a c'est tout le crypto-système qui est appelé à changer et pas seulement la clé. Les systèmes conçus dans le secret révèlent souvent rapidement des défauts de sécurité qui n'avaient pas été envisagés par les concepteurs. **[DAN10]**

I.5 Les éléments de la cryptographie :

La cryptographie est la science des techniques et des protocoles destinés à assurer la sécurité de l'information. Cryptanalyse est l'art d'analyser et de briser une communication sécurisée; ça consiste à attaquer des crypto-système.

La cryptographie est pratiquée par des cryptographes et la cryptanalyse est pratiquée par les cryptanalystes. La cryptologie englobe la branche de la cryptographie et la cryptanalyse **[Sch96]**.

Le but de la cryptographie est de fournir une gamme de fonctionnalités pour la sécurité de l'information. Les plus importants d'entre eux sont **[Den07]**:

- **Confidentialité :** fournit le secret du contenu de l'information. il transforme des données significatives en un message insensé. Les *Ciphers* sont les algorithmes cryptographiques utilisés pour garantir cette caractéristique.

- **L'intégrité des données :** signifie que sa protection contre l'accès non autorisé et l'altération. Les changements possibles dans les données d'origine sont effacement et insertion. L'intégrité est assurée par fonctions de hachage cryptographique.

- **Authentification :** est l'identification et l'assurance de l'origine des informations transmises et des pièces engagées dans une communication.
- **La non-répudiation :** signifie de respecter les obligations d'un contrat. Cette propriété peut être obtenue en utilisant des signatures.

Les données qui ont un sens compréhensif pour nous, est appelé texte clair ou texte brut. La transformation du texte clair en un fichier illisible est appelé chiffrement. Un texte clair devient un texte chiffré. Afin d'obtenir le texte clair d'origine, le processus de déchiffrement est appliqué sur les données du texte chiffré. Ces deux procédés de chiffrement et de déchiffrement sont des parties qui composent un algorithme cryptographique nommé « cipher » ou « cryptogramme ». Un procédé de chiffrement est appliqué sur l'ensemble des données avec une clé secrète. Un Crypto-système inclut le chiffrement, déchiffrement et l'algorithme associé et tous les protocoles nécessaires pour assurer une communication sécurisée **[PGP04]**. L'espace clé est le nombre d'éléments de l'alphabet élevé à la puissance de la longueur de la clé. L'espace de clé est important pour la cryptanalyse; elle donne le nombre de toutes les clés possibles qui peuvent être la clé secrète.

I.6 Les types de la cryptographie

I.6.1 Cryptographie symétrique

La cryptographie symétrique est également nommée conventionnelle ou classique, car elle a été la première et unique type de cryptage jusqu'au années 1970, lorsque la cryptographie à clé publique était introduite. L'algorithme de chiffrement symétrique utilise la même clé pour le cryptage et le décryptage. Cette clé doit être secrète et communiqué via un canal sécurisé à toutes les parties impliquées dans une communication.

Les opérations les plus couramment utilisés dans ce type de cryptographie sont substitution et transposition. La substitution consiste à donner une valeur différente à chaque mot du texte clair. Les mots en clair transformées deviennent des valeurs de texte chiffré. La Transposition, également nommée permutation, consiste à échanger les positions des mots du texte clair.

Les premiers chiffres sont apparus dans les années AJ dans l'Egypte ancienne, la Grèce et Rome. Ils étaient destinés à sécuriser un texte écrit et basés sur des substitutions et des transpositions des lettres de l'alphabet. Un exemple bien connu des premiers cryptogrammes sont « Le chiffre de César ». Dans le XV e siècle, les chiffres poly-alphabétiques, comme « Vigenère », sont apparus. Ils étaient résistants aux techniques d'analyse de fréquence pour laquelle les chiffres anciens étés vulnérables. **[MOV96]**

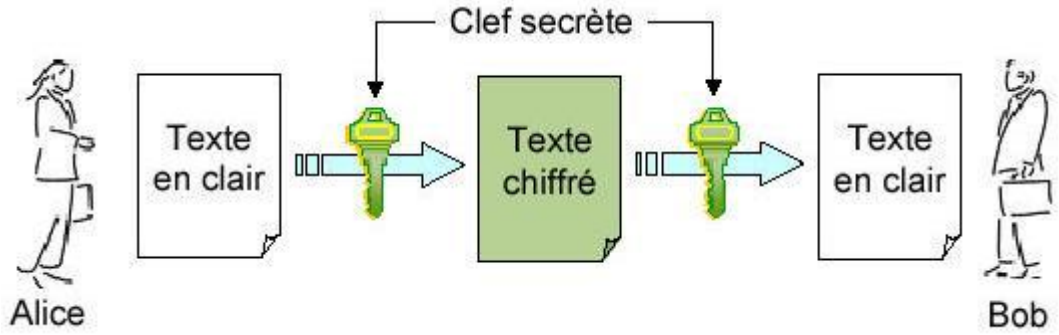


Figure 1.5 Modèle de cryptage symétrique.

L'évolution de l'informatique a apporté différents niveaux de cryptographie. La sécurité à commencé à être appliquée sur le flux binaire, plutôt que sur les lettres de l'alphabet et par conséquent sur tout type de données: texte, image, vidéo, etc.

La cryptanalyse des premiers crypto-systèmes est devenu plus rapide et plus facile avec l'essor de la technologie et l'apparition de nouvelle force de calcul. En 1972, le NIST[1] a lancé un programme de sécurité informatique pour introduire des algorithmes cryptographiques certifié, afin de les utiliser comme standard dans la sécurité de l'information **[NIST01]**. En 1976, le Data Encryption Standard (DES), un algorithme développé par IBM en 1970, a été adopté comme norme fédérale par NIST. En raison de l'augmentation de la puissance de calcul dans les années 1990, la vulnérabilité de DES est devenu sa clé de chiffrement courte. Il a été remplacé par 3DES, qui est le DES appliqué 3 fois. La sécurité de 3DES était forte, mais il a été jugé lent. En 2002, l'algorithme de chiffrement AES a été adopté comme norme fédérale par le NIST et il est encore en usage.

OTP : Le One-Time Pad / Le masque jetable

One-Time-Pad (OTP) est un principe de génération de clé appliquée sur le chiffrement par flux , méthode qui offre un secret parfait si toutes les conditions sont remplies. Il est également considéré incassable en théorie, mais difficile à réaliser dans les applications pratiques. **[SCH09]**.

Une partie de la méthode de chiffrement OTP est apparu en 1917 par Vernam. **[Ver26]**. Il défini un chiffrement par flux. Une clé stockée sur un ruban perforé, a été combinée, et une opération XOR a été appliquée caractère par caractère du message clair qui a produit un texte chiffré. J. Mauborgne[2] a ajouté que la chaîne de la clé doit être véritablement aléatoire et à usage unique. Ensuite le chiffre de Vernam est devenu OTP co-inventée par G. Vernam et J. Mauborgne. Le principe de ce chiffre est représenté sur la Figure 1.6.

[1] **National Institute of Standards and Technology**, est une agence du département du Commerce des États-Unis. Son but est de promouvoir l'économie en développant des technologies, la métrologie et des standards de concert avec l'industrie. [Wikipedia]

[2] Le général **Joseph Mauborgne**, né le 26 février 1881 à New York et mort le 7 juin 1971, fut un officier de transmissions et un cryptanalyste de l'armée de terre américaine. [Wikipedia]

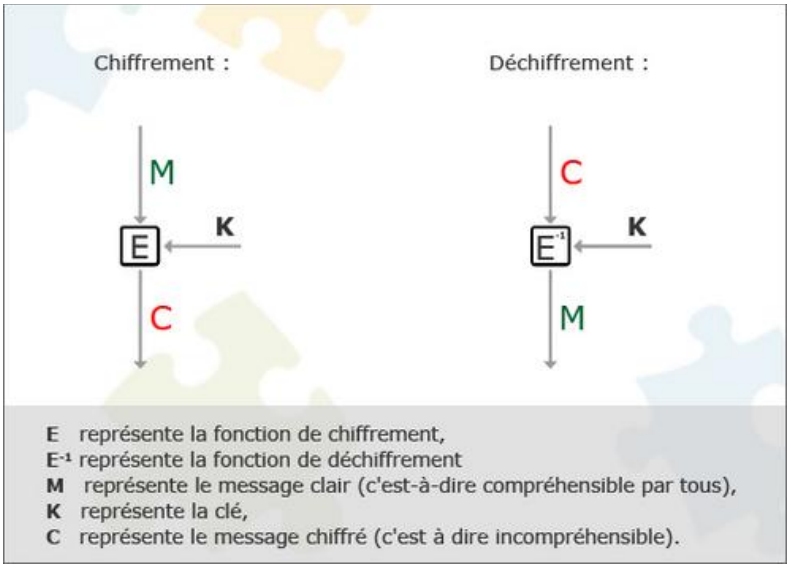


Figure 1.6 : Le Chiffre de Vernam

$C_i = M_i$ XOR K_i M_i = est le i ème bit du bitstream du texte clair

K_i = est le i ème bit de la clé C_i = est le i ème bit du texte chiffré

Claude Shannon a décrit dans son œuvre « les principes du secret parfait ». Ces caractéristiques pour le système de chiffrement incassables sont les même avec OTP. Ils peuvent être résumés dans les contraintes suivantes concernant la clé de chiffrement :

- Elle doit être absolument aléatoire.
- Au moins aussi longue que le texte clair.
- Jamais réutilisée en tout ou une partie.
- Un secret bien gardé.

En fait, il ya des débats contradictoires sur le schéma de chiffrement OTP. Par exemple, Bruce Schneier[1], a déclaré à propos de OTP en 1996 ce qui suit :

« Croyez-le ou non, il y a un schéma de chiffrement parfait. »

« En supposant une oreille indiscrète ne peut pas avoir accès au one-time-pad utilisé pour chiffrer le message, ce système est parfaitement sécurisé. »

« Une séquence clé aléatoire ajoutée à un message en clair non aléatoire produit un message chiffré complètement aléatoire et aucune quantité de puissance de calcul ne peut changer cela. » **[Sch96]**

En 2002, il a écrit sur OTP dans une perspective tout à fait différente :

[1] **Bruce Schneier**, né le 15 janvier 1963 à New York, est un cryptologue, un spécialiste en sécurité informatique et un écrivain américain. Il est l'auteur de plusieurs livres sur la cryptographie et il est le fondateur de la société « Counterpane Internet Security ». [www.futura-sciences.com]

« Les OTP sont inutiles pour toutes les applications, mais très spécialisés, principalement historique et non informatiques. »

« Ils remplacent un problème cryptographique que nous en savons beaucoup sur la résolution - la façon de concevoir des algorithmes sécurisés - avec un problème d'implémentation que nous avons très peu d'espoir de résoudre. Ils ne sont pas l'avenir. Et vous devriez regarder quiconque affirme le contraire avec suspicion profonde. » **[Sch02]**

Ce qui est sûr c'est que ce schéma de chiffrement a reçu beaucoup d'attention.

I.6.2 Cryptographie asymétrique

Le principe de base de la cryptographie asymétrique (à clé publique) est d'utiliser une paire de clés secrètes. Dans de tels crypto-systèmes, le chiffrement est effectué avec une clé publique et le déchiffrement avec sa paire clé privée. La clé de chiffrement (publique) est différente de la clé de déchiffrement (privé), mais ils sont fortement liés.

$$K_E \neq K_D$$

Cette paire de clés est générée à l'aide des fonctions mathématiques et dans certains algorithmes, comme RSA, l'une de ces deux clé peut être rendue publique, et utilisé pour le chiffrement, ce qui signifie que l'autre (clé privée) doit être gardée secrète et utilisée pour le déchiffrement. Il est mathématiquement impossible pour dériver la clé secrète et connaître sa paire publique.

Les crypto-systèmes symétriques sont rapides, ils sont utilisés pour crypter de grandes quantités de données, mais avant de les utiliser, la transmission de la clé secrète doit ce faire par un canal sécurisé. Cette partie peut être résolue par des algorithmes asymétriques qui ne sont pas si rapide, mais ils peuvent crypter une clé secrète d'un algorithme symétrique, sans échange de clés précédente.

C'est la notion de la cryptographie hybride, sa consiste à combiner la rapidité d'un crypto-système symétrique avec la force d'un crypto-système asymétrique. Sécuriser la communication avec un crypto-système symétrique en protégeant la clé avec un crypto-système asymétrique.

Le concept de cryptographie à clé publique a été introduit par Diffie et Hellman[1] **[DH76]**. Un des algorithmes à clés publiques les plus connues est RSA inventé par R. Rivest, A. Shamir et L. Adleman **[RSA78]**. La force de ce chiffrement est donnée par la complexité de calcul de la factorisation de grands nombres premiers. Un autre chiffrement asymétrique est ElGamal créée par Taher ElGamal **[ElG85]**. Cet algorithme, comme

[1] **Bailey Whitfield Diffie** (né le 5 juin 1944) - **Martin E. Hellman** (2 octobre, 1945 -)
En 1976, ils publient « *New Directions in Cryptography* ». La méthode révolutionnaire décrite dans cet article permet de résoudre un problème fondamental en cryptographie : la distribution des clés. Cette méthode sera par la suite renommée en méthode d'échange de clés Diffie-Hellman. Ce principe est aussi à l'origine de méthodes à clés asymétriques plus évoluées comme le RSA ou ElGamal [Wikipedia].

Diffie-Hellman, est basée sur la difficulté de résoudre le problème de logarithmes discrets. L'algorithme de signature numérique (DSA) a été déclarée une norme (DSS Digital Signature Standard) par le NIST, attribué comme invention à David Kravitz. **[FIPS94]**.

La cryptographie à clé publique est basée sur des fonctions mathématiques et beaucoup sur la théorie des nombres **[Sta11]**. La sécurité de chiffrement asymétrique se base sur la recherche des grands nombres premiers, au moins 100 chiffres décimaux. La multiplication des grands nombres premiers est un calcul simple, mais trouver les numéros originaux étant donné le produit final en un temps acceptable est une tâche très difficile.

Les formules mathématiques suivantes décrivent le principe de l'algorithme de chiffrement RSA. Laissant envisager deux grands nombres premiers « p » et « q » et leur produit est « n ».

1) $n = p * q$

Ensuite on calcul l'indicatrice d'Euler, trouver un entier « e » et un entier « d » tel que:

2) $\varphi(n) = (p-1)(q-1)$
3) $d * e \bmod \varphi(n) = 1$
4) $C = P^e \bmod n$
5) $P = C^d \bmod n$

Cet algorithme offre la clé publique (n, e) pour le chiffrement et la clé privée (n, d) pour le déchiffrement (d est secret). Le nombre entier n est le produit de deux nombres premiers (1), tandis que d et e mathématiquement dériver de n (3). C est le texte chiffré et P est le texte clair.

Le Chiffrement (4) et le déchiffrement (5) sont des processus de calcul simples, sachant les clés (n, e et n, d).

I.6.3 Chiffrement par flux et Chiffrement par bloc

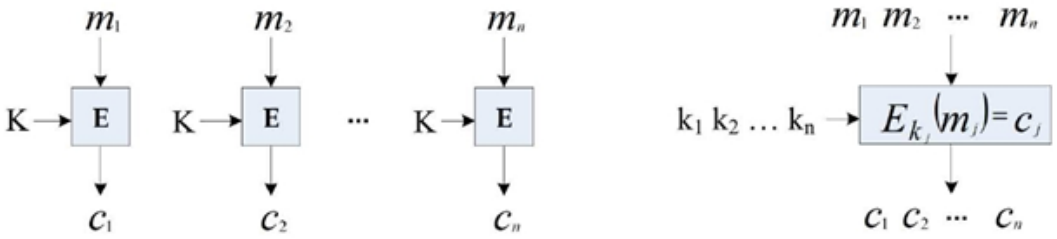
Les algorithmes symétriques peuvent être de 2 types: les chiffrements par flux et le chiffrement par blocs. Dans les chiffrements par flux, le cryptage est effectué sur les petites unités de texte clair, comme un bit ou un octet à la fois. Le chiffrement par bloc fonctionne sur de plus grandes unités de message, comme des blocs de 64 bits dans DES.

La distinction fondamentale entre ces deux types d'algorithmes de chiffrement est en fonction du temps ou de la «mémoire».

Prenons le message clair $M=(m_1 , m_2 , m_3 \dots m_n)$ comme une séquence de blocs et correspondant à ce texte chiffré $C = (c_1 , c_2 , c_3 \dots c_n)$ une séquence de blocs après le chiffrement.

Le chiffrement par bloc est sans mémoire, elle transforme indépendamment chaque unité de message claire en une unité de texte chiffré en utilisant une fonction de

chiffrement et une clé secrète (Figure 1.7, a). Le chiffrement par flux transforme une unité de un texte clair en une unité de texte chiffré en utilisant une clé en fonction du temps et une fonction de chiffrement (Figure 1.7, b). Le chiffre de Vernam[1] basé sur le principe de OTP est l'un des chiffres par flux les plus connues.



a) Chiffrement par bloc **b) : Chiffrement par flux**

Figure 1.7 Chiffrement par bloc / chiffrement par flux

En général, deux unités identiques en clair chiffré avec un chiffrement par flux ne possèdent pas la même sortie de texte chiffré parce que la séquence de touches est différente à différents moments de cryptage. En cas de chiffrement par bloc, deux blocs de texte clair identiques peuvent entraîner après chiffrement à un même texte chiffré qui apporte la vulnérabilité à l'attaque clair connu et de la vulnérabilité à l'insertion ou la suppression de certains texte chiffré. Afin de résoudre ce problème, un certain enchaînement ou de rétroaction peut être introduit entre les blocs de cryptages. Certains modes possibles pour le faire sont: Cipher Block Chaining (CBC), Cipher Feedback Mode (CFB), Output Feedback (OFB), etc. **[Bor11]**.

I.7 La cryptanalyse

L'objectif de la cryptanalyse est de briser le système de chiffrement en exploitant ses faiblesses. Révéler la clé secrète est équivalent à une rupture totale du chiffre. Trouver un moyen de récupérer le texte en clair à partir du texte chiffré sans connaître la clé secrète est considéré comme une rupture partielle. **[Knu94]**.

La façon simple de trouver la clé secrète est par une attaque par force brute qui consiste à essayer toutes les clés possibles. Par le principe de Kerckhoff[2] et la maxime de

[1] **Chiffre de Vernam** également appelé le **masque jetable**, est un algorithme de cryptographie inventé par Gilbert Vernam en 1917 et perfectionné par Joseph Mauborgne, qui rajouta la notion de clé aléatoire. Bien que simple, facile et rapide, tant pour le codage que pour le décodage, ce chiffrement est théoriquement impossible à casser, même s'il présente d'importantes difficultés de mise en œuvre pratique. [www.futura-sciences.com]

[2] **Auguste Kerckhoffs von Nieuwenhoff** (19 janvier 1835 - 9 août 1903) est un cryptologue militaire néerlandais. Il entama des études à l'Université de Liège, où il obtient le grade de Docteur en Lettres. Après

Shannon, l'algorithme de chiffrement est censé être connu et la force du système de chiffrement est dans la clé. Ainsi, l'espace clé doit être assez grand rendant l'attaque de force brute infaisable. Les techniques de cryptanalyse sont utilisées pour trouver la clé avec un nombre d'essais le plus petit possible en cherchant les défauts dans le crypto-système. Les attaques des cryptanalyses peuvent être classifiées par la quantité des informations connues par l'attaquant **[Koh08]**:

- Attaque cryptogramme seulement : l'algorithme de chiffrement et le texte chiffré sont disponibles pour la cryptanalyse.
- Attaque texte clair connu : l'algorithme de chiffrement, le texte chiffré, et une partie du text claire correspondant au texte chiffré sont connus pour le cryptanalyste.
- Attaque texte clair choisi (cryptogramme choisi): l'attaquant est capable de crypter ou décrypter toute information de son choix et obtenir une paire (texte clair - texte chiffré). Ce qui se produit lorsque le système de chiffrement avec la clé secrète intégré est disponible au cryptanalyste.
- Attaque adaptative texte clair ou chiffré choisi : est la version adaptative de l'attaque présentée ci-dessus. L'attaquant peut adapter le texte clair à partir des résultats obtenus de cryptages précédents.
- Attaque clés liés: est basée sur la relation entre les différentes clés connues des attaquants.

Lorsque le cryptanaliste n'a que le texte chiffré, afin de réduire le nombre de tentatives à travers l'attaque par force brute, les données de texte chiffré sont analysées et généralement une statistique des tests est appliquée sur elle. La connaissance des types de données, telles que l'anglais ou d'autres langues, texte, image, code source, etc. donne la possibilité d'effectuer l'attaque texte clair connu. Certains modèles dans le texte, en-têtes qui apparaissent au début du fichier, etc. offres des parties de texte en clair - des paires de texte chiffré correspondant à la clé secrète.

une période où il enseigna en France et dans les Pays-Bas, il devint professeur d'allemand à l'École des Hautes Études Commerciales et à l'École Arago. [Wikipedia]

I.8 Crypto-systèmes pour Duplex transmission et stockage

La conception d'un crypto-système impose le choix des techniques et des méthodes appropriées pour assurer la confidentialité des messages, l'intégrité et l'authenticité. Il comprend également l'authentification des parties impliquées dans la communication sécurisée, la non-répudiation, l'autorisation. **[Bor11]**.

Duplex communication est un échange d'informations entre les deux partis dans les deux sens. La transmission half-duplex ou full-duplex. La transmission half-duplex n'est pas simultané; ce qui signifie qu'une seule entité dans une communication transmet, le destinataire doit attendre et ne peut pas transmettre des informations en même temps. Dans la communication full-duplex, la transmission est continue; elle peut être réalisée dans les deux sens en même temps. **[BTT13]**.

La Figure 1.8 présente un schéma de système de chiffrement pour la transmission sécurisée des cryptogrammes et des clés secrètes. Dans ce système, chaque nouvelle communication sécurisée des messages est effectuée avec une nouvelle clé, dans le cadre du principe OTP. Elle montre une communication en duplex entre deux parties A et B. Les lignes continues montrent la transmission de la partie A à B et la ligne pointillées dans le sens opposé. Lorsque la partie A veut transmettre des informations cryptées à la partie B, il génère d'abord la clé (K_A), puis crypte la clé en utilisant un algorithme symétrique ou public (bloc K_AE), transmet la clé à la partie B, puis crypte le message en utilisant cette clé K_A (M_A) = C_A. B décrypte la clé secrète, puis l'utilise pour le décryptage du message. La partie B suivra les mêmes étapes afin de transmettre son message.

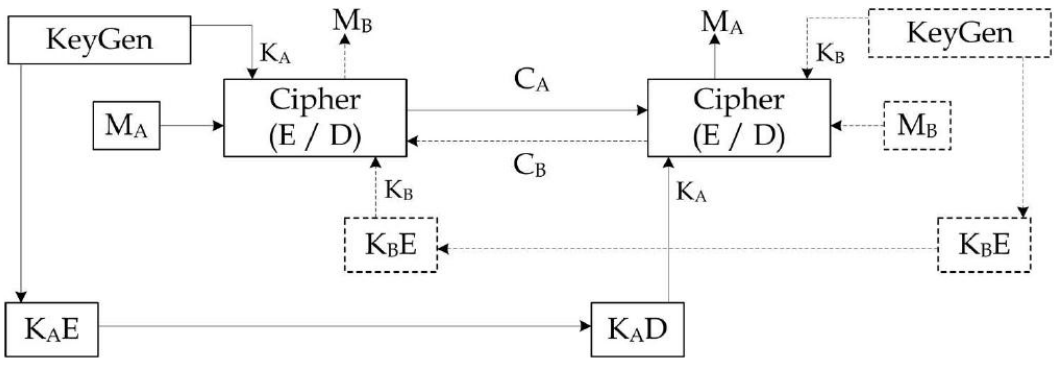


Figure 1.8 Crypto-systèmes pour Duplex transmission

Légende de la figure 1.8:

M_A - un message clair de l'utilisateur A

M_B - un message clair de l'utilisateur B

KeyGen - bloc de génération de clé secrète (K_A, K_B)

K_AE - Bloc de chiffrement à clé secrète, en utilisant un algorithme symétrique ou public pour l'utilisateur A

K_BE - Bloc de chiffrement à clé secrète, en utilisant un algorithme symétrique ou public pour l'utilisateur B

K_AD - Bloc de décryptage à clé secrète (au côté B), en utilisant le même algorithme que K_AE

K_BD - Bloc de décryptage à clé secrète (à la face A), en utilisant le même algorithme que K_BE

Cipher (E / D) - est un algorithme de chiffrement symétrique utilisé pour le chiffrement et le déchiffrement des messages M_A et M_B

La figure 1.9 montre un système de chiffrement pour le stockage des clés secrètes et des cryptogrammes. Le côté de cryptage est l'unité d'écriture. Après la génération de clé secrète (K) et son cryptage (bloc E *), il est placé sur le support de stockage. Après le cryptage des données: $E_K(M) = C$, le cryptogramme C est également placé sur le support de stockage. Le côté de décryptage est l'unité de lecture. Il décrypte la clé secrète (bloc D*) et l'utilise pour déchiffrer le cryptogramme: $M=D_K(C)$.

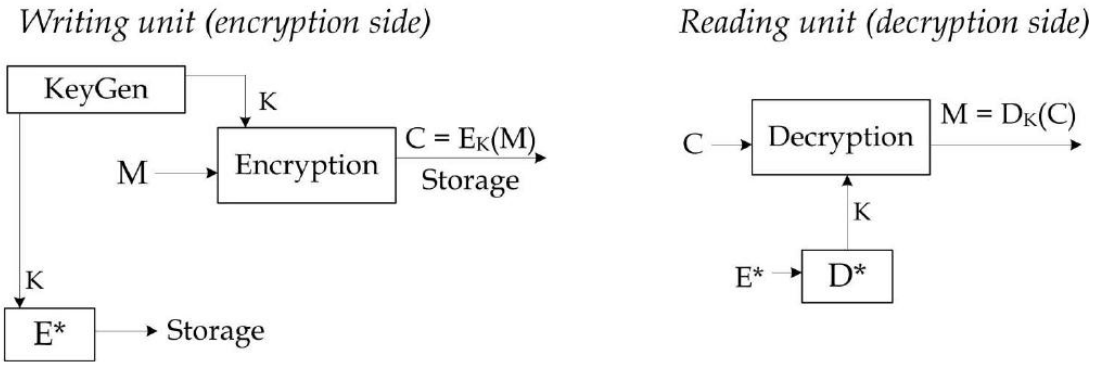


Figure 1.9 Crypto-system de stockage

I.9 Compression des données

Les techniques de compression de données sont basées sur l'élimination de certaine redondance (modèles) présents dans les données et l'élimination des détails fins dans les données multimédia qui ne sont pas important pour la perception humaine. La redondance peut être de différents types selon la nature des données. Les pixels voisins de l'image ont généralement des valeurs proches; dans toute langue, certaines lettres apparaissent plus souvent que les autres; il peut y avoir une certaine périodicité dans le signal audio. Les détails de haute fréquence dans les données multimédia qui ne sont pas importants pour la perception humaine peuvent être éliminés. Les algorithmes de compression exploitent ces aspects des données afin d'obtenir une taille compacte. **[Say03]**.

Le procédé de compression est nommé codage et la phase de reconstruction est nommée décodage. Le signal d'entrée pour le codeur peut être tout type de données comme: audio, vidéo, image signaux (Figure 1.10). Par l'encodage du format et la taille des données d'entrée d'origine, X est transformé et une représentation comprimée est obtenue soit X_C. Considérant que les données ne sont pas modifiées par transmission de canal, l'entrée du décodeur est X_C. Le décodeur de la source prend la représentation comprimée et reconstitue le signal original. Le signal reconstruit X_R est identique en compression sans perte et en compression avec perte, il est un rapprochement de X.

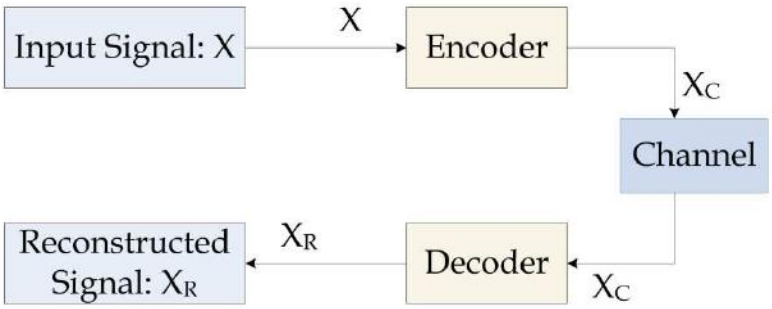


Figure 1.10 : Schéma général du processus de compression

Le but de la compression de données est de réduire la taille, le nombre de bits utilisé pour représenter une certaines informations. Le débit binaire est la mesure utilisée pour estimer le nombre de bits par valeur dans les données compressées. Dans le cas d'un fichier image, le débit binaire est mesuré en bits par pixel.

La section suivante décrit les étapes du processus de compression. Dans le sous-chapitre 1.6.2, les éléments de la théorie de la distorsion de taux sont présentés comme il est pertinent pour l'efficacité de la compression. Dans les prochains sous-chapitres 1.6.3 - 1.6.4 les différents types de compressions sont expliqués, suivi par la combinaison des méthodes de compression avec le chiffrement.

I.9.1 Chaîne de compression des données

Dans la figure 1.10, un schéma général de la compression est montré. Cette section présente les étapes et vue schématique des processus de codage et de décodage. Comme le montre la figure 1.11, le processus d'encodage est composé de 3 grandes opérations. La première consiste à transformer les données originales afin d'obtenir une représentation qui est plus commode pour le codeur. C'est fait en appliquant des transformations comme DCT[1] ou DWT[2] sur les données. La distribution des probabilités des coefficients obtenus peut être approchée par la fonction de Gauss[3] généralisée (GGF) ou la fonction de la probabilité de la densité Laplace[4] (PDF) **[PAR03]**. Cette nouvelle représentation des données est bien adaptée pour une bonne performance de codage.

L'opération suivante est la quantification des données. Il peut être inclus ou non dans le régime de compression selon le type de compression (sous-chapitre 1.6.3). La Quantification introduit la perte de l'information. Elle peut être utilisée pour éliminer des détails très fins, comme les informations de haute fréquence dans une image ou un fichier audio, sans perte pour notre système de perception visuelle et auditif. C'est un bloc de compression qui introduit un compromis entre la distorsion et le débit binaire (sous-chapitre 1.6.2). La dernière opération de codage est un codage entropique. Elle attribue des codes de longueurs différents à des valeurs de signal en fonction de leur probabilité (Figure 1.11). **[Ant11]**.

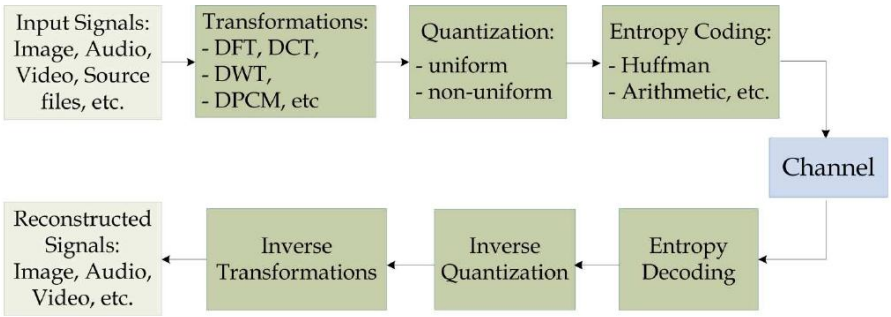


Figure 1.11 : Chaîne de compression : procédés de codage / décodage

[1] ***Discrete Cosine Transform :*** est une transformation proche de la transformée de Fourier discrète (DFT). Le noyau de projection est un cosinus et crée donc des coefficients réels, contrairement à la DFT, dont le noyau est une exponentielle complexe et qui crée donc des coefficients complexes. On peut cependant exprimer la DCT en fonction de la DFT, qui est alors appliquée sur le signal symétrisé. [Wikipedia]

[2] ***Discrete Wavelet Transform : (*ondelette)** est une fonction à la base de la décomposition en ondelettes, décomposition similaire à la transformée de Fourier à court terme, utilisée dans le traitement du signal. Elle correspond à l'idée intuitive d'une fonction correspondant à une petite oscillation, d'où son nom. [www.futura-sciences.com]

[3] **Johann Carl Friedrich Gauss :** né le 30 avril 1777 à Brunswick et mort le 23 février 1855 à Göttingen, est un mathématicien, astronome et physicien allemand. Il a apporté de très importantes contributions à ces trois domaines. Surnommé « le prince des mathématiciens », il est considéré comme l'un des plus grands mathématiciens de tous les temps. [Wikipedia]

[4] **Pierre-Simon Laplace :** né le 23 mars 1749 à Beaumont-en-Auge et mort le 5 mars 1827 à Paris, est un mathématicien, astronome, physicien et homme politique français. [Wikipedia]

a) Transformations:

Les méthodes de prévision comme DPCM (Differential Pulse Code Modulation) utilisent des corrélations entre des échantillons successifs. Par exemple dans le cas d'une combinaison des images voisines, les pixels déjà codés sont utilisée comme une prédiction pour le pixel courant. Dans JPEG sans perte, la différence entre la valeur de certain pixel et sa prédiction est codée par codage de Huffman[1]. Dans la compression avec perte, la différence quantifiée est codé. **[Ble10]**.

La transformation de fréquence comme DFT (Transformée de Fourier discrète) (Discrete Fourier Transform) et DCT (transformée en cosinus discrète) (Discrete Cosine Transform) sont utilisés pour représenter des données sous une forme plus efficace pour la compression. Les coefficients de haute fréquence représentent quelques détails sur les données multimédia qui sont moins perceptive pour les systèmes visuels et auditifs humains. Cette caractéristique est exploitée dans la compression avec perte pour les fichiers images, audio et vidéo. La représentation des fichiers dans le domaine de transformation de fréquence facilite l'extraction et l'élimination des coefficients correspondant aux hautes fréquences. Par cette transformation les données spatiales fortement corrélé sont transformés en coefficients non corrélées. Dans transformation DCT, les coefficients les plus importants pour la perceptibilité sont dans le coin haut gauche de l'image transformée (les coefficients de bas niveau).

Une Décomposition comme DWT (Discrete Wavelet Transform) (Discrete Wavelet Transform) est largement utilisé dans la compression d'image. Décomposition en ondelettes est une application successive des filtres passe haut et bas qui conduit à sous-bandes de détails haute fréquence dans des directions diagonales, horizontales et verticales, respectivement, et l'approximation de l'image des basses fréquences restante. Les coefficients correspondant aux hautes fréquences sont presque tous proches de zéro et que quelques-uns d'entre eux ont des valeurs plus élevées. En utilisant un seuil, tous les coefficients proches de zéro peuvent être mis à zéro et le reste des coefficients de haute fréquence au-dessus du seuil peut être codé.

[1] **Le professeur David Albert Huffman** (9 août 1925 - 7 octobre 1999) fut un pionnier dans le domaine de l'informatique. Il est principalement connu pour l'invention du codage de Huffman utilisé dans presque toutes les applications qui impliquent la compression et la transmission de données digitales comme les fax, les modems, les réseaux informatiques et la télévision à haute définition. [www.futura-sciences.com]

b) Quantification:

La quantification est la méthode d'approximation utilisée dans la compression avec perte. Le but de la quantification est d'obtenir un débit plus petit par la représentation de chaque groupe de valeurs proches par une seule valeur. Cette cartographie est irréversible et au décodage d'un groupe d'origines différentes, des valeurs proches ont la même valeur. L'échelle d'approximation est mesurée par l'étape de quantification.

Le procédé de quantification peut être réalisé par un quantificateur scalaire ou vecteur selon la représentation des données qui peut être un ensemble de scalaires ou d'un ensemble de vecteurs. Une quantification scalaire réalise une correspondance entre les valeurs de données et une gamme de valeurs de codage possibles. Par exemple, en considérant une source distribué uniformément dans l'intervalle [-A,A] et M mots de code possibles pour le codage, l'étape de quantification (Q_s) est 2A/M. L'intervalle de la source d'origine [-A,A] est divisé en intervalles uniformes en largeur Q_s. La valeur moyenne de chaque intervalle est utilisée pour représenter toute la gamme des valeurs de la source de cet intervalle. Une source d'un groupe de quantification vectoriel mène dans des vecteurs de valeurs fermé, puis de les encoder en trouvant le code-vecteur le plus proche. Le décodeur de quantification vectorielle a une table de consultation pour la reconstruction des vecteurs. **[GG91]**.

c) Codage entropique:

Le Codage d'entropie est composé à partir d'un modèle de données et un codeur. Un modèle de données est une carte de probabilité pour chacun de ses éléments; leur distribution de probabilité. Le codeur utilise cette distribution de probabilité et crée les codes. Un code plus long correspondant au symbole moins probable et un code court correspondant au symbole le plus probable. Codage d'entropie est sans préfixe, ce qui signifie qu'aucun des mots de code n'est un préfixe aux autres mots de code. Cette propriété sans préfixe est nécessaire pour le décodage correct du bitstream composée de mots de code de longueur variable. La technique Huffman et le codage arithmétique entropique sont les algorithmes les plus couramment utilisés.

I.9.2 Rate-Distortion Théorie

La théorie de la distorsion du taux fait partie de la théorie de l'information et c'est une mesure de la performance de la compression. Compte tenu d'un certain niveau de qualité de données , il est souhaitable d'obtenir le plus petit nombre de bits par valeur, ce qui signifie le meilleur bit-rate possible. **[Boc09]**.

La distorsion (D) des données est mesuré en tant que différence par rapport à ses valeurs initiales {xi} et des valeurs approximarives obtenues après compression, à la reconstruction $\{\hat{x}i\}$. Il est généralement mesurée par erreur quadratique moyenne:

$$D(x,\hat{x}) = \frac{1}{n}\sum_{i=1}^{n}(x-\hat{x})^2$$

Le bit-rate (R) est le nombre moyen de bits par symbole dans les données. Considérant qu'il existe n symboles (S) dans les données, le débit binaire peut être exprimée comme suit:

$$R = \frac{1}{n}\sum_{i=1}^{n} long(S)$$

Compte tenu d'une source et d'une mesure de distorsion, selon la théorie de la distorsion des taux, il ya un taux (R) - distorsion (D) - fonction R(D) qui a généralement une forme comme dans la figure 1.12). **[GG91]**.

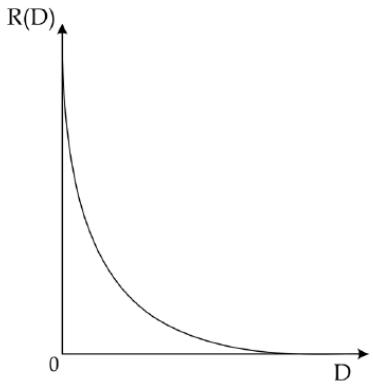


Figure 1.12 : Fonction de distorsion du taux

Le bloc de quantification est une partie de la compression qui introduit une distorsion, mais il est aussi celui qui permet de réduire le bit-rate. Différentes méthodes de quantification peuvent être comparées grâce à la fonction R(D). La fonction R(D) spécifie le meilleur compromis qui peut être obtenu au bit-rate le plus bas possible et la distorsion du signal minimal.

I.9.3 Type de compression de donnée :

Il existe deux types de compression: avec et sans perte. L'objectif de la compression sans perte est d'assurer la fidélité de données. Dans ce cas, les données décompressées correspondent exactement à l'original. Seulement la redondance statistique est utilisée pour la compression. **[RJ91]**. La compression sans perte est appliquée à des données lorsque le rapprochement est inacceptable, comme: des documents texte, des programmes exécutables, le code source. La quantité de compression est limitée à l'entropie de la source. Le procédé le plus connu utilisé dans la compression des données sans perte est Lempel-Ziv[1], inventé en 1977 et nommé par leurs auteurs. C'est utilisé dans des formats d'image comme GIF et PNG. **[ZL78]**

[1] (**LZ77** et **LZ78)** sont deux algorithmes de compression de données sans perte proposés par Abraham Lempel et Jacob Ziv en 1977 et 1978. Ces deux algorithmes posent les bases de la plupart des algorithmes de compression par dictionnaire [Wikipedia].

Le but de la compression avec perte est d'assurer le meilleur compromis entre la qualité des données et son efficacité dans le stockage. Les données décompressées sont une approximation de l'information originale. Les techniques courantes qui introduisent une perte d'informations sont la quantification et l'arrondissement des nombres. Une fois une certaine valeur a été arrondie ou représenté par l'indice de niveau de quantification, elle ne peut pas être récupérée. **[Rum09]**.

La compression avec perte d'image et de vidéo la plus connu et largement utilisé sont JPEG, MPEG, H264, et HEVC. Une plus récente norme de compression d'image est JPEG2000, elle a une meilleure performance de compression, puis JPEG, mais il n'a pas encore été couramment utilisé sur l'Internet.

I.9.4 Classification des méthodes de compressions / chiffrement multimédia

Durant le chiffrement, les données passe à travers une série de transpositions et de substitution. Si la sécurité de l'algorithme est forte, la redondance du texte en clair ne sera pas transférée sur le texte chiffré. Si la redondance des données est élevée, comme dans des images, fichiers audio, vidéo, il y aura une forte probabilité que les fichiers cryptés garderont une partie de la structure du texte clair. Ceci est une des raisons pour lesquelles la compression de la donnée multimédia est appliquée en premier et ensuite le chiffrement. Ce modèle a été proposé dans un système de cryptographie fort, hybride nommée « PGP - Pretty Good Privacy ». **[PGP04]**

Le processus de chiffrement n'est jamais appliqué avant la compression en raison des questions pratiques. Le processus de chiffrement rend aléatoire les données d'origine qui tentent d'atteindre une probabilité égale d'apparition de données; par conséquent, il ne restera pas une information qui peut être compressée.

Une méthode classique consiste à effectuer la compression des données, puis d'effectuer le chiffrement de l'ensemble du bitstream. Ce processus est appelé chiffrement complet ou directe (Figure 1.13.); il est gourmand en temps et en espace et donc peut parfois ne pas être approprié pour les applications en temps réel. Le chiffrement complet est utilisé quand un haut niveau de sécurité est nécessaire et surtout pour le stockage. Les données multimédias sont généralement impliquées dans les interactions en temps réel où la transmission doit être rapide, et ils sont déjà volumineux sans cryptage, ce qui peut augmenter la taille des données. Afin de résoudre ce problème de sécurité, le chiffrement partiel ou sélectif a été proposé. **[CL00]**. L'idée du chiffrement sélectif (SE) est à chiffrer une partie seulement des données compressés (Figure 1.13). De cette manière, le volume de données est réduit et la vitesse de transmission augmente.

Un autre procédé pour éviter le coût de calcul et de stockage du au cryptage est de l'intégrer à l'intérieur du codage entropique. Cette méthode est un chiffrement combinée compression également nommé conjointe compression-chiffrement. Dans ce cas, la sécurité est intégrée à l'intérieur du processus de compression. **[SK05]**

Le chiffrement de donnée multimédia complet signifie que l'ensemble du bitstream codé est chiffrée sans tenir compte du format des données. Ce qui peut changer est le cryptogramme choisi pour la protection des données. Les algorithmes de chiffrement symétriques connus les plus utilisés sont «DES Data Encryption Standard» **[FIPS93]**, «AES - Advanced Encryption Standard» **[FIPS01]**, «IDEA - International Data Encryption Algorithm» **[LM91]**, etc.

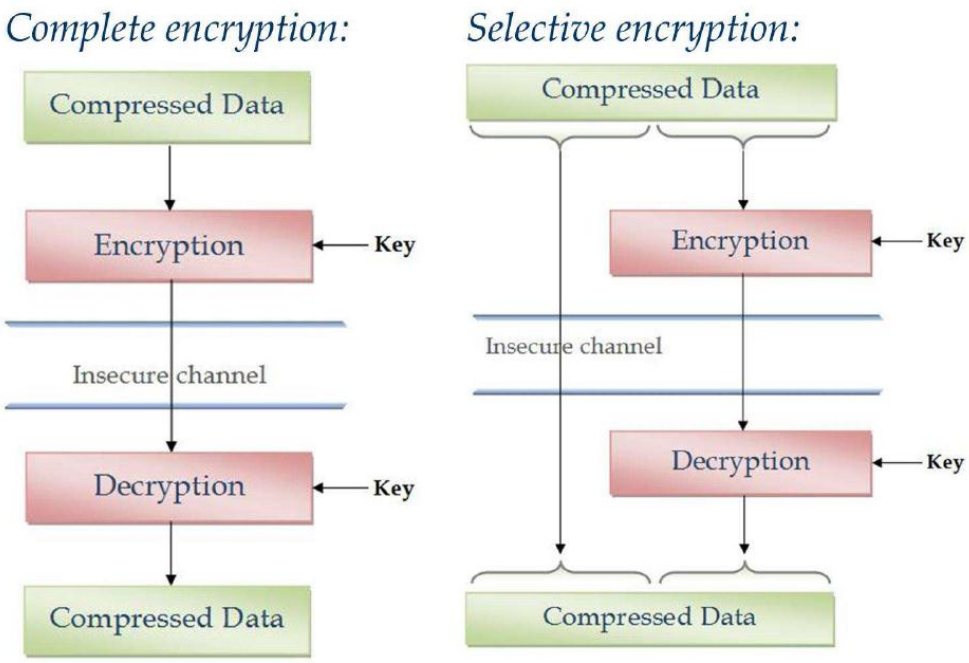


Figure 1.13 : Schéma de chiffrement complet / partiel

Le chiffrement sélectif peut être réalisé de différentes manières; il dépend de la partie des données sélectionnées pour le chiffrement. Une des techniques SE est basée sur la sélection de quelques coefficients DCT qui sont les plus importants pour la perception humaine des données. DCT (Discrete Cosine Transform) représente des données dans le domaine fréquentiel transformé et est utilisé dans la compression pour éliminer les coefficients de haute fréquence qui ne sont pas perceptibles par le système auditif et visuel humain. Cette transformation fait partie des normes de compression JPEG et MPEG. **[SB98]**.

I.10. Conclusion du chapitre

Dans ce premier chapitre nous avons présenté les aspects fondamentaux de la cryptologie, avec ses deux disciplines : la cryptographie et la cryptanalyse. Dans un premier temps, nous avons définit quelques notions sur la cryptographie, ainsi que ses principaux objectifs.

Ensuite, nous avons abordé la cryptographie avec ces deux types : la cryptographie symétrique et asymétrique, tout en expliquant leur fonctionnement à travers des exemples d'algorithmes cryptographiques anciens et modernes. Enfin, nous avons présenté la compression et son rôle dans la cryptographie.

Récemment, un nouvel axe de recherche en cryptographie a vu le jour. Il utilise l'ADN pour la résolution de problèmes. Cet axe qui est la cryptographie à l'ADN sera introduit dans le deuxième chapitre. Mais avant, nous nous voyant dans l'obligation de présenter l'ADN à travers quelques notions biologique.

Contenu en bref

II.1 Introduction

II.2 L'ADN

II.3 La cryptographie au niveau moléculaire

II.4 Les aspects d'évaluation de performance

II.5 Conclusion du chapitre

II.1. Introduction :

La cryptographie ADN est une nouvelle branche scientifique large, qui comprend une variété de domaines scientifiques. La sécurité de l'information (cryptographie, stéganographie, gestion de clés), la biologie moléculaire, la bioinformatique, le calcul biomoléculaire. C'est un nouveau et prometteur domaine de la sécurité de l'information. Elle combine les solutions classiques en cryptographie avec la résistance du matériau génétique. L'ADN biologique peut être utilisé dans la stéganographie et cryptographie comme matériau de stockage. Le calcul moléculaire peut être effectué avec les structures d'ADN biologiques et ensuite appliqué sur les chiffres classiques. Plusieurs projets de séquençage de génome offrent la possibilité d'exploiter les bases de données d'ADN numériques pour des fins cryptographique.

Main avant d'entamer le sujet il est primordial de définir la molécule d'ADN d'un côté purement biologique en parcourant ses structures physique et chimique ainsi que ses caractéristiques. Ensuite les différentes techniques de la biologie moléculaire qui permettent la manipulation de l'ADN et le calcul à l'ADN.

II.2 L'ADN :

II.2.1 Comprendre L'ADN :

Au sein de nos cellules se trouve l'ensemble de nos gènes (le génome), contenus dans une grande molécule en forme de double hélice, l'ADN ou Acide DésoxyriboNucléique. L'ADN contient toutes les informations permettant à l'organisme de vivre et de se développer ; il est le support de notre information génétique, mais également celui de l'hérédité (Fig 2.1). Une molécule a priori relativement simple peut commander à elle seule tout le fonctionnement d'un organisme.

Si nous connaissons maintenant tout le génome humain depuis son séquençage en 2003, nous ne connaissons néanmoins les fonctions que de 10% de nos gènes. Il reste à découvrir quelles informations renferment les autres. Pour avancer dans la connaissance du génome, une méthode consiste à cibler et à agir sur ces gènes pour qu'ils ne s'expriment plus normalement, c'est de cette manière que l'on peut parvenir à identifier leurs fonctions respectives. **[MSTM06]**.

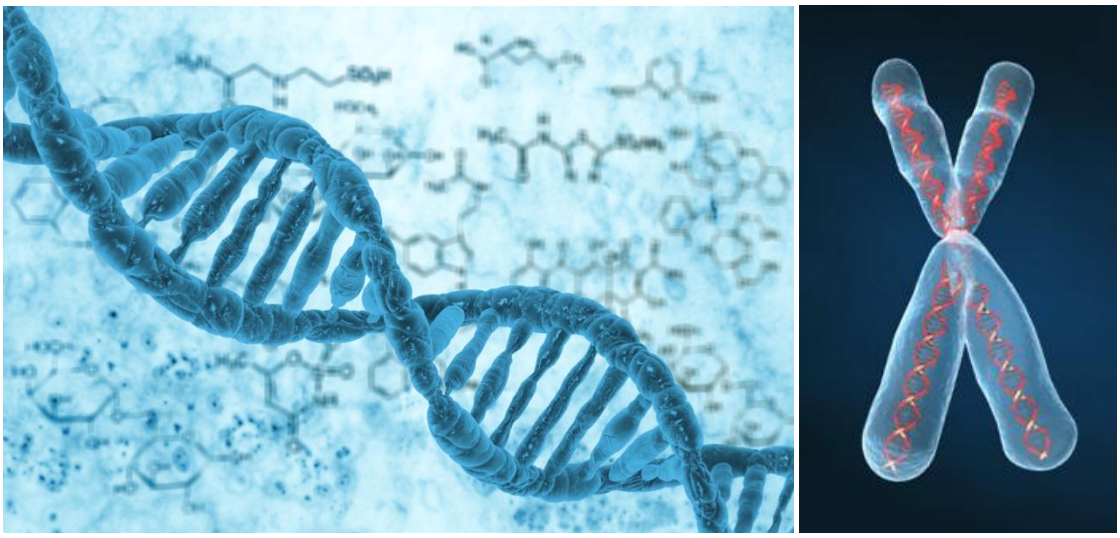


Figure 2.1 : ADN : vue globale.

II.2.2 Un peu d'histoire sur l'ADN [GG10] :

- **1865** : Johann Gregor Mendel établit les bases de l'hérédité en définissant la manière dont les gènes se transmettent de génération en génération : ce sont les lois de Mendel.

- **1869** : Johann Friedrich Miescher découvre dans le noyau des cellules vivantes une substance riche en phosphate – la nucléine –, qui sera nommée au XXe siècle «ADN» ou Acide DésoxyriboNucléique.

- **1882** : Walther Flemming met en évidence les chromosomes, constitués de molécules d'ADN, qui regroupent plusieurs gènes. Il décrit pour la première fois la

mitose, phénomène par lequel les cellules se divisent et permettent la croissance et le renouvellement cellulaires.

- **1928** : Phoebus Levene puis Erwin Chargaff déterminent la structure chimique de l'ADN avec sa composition en bases azotées : adénine A, thymine T, guanine G et cytosine C.

- **1944** : Oswald Avery établit que l'ADN est le transporteur de l'information génétique.

- **1952** : James Watson et Francis Crick établissent la structure en double hélice de l'ADN, ce qui leur valut le prix Nobel de physiologie et de médecine en 1962.

- **En 1995**, les chercheurs ont pu « lire » pour la première fois tout l'ADN contenu dans le génome d'un organisme unicellulaire. Depuis cette date, des pas de géant ont été franchis en génétique, et les chercheurs ont ainsi séquencé des génomes de plus en plus longs, aboutissant en avril 2003 au séquençage complet du génome humain.

II.2.3 Structure de l'ADN :

La découverte de la structure de l'ADN bouleverse l'étude des phénomènes biologiques en introduisant la dimension moléculaire. Proposée par Watson et Crick en 1953, elle est obtenue non seulement à partir de l'interprétation de clichés de diffraction des rayons X réalisées par Franklin, mais aussi des travaux d'Erwin Chargaff qui avaient montré que pour toute molécule d'ADN, le nombre de molécules d'adénine est égal au nombre de molécules de thymine, et que celui de cytosine est égal à celui de guanine et enfin l'analyses en microscopie électronique, qui avaient montré que le diamètre de la molécule d'ADN est de 20micron , ce qui suggérait que cette molécule comportait deux chaînes de désoxyribose-phosphate. Les deux points fondamentaux de la structure sont les suivants **[GG10]**:

- Les bases ou nucléotides (A,T,C,G) s'organisent en pairs A-T et G-C
- Cet appariement permet un enroulement quasi-parfait en hélice droite des deux chaînes sucre-phosphate qui portent ces nucléotides.

La structure est stabilisée par l'interaction (liaisons hydrogène) entre les bases et l'empilement successif des paires de nucléotides le long de la double-hélice.

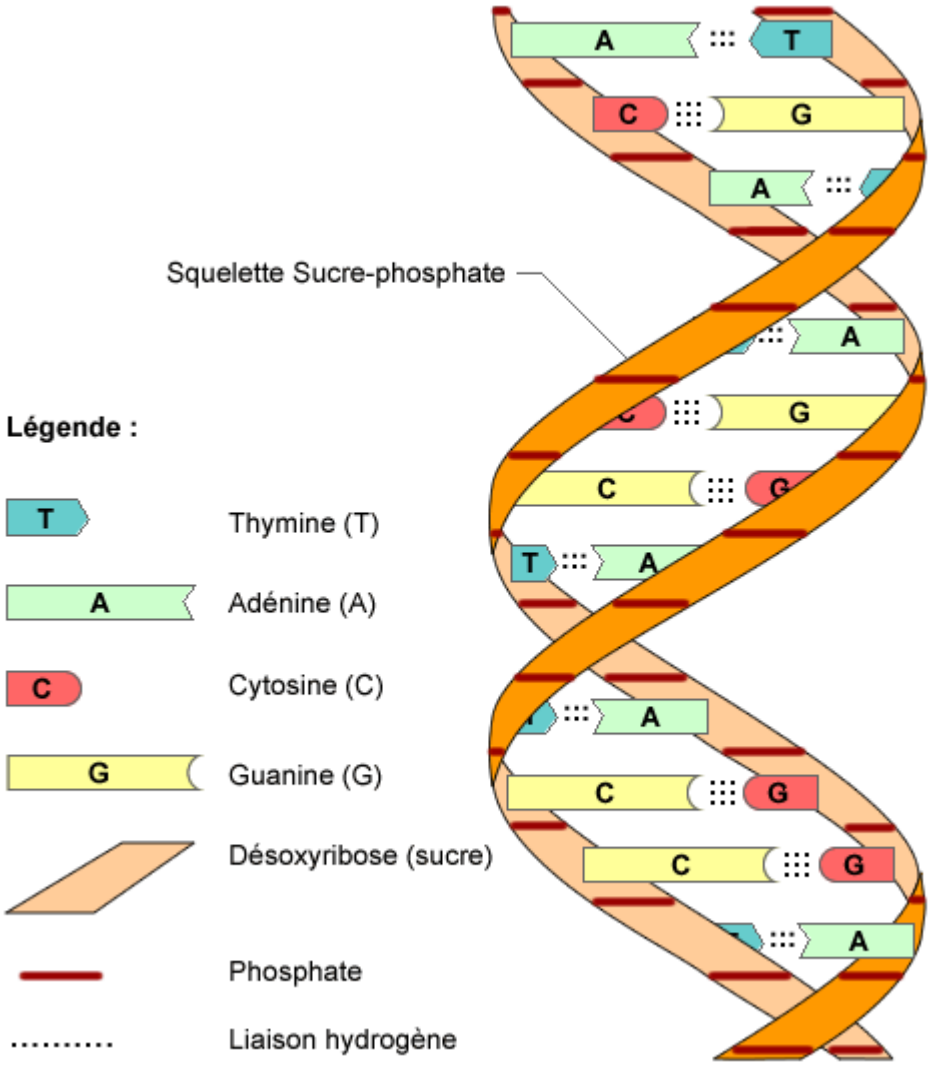


Figure 2.2 : Structure de l'ADN

II.2.4. Définition de quelles que notions :

II.2.4.1. Un Chromosome :

L'information génétique (ADN) qui commande la fonctionnalité de la cellule est divisée en chromosomes. Chaque chromosome est constitué d'une molécule unique d'ADN qui porte des gènes (Fig. 2.2).

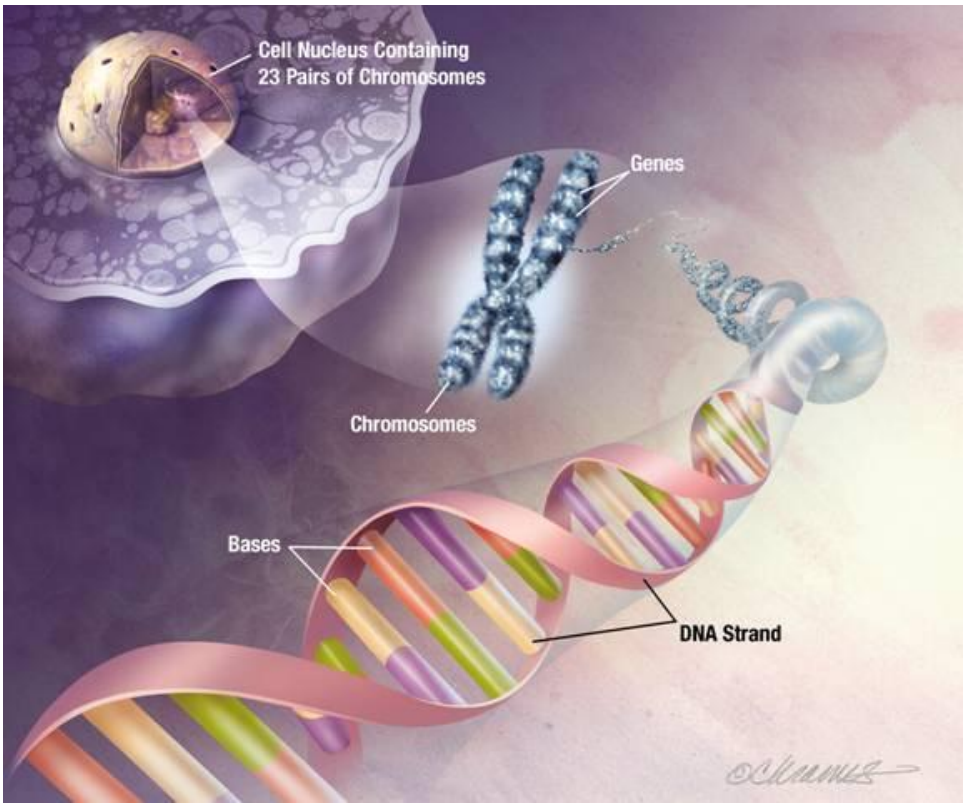


Figure 2.3 Structure d'un chromosome

II.2.4.2. Les bases azotées

Ce sont les quatre bases azotées qui assurent la variabilité de la molécule d'ADN, ainsi que la complémentarité des deux brins. En effet, il n'existe que deux types complémentaires de bases : une pyrimidine sera toujours en face d'une purine. La thymine (T) et la cytosine (C) sont de la famille des pyrimidines. L'adénine (A) et la guanine (G) sont de la famille des purines. Les bases azotées sont complémentaires deux à deux, une purine s'associant toujours à une pyrimidine: l'adénine (A) s'associant avec la thymine et la guanine (G) avec la cytosine. Les bases azotées complémentaires sont reliées entre-elles par des liaisons hydrogène. **[JCP99]**

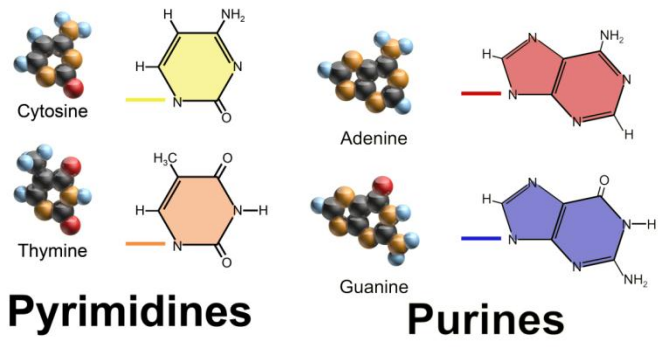


Figure 2.4 : Structure chimique des bases azotées

II.2.4.3. Le nucléotide :

Un nucléotide est l'ensemble d'une base azotée, d'un sucre et d'un groupement de phosphate. Le sucre présent dans l'ADN est le *Désoxyribose*, ce sucre est relié à l'une des bases azotée (A, T, C, G). **[GG10]**

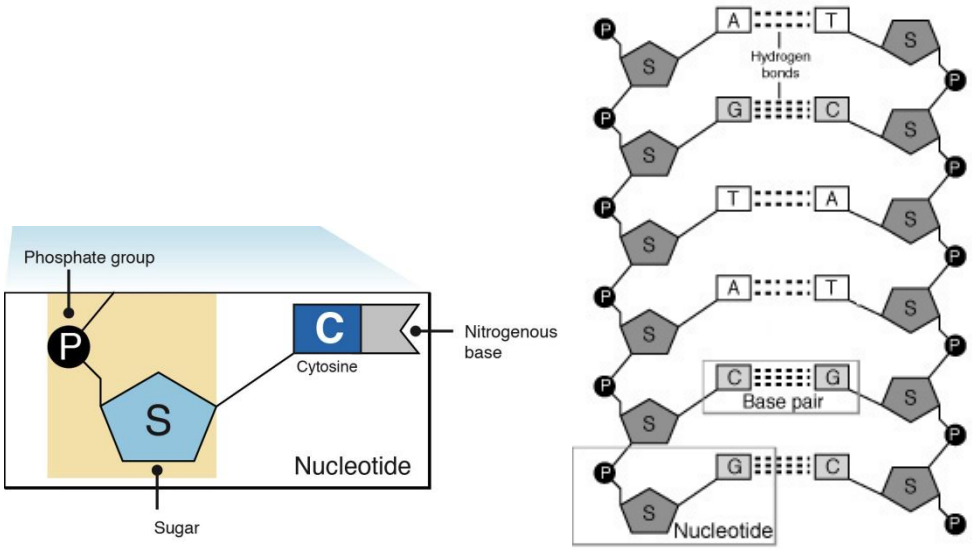


Figure 2.5 Structure d'un nucléotide

II.2.4.4. Le nucléoside :

Un nucléoside est l'ensemble d'une base azotée, d'un sucre. En d'autre termes, c'est un nucléotide sans groupe de phosphate. **[GG10]**

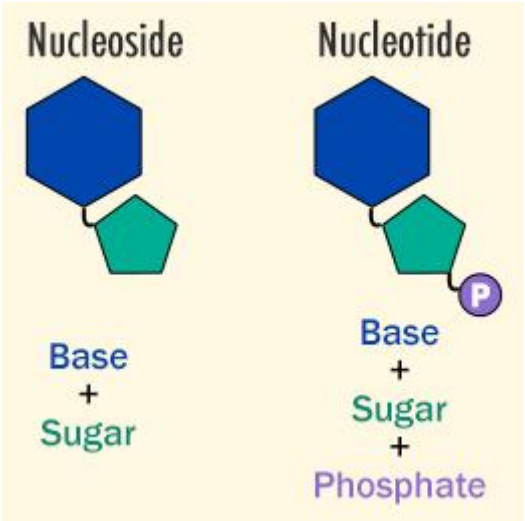


Figure 2.6 : Nucléoside VS Nucléotide

II.2.4.5. Le brin d'ADN :

Les nucléotides s'enchaînent avec la formation de liaisons entre le phosphate d'un nucléotide et le désoxiribose (sucre) porté par le carbone 3' du nucléotide sous-jacent. On a ainsi un brin d'ADN dont on lit la séquence dans la direction 5' à 3'. **[GG10]**

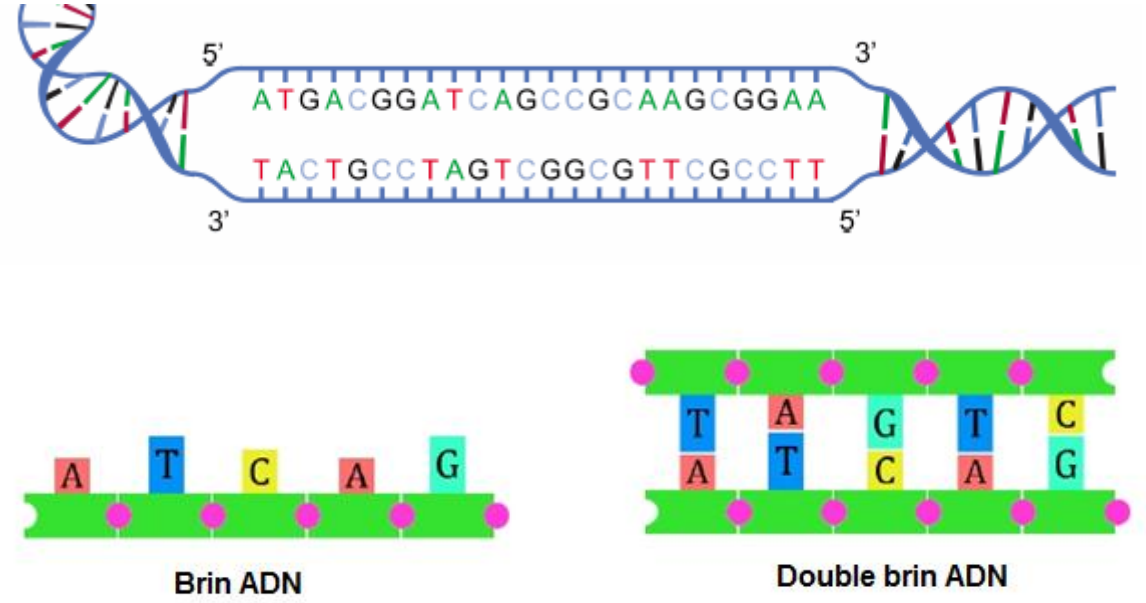


Figure 2.7 Structure d'un brin d'ADN

II.2.4.6. Qu'est-ce qu'un gène?

Un gène est une portion d'ADN qui contient toute la recette d'assemblage d'une protéine. Ce sont les protéines qui font l'essentiel du travail dans l'organisme et les gènes permettent leur reproduction de génération en génération.

Concrètement, un gène est une longue séquence de A, C, G et T dans l'ADN. Une partie de cette longue séquence contient les « plans de construction » de la protéine en langage génétique. (Chaque combinaison de trois lettres génétiques correspond à un acide aminé particulier). Le reste de la séquence d'un gène indique à la cellule dans quelle circonstance elle doit exécuter les instructions de ce gène. Par exemple, l'insuline est fabriquée dans le pancréas. Le gène de l'insuline, contient donc une séquence spécifique de A, C, G et T qui attire la « tête de lecture » du pancréas pour enclencher le décodage du gène. Même si le même gène de l'insuline est aussi présent dans les cellules des autres organes, ceux-ci n'ont pas la bonne tête de lecture et ne peuvent donc pas l'utiliser. **[MSTM06]**

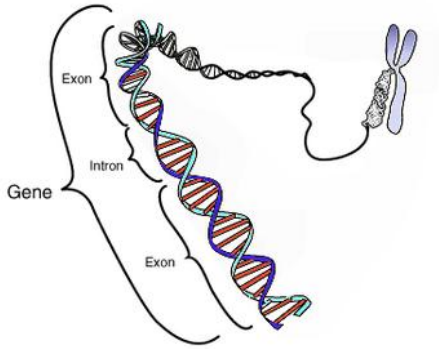


Figure 2.8 Le gène.

II.2.4.7. Le génome :

Le génome est l'ensemble du matériel génétique d'un individu ou d'une espèce codée dans son acide désoxyribonucléique (ADN). **[GG10]**

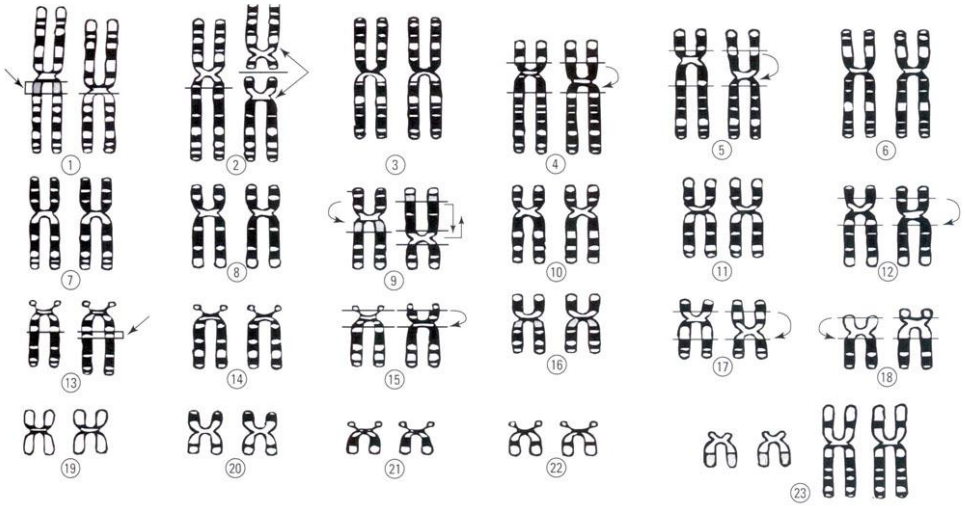


Figure 2.9 Génome : Homme à gauche / Chimpanzé à droite[1]

II.2.4.8. Le codon :

Un codon est une séquence de trois nucléotides spécifiant l'un des 22 acides aminés protéinogènes. **[GG10]**

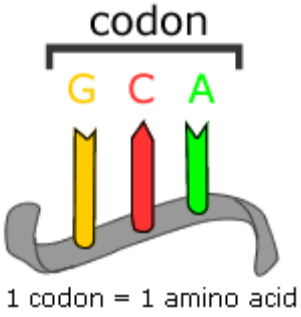


Figure 2.10 Un Codon

[1] http://www.futura-sciences.com

II.3 Le calcul à l'ADN :

Nous assistons de nos jours à des avancées remarquables en matière de miniaturisation. Les composants électroniques sont de plus en plus petits et de plus en plus puissants. Mais la miniaturisation a des barrières physiques évidentes et tôt tard, l'homme atteindre l'échelle de la molécule.

II.3.1. Les ordinateurs moléculaires :

L'idée des ordinateurs moléculaires a pris naissance dans les années 50 lorsque Richard Feynman, lauréat d'un prix Nobel, a exposé le concept du calcul à un niveau moléculaire**. [ANT03]**

Les recherches sur l'ordinateur à base d'ADN ont un grand succès de point de vue théorique. En effet, des mathématiciens ont démontré que l'ADN pouvait trouver la solution de n'importe quel problème résoluble par une machine de Turing. En termes plus clairs, ceci veut dire que l'ordinateur à l'ADN serait capable (au moins en théorie) de résoudre n'importe quel problème exprimable par une fonction récursive**. [ANT03]** La puissance des ordinateurs moléculaires réside dans la puissance de l'ADN et des outils de la biologie moléculaire. En effet, l'ADN permet un stockage d'information et un calcul parallèle très importants, alors que les outils de la biologie moléculaire permettent de manipuler (créer, supprimer, modifier, insérer…) les chaînes de bases ce qui permet de réaliser des opérations diverses. Un ordinateur à l'ADN peut être 1 200 000 fois plus rapide que nos ordinateurs **[ANT03].** C'est le cas de l'ordinateur moléculaire conçu en 2002 par une équipe de l'université Israélienne Weismann Institute, et qui a les caractéristiques suivantes :

- Il tient dans une éprouvette.
- Très faible consommation d'énergie (de l'ordre du millionième de watt).
- Une grande capacité de traitement en parallèle.
- Des réactions chimiques permettent les différents calculs. **[ANT03]**

La première concrétisation des ordinateurs moléculaires remonte à 1994, lorsque Leonard Adleman, chercheur au laboratoire de science moléculaire du département de science informatique de Californie du Sud, réussit à faire le mariage entre l'informatique et la biologie. En effet, il proposa une solution au problème hamiltonien en créant le premier ordinateur moléculaire.

II.3.2. L'expérience d'Adleman

En 1994, Adleman a résolu le problème du chemin hamiltonien dans un graphe orienté de 7 sommets en utilisant les techniques de la biologie moléculaire et le calcul à l'ADN. Profitant du parallélisme de l'ADN, il a pu générer tous les chemins possibles entre les sommets, puis en procédant par élimination, il a aboutit à la solution.

II.3.2.1. Le problème du chemin hamiltonien :

Soit un graphe G possédant un sommet de départ SIn (Départ) et un sommet d'arrivée SOut (Arrivée), On dit que G est un graphe qui possède un chemin hamiltonien si et seulement s'il existe un chemin qui commence par SIn et passe par tous les autres sommets une seule fois et se termine par SOut. **[ADL94]**

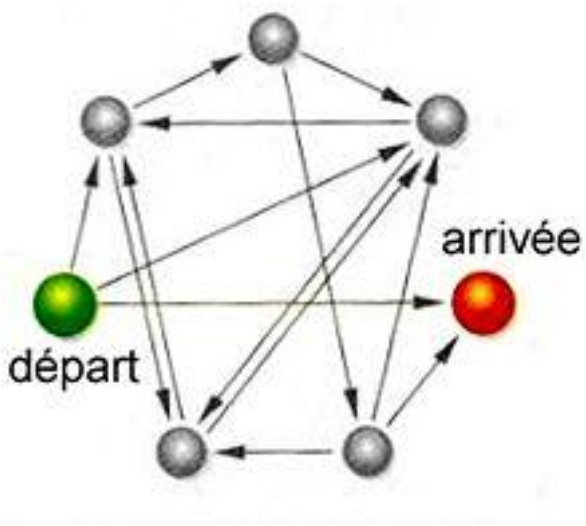


Figure 2.11 : Le problème hamiltonien

La difficulté du problème n'est pas de trouver un algorithme qui trouve un chemin hamiltonien (s'il existe) dans un graphe orienté, mais de trouver un algorithme qui donne ce chemin en un temps raisonnable. En d'autres termes, tous les algorithmes qui existent croient de façon exponentielle au fur et à mesure que le nombre de sommets augmente. A ce sujet Adleman a dit : « On connaît des graphes de moins de 100 sommets pour lesquels, même en utilisant le meilleur algorithme et le meilleur ordinateur, la résolution du problème du chemin hamiltonien prendrait des centaines d'années. ». Et il a ajouté : « Pour cette première expérience, j'ai cherché un problème suffisamment simple pour qu'un prototype d'ordinateur à ADN le résous, mais suffisamment important pour constituer une preuve indubitable de l'intérêt de ces nouvelles machines. » **[ADL98]**

II.3.2.2. Les étapes de l'algorithme de HPP (Hamiltonian path problem) [ADL94]

- Etape1: Générer un ensemble de chemins aléatoires (tous les chemins possible).
- Etape2: Garder seulement les chemins qui commencent et se terminent respectivement par le sommet de départ S_{in} et celui d'arrivée S_{out}.
- Etape3: Si le graphe possède **n** sommets, garder seulement les chemins qui ont exactement **n** sommets.
- Etape4: Garder seulement les chemins qui possèdent tous les sommets du graphe.
- Etape5: S'il reste un chemin, celui-ci est une solution.

Dans chaque étape, Adleman a utilisé plusieurs techniques et outils de la biologie moléculaire décrit précédemment. Voici une brève description de ces étapes :
La première étape : Pour représenter le graphe (sommets et arrêtes), Adleman a utilisé les conventions suivantes :

Chaque sommet est représenté par un brin d'ADN (simple brin) de 20 bases générées aléatoirement. Par exemple on peut représenter les trois sommets S2, S3, S4:

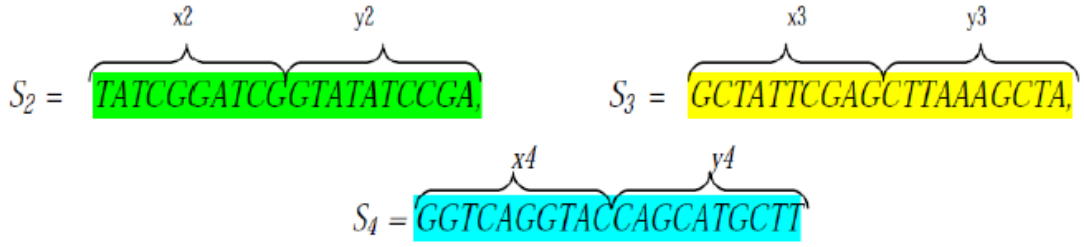


Figure 2.12: S ⇨ brin d'ADN.

Chaque arrête de sommet i vers le sommet j est représentée par un brin d'ADN de 20 bases résultant de la concaténation des 10 derniers bases du sommet i avec les 10 premiers bases du sommet j. Voila la représentation de l'arrête 2⇨3 et 3⇨4 (de l'exemple précédant) :

Figure 2.13 : la représentation de l'arrête 2⇨3 et 3⇨4.

Et puisque l'ADN est un polymère dirigé, avec un coté 5' et un coté 3', le lien i⇨j et le lien j⇨i sont représentés différemment. (Cette présentation préserve l'orientation des liens). La séquence Watson-Crick de 20 bases complémentaires à **Si** est dénotée **Si**

Une arrête qui commence par le sommet de départ **Sin** est représentée par un brin de 30-bases, dont les 20 bases coté 3' (à gauche) sont les 20 bases de Sin, les dix autres restantes sont les 10 bases premiers du sommet destinataire. De même, pour une arrête qui se termine par le sommet d'arriver **Sout**, elle est représentée par un 30 bases dont les 20 bases cotés 5' (à droite) sont les bases de **Sout** et le reste (10 bases) sont les 10 dernier bases de l'autre sommet (sommet i).

Tous les brins (sommet **Si**) et les brins complémentaire (sommet) sont amplifiés (créer plusieurs copies pour chaque brin), et laissés s'hybrider puis ajouter les enzymes de ligase pour souder les liens entre les brins (sommet). Donc la réaction de légation produit la formation de brins d'ADN formant les chemins aléatoires du graphe. Notez également que seuls les chemins qui commencent par le sommet de départ et se terminent par le

sommet d'arriver sont entièrement doubles brins. Les autres auront toujours, d'un coté ou de l'autre, une séquence de 10 bases simple brin.

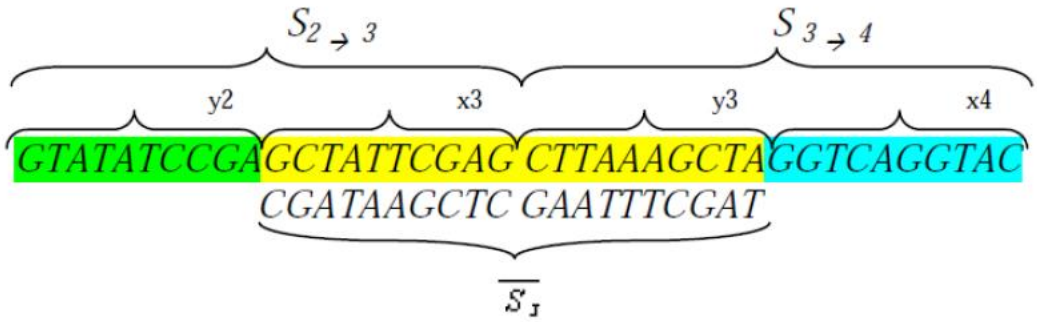


Figure 2.14: Les brins soudés avec des ligases.

La deuxième étape : Pour ne sélectionner que les chemins qui commencent par Sin et se terminent par Sout, il suffit d'appliquer la technique d'amplification des gènes, en ajoutant au tube un brin complémentaire au Sin et un autre complémentaire au Sout , avec cette technique, seuls ces chemins seront amplifiés.

La troisième étape : La sélection des chemins qui ont exactement **n** sommets, se fait par une simple électrophorèse : cet outil permet essentiellement de distinguer les longueurs.

La quatrième étape : Pour sélectionner, dans tous les chemins qui nous restent, ceux qui contiennent le sommet Si, on greffe des simples brins Si sur des billes micrométriques. On ajoute l'ensemble des chemins dont on dispose ; on chauffe pour séparer les doubles brins ; on laisse refroidir pour permettre aux brins complémentaires de s'hybrider à nouveau. Certains des chemins qui contiennent le sommet Si vont alors s'hybrider avec les brins Si sur les billes. On lave ensuite la solution pour enlever tous les ADN qui ne se sont pas hybridés. Finalement, on chauffe à nouveau la solution pour séparer les doubles brins et on récolte l'ADN en solution. Ces ADN contiennent sûrement la séquence du Si. Et on continue de la même manière pour tester la présence des autres sommets dans les chemins restants.

La cinquième étape : Pour vérifier s'il reste encore des brins d'ADN (les solutions), il suffit d'utiliser un gel qui détecte leur présence.

II.4. La cryptographie au niveau moléculaire

Cette partie décrit les opérations qui peuvent être effectuées avec des séquences d'ADN dans un laboratoire. Compte tenu des techniques disponibles pour manipuler des molécules d'ADN, les possibilités de stockage et des calculs au niveau moléculaire sont présentés au dessus.

II.4.1. Les motivations

Plusieurs motivations ont été à l'origine de l'apparition de la cryptographie à l'ADN, et ont permis à ce domaine de se propager dans plusieurs centres de recherche scientifique. Parmi ces motivations, on trouve :

II.4.1.1. Les limites de l'utilisation du silicium

En effet, actuellement plusieurs problèmes causés par les circuits à base de silicium sont apparus, comme par exemple, la distance entre les transistors qui s'approche de son seuil minimal, et le problème du bruit électromagnétique qui perturbe le bon fonctionnement de certains équipements électroniques **[LAM02]**

II.4.1.2. La capacité importante du stockage et le parallélisme de l'ADN

Ont été les premières motivations de l'apparition de la cryptographie à l'ADN. En effet, on a vu dans le chapitre précédent qu'un ordinateur à base d'ADN peut être 1 200 000 plus rapide qu'un ordinateur ordinaire et qu'un gramme d'ADN peut contenir 108 Téra octets. **[GEH99]**

II.4.1.3. L'évolution remarquable du calcul à l'ADN

Et le développement des outils de la biologie moléculaire ont permis un bon départ pour la cryptographie à l'ADN.

I.4.1.4. Le fort besoin du parallélisme et du stockage dans la cryptographie

Notamment dans le chiffrement et le déchiffrement **[LAM02]** Pour bien expliquer comment l'ADN est utilisé en cryptographie, voici l'exemple d'une méthode cryptographique à l'ADN (la plus connue dans ce domaine) qui utilise plusieurs processus et outils de la biologie moléculaire dans le but de construire un brin d'ADN dont le contenu est très difficile à déchiffrer.

II.4.2 Techniques d'analyse d'ADN

Au cours des 60 dernières années, d'importantes découvertes ont été faites dans la biologie moléculaire. La structure chimique de l'ADN **[WC53]**, la séparation de l'ADN polymérase, le mécanisme de la synthèse biologique, le séquençage de l'ADN. L'analyse de l'ADN et son manipulation a été possible grâce aux techniques importantes qui ont été conçus et développés à cet effet.

Le séquençage d'ADN englobe plusieurs techniques pour déterminer l'ordre des bases de nucléotides dans une séquence d'ADN. Une des méthodes consiste à utiliser des marqueurs fluorescents pour les bases nucléotidiques et l'obtention d'un brin d'ADN complémentaire fluorescent à celui voulu. L'analyse des gènes et une meilleure compréhension de leur rôle dans la vie des organismes sont possibles grâce à ce processus.

La recombinaison de l'ADN est une technique permettant de manipuler les gènes. Il est également appelé «épissage de gènes" ou "génie génétique". Dans cette méthode, certaines protéines - enzymes sont utilisés pour couper et coller des morceaux de la spirale d'ADN. Quand un double brin d'ADN est coupé par cette technique, il aura certaines terminaisons qui peuvent être utilisées pour coller d'autres morceaux d'ADN.

L'hybridation est un processus naturel dans les molécules d'ADN. Il se produit quand deux brins d'ADN complémentaires sont réunis pour former un double brin. Le procédé d'hybridation est lent au départ, jusqu'à ce qu'une région de deux brins complémentaires de liaison apparaisse; le reste du processus d'appariement est rapide si les brins sont tous complémentaires.

Figure 2.15 : Hybridation entre 2 brins complémentaires d'ADN

a) début de la procédé - b) produit final de l'hybridation

La synthèse d'ADN est la création de molécules synthétique d'ADN nommées oligo-nucléotides. Des oligo-nucleotides synthétiques représentent un brin d'ADN habituellement 10-100 nucléotides de longueur. Cette technique peut être utilisée pour remplacer une partie endommagée de l'ADN ou dans la technologie pour le stockage et la transmission de l'information.

La synthèse de l'ADN ainsi que l'hybridation peut être utilisée pour créer différentes formes moléculaire. Un de ces structures étant largement utilisé dans le calcul moléculaire est la tuile d'ADN conçue par Wang Carrelage. H. Wang conçu des carrés de taille égale avec une couleur de chaque côté. Ces carrés peuvent être combinés ensemble en respectant la même couleur pour les côtés voisins. **[TRB99]**

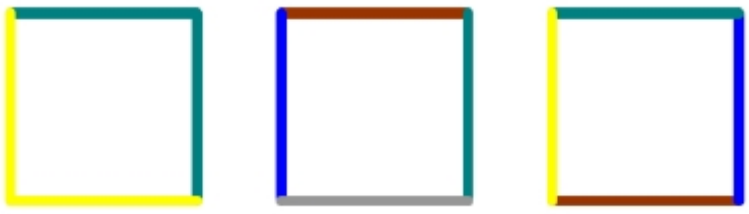


Figure 2.16 : Tuiles De Wang

Le principe de tuiles de Wang a été utilisé dans le domaine du calcul de l'ADN pour créer des structures d'ADN avec des fonctionnalités similaires. Les brins d'oligo-nucléotides sont conçus pour hybrider dans différentes hélices.

La structure finale contient quelques hélices liées entre eux. Les hélices sont formées à partir de différents brins d'ADN. Les terminaisons des tuiles sont nommées «extrémités collantes» parce qu'ils sont formés de nucléotides d'ADN qui peuvent hybrider avec d'autres terminaisons complémentaires, et de cette façon deux tuiles collent les unes aux autres.

II.4.3 Stockage et calculs moléculaire

Le milieu biologique de molécules d'ADN peut être considéré du point de vue du stockage matériau ou la force de calcul. Le génome humain est d'environ 0,72 Go de données en seulement environ 3,5 pictogrammes **[DBV3]**. Cela signifie qu'une grande quantité d'information peut être stockée dans un format d'espace compact, invisible pour nous.

Différentes techniques de dissimulation ont été proposées pour explorer une structure moléculaire à peine perceptible. Une expérience réussie de l'ADN stéganographie est présenté dans **[TRB99]**. Sa consiste à cacher un message d'ADN codé dans le fond d'une autre séquence d'ADN. Des techniques de laboratoire et un point de départ du message est requis afin de le lire.

Il ya une variété de techniques proposant d'introduire des filigranes dans l'ADN d'un organisme d'un être vivant. Un travail important dans ce domaine est présenté dans **[BDH04]**. Un filigrane peut être introduit dans les régions de codon d'ADN utilisant des codons de redondance et de cette façon, des changements fonctionnels n'apparaissent pas. **[SFP02]**.

Pour stocker des données dans des molécules d'ADN, il doit être codé dans alphabet de l'ADN. Dans **[TRB99]**, un procédé de mise en correspondance est présentée où chaque lettre de l'alphabet anglais (A-Y), numéros (0-9), et certains signes de ponctuation où encodés en 3 lettres de l'ADN, comme: Q - AAC, ou 9 - GCG. Considérant que toutes les informations numériques sont codées en binaire, une méthode simple est de faire une table de correspondance entre l'ADN et alphabets binaires (Tableau 2.1) **[TB09]**. En utilisant cette méthode de mise en correspondance, des informations numériques peuvent être transformée facilement dans la séquence d'ADN.

Binaire	ADN
00	A
01	C
10	G
11	T

Tableau 2.1 : Conversion binaire de l'ADN

L'autre utilisation de molécules d'ADN est leur capacité de calcul. Le biomoléculaire calcul est basé sur les techniques existantes en matière d'analyse de manipulation de l'ADN. Le domaine de l'informatique de l'ADN a émergé en raison de l'expérience "in vitro" effectuée sur des molécules d'ADN par Leonard Adleman[1] en 1994 **[Adl94]**. Il a montré que l'informatique biomoléculaire peut être utilisée pour résoudre un problème comme trouver un chemin hamiltonien dans un graphe. Ses réalisations ont apporté un intérêt scientifique à ce domaine, suivies par de nombreuses autres expériences et de découvertes.

En 1980 Nadrian Seeman a présenté la conception des structures ADN contrôlables **[See81]**. Le rôle de ces structures est la construction de blocs de structures moléculaires plus complexes. Le processus d'hybridation est appliqué sur ces unités de base pour obtenir une variété de nanostructures d'ADN.

Afin d'effectuer une opération arithmétique biochimique ADN, les ingrédients suivants sont nécessaires: les outils d'un laboratoire, des structures de matériau d'ADN avec des terminaisons collantes. Solution dans laquelle une hybridation naturelle peut se produire entre ces terminaisons (Fig 2.13.). Les carreaux d'ADN doivent être correctement codés en symboles binaires en vue d'obtenir le fonctionnement souhaité de l'arithmétique binaire.

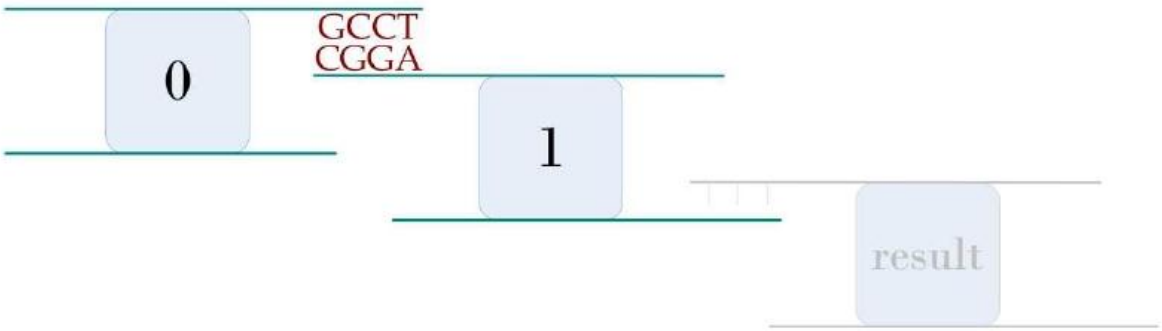


Figure 2.17 hybridation entre des tuiles d'ADN qui peuvent être utilisés pour des opérations arithmétiques

L'informatique ADN peut être utilisé pour mettre en œuvre des algorithmes cryptographiques existants, au niveau moléculaire. Le chiffrement Vernam, basé sur le OU exclusif binaire entre le texte clair et le bitstram du texte chiffré peut être réalisée en utilisant des tuiles d'ADN en les codant en binaires. **[GLR04]**.

[1] **Leonard Max Adleman :** né le 31 décembre 1945, est un chercheur américain en informatique théorique et professeur en informatique et en biologie moléculaire à l'université de la Californie du Sud. [www.futura-sciences.com]

II.4.4 ADN Numérique en cryptographie

L'intérêt de découvrir la séquence exacte de l'ADN à l'intérieur de différents organismes vivants a apporté une énorme quantité de ressources financières, technologiques et humaines à ce sujet. En 1985, le Projet du génome humain (HGP) a commencé. Il a été réalisé en 2003 par l'effort d'une variété d'organisations internationales. La procédure de base est le séquençage de l'ADN. Le partage de données est le principe de base de ce projet: L'adoption de presse libre et de partage de données a été parmi les principales réalisations du Projet génome humain. **[Sanger]**. Par conséquent, il y a en ce moment une grande variété de bases de données génétique et des outils d'analyse en ligne, en accès libre. Un lien vers une liste complète des bases de données génétiques, des organisations et des outils est indiqué dans **[IHGM]**. Il ya une grande quantité de séquences génomiques et protéiques disponibles appartenant à différents organismes.

II.4.5 La méthode de « Olga TORNEA » [OT13]

II.4.5.1 : Principe de « l'indexation ADN »

Olga TORNEA a nommé cette approche « l'indexation ADN ». C'est un chiffrement par flux ou l'information est traitée un octet à la fois.

Le principe et de transformer un octet du texte clair en une séquence de 04 lettre ADN (base nucléotide).

L'étape suivante, est de chercher cette courte séquence dans un brin d'ADN qui est choisi comme clé de chiffrement. Une fois la courte séquence trouvée la position est mémorisé dans un vecteur comme une valeur de substitution potentiel.

Les vecteurs de substitution de touts les octets sont mémorisés dans un dictionnaire

Par conséquent, pour chaque octet du texte clair, il existe un ensemble de valeurs possibles parmi lesquelles une est choisi aléatoirement pour le chiffrement par substitution.

Afin d'obtenir un grand nombre de substitutions pour chaque octet la clé de séquence ADN doit être suffisamment longue, au minimum 30 000 bases. Les étapes de cryptage sont présentées ci-dessous.

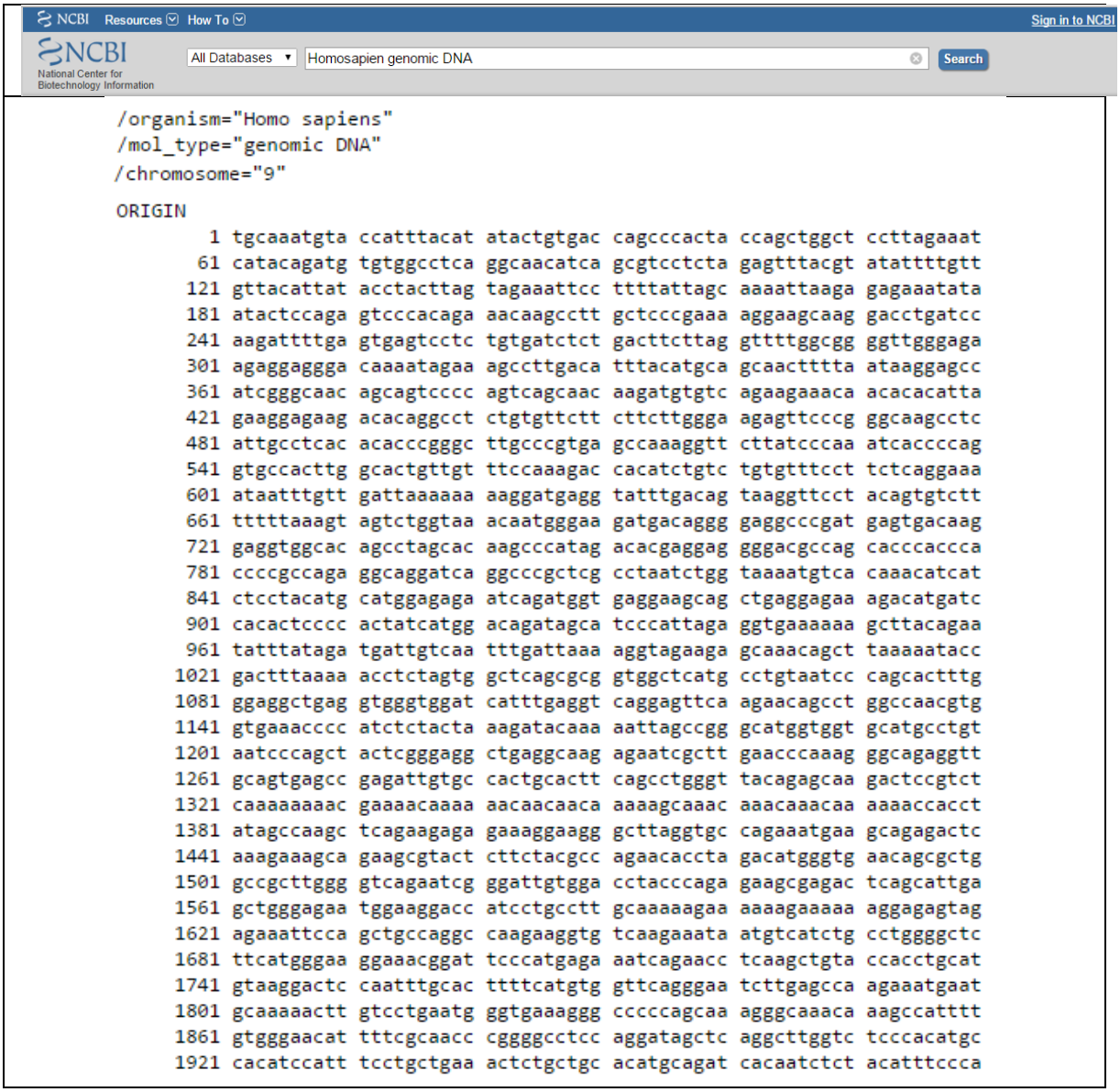


Figure 2.18 : Séquence chromosomique à partir d'une base do donnée génétique

1- calcul des clés de dictionnaire:

a) Chaque octet de 256 valeurs possibles est transformée en une séquence de 4 lettres par le principe suivant: (141 dec) (10 00 11 01 bin) → GATC selon de tableau suivant.

Binaire	Base
00	A
01	C
10	G
11	T

Tableau 2.2 : Table de Conversion du binaire à base nucléotide

b) Une recherche est effectuée pour tous les octets travers une séquence chromosomique composée des lettres A, C, G et T.

c) chaque fois que la courte séquence est récupéré dans la séquence chromosomique, l'indice de cette position est mémorisée dans un vecteur dédié pour cet octet.

d) Le résultat de ces opérations est une matrice de clé de taille 256 x N, où N est une variable longueur, car chaque octet peut avoir un nombre différent de valeurs correspondantes dans ce tableau.

2- Le chiffrement est effectué un octet à la fois. Elle consiste en la substitution de l'octet avec une valeur récupérée de manière aléatoire à partir de son vecteur.

3- Le texte chiffré final est un tableau composé des valeurs entières.

II.4.5.2 : Exemple pratique

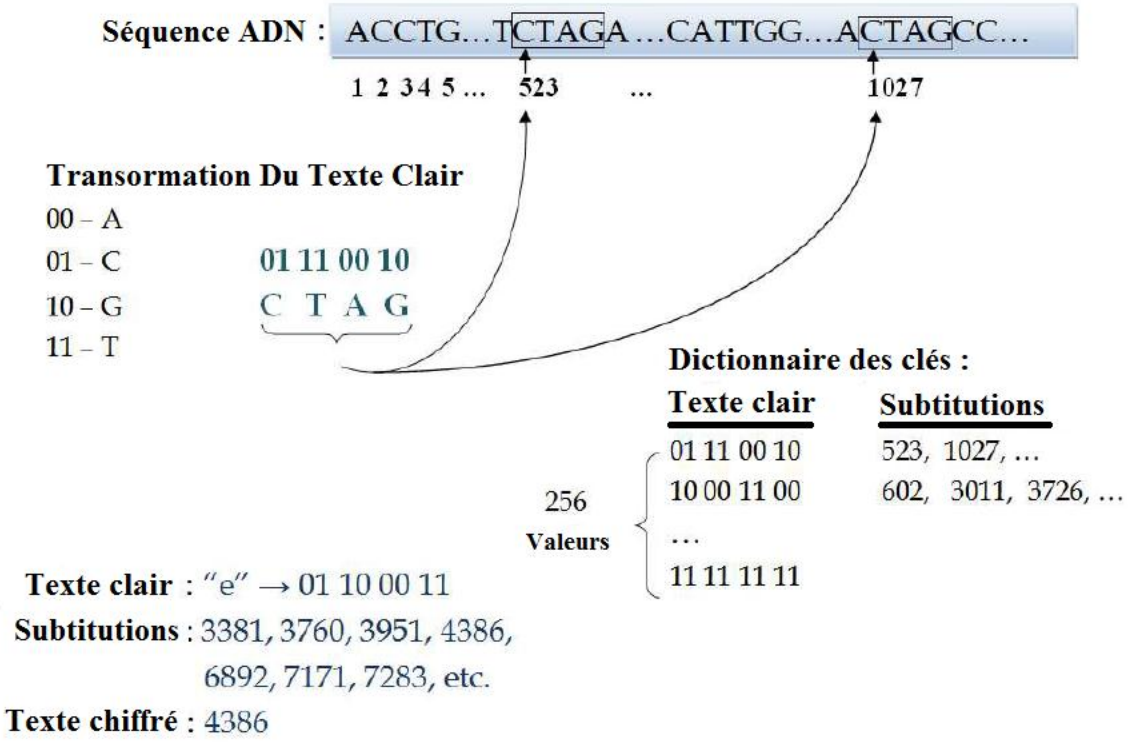


Figure 2.19 : Processus de chiffrement de la méthode « indexation ADN »

II.4.5.3 : Déchiffrement

Etant donné que cela est un algorithme à clé symétrique, la même séquence chromosomique est utilisée pendant le processus de déchiffrement :

- Chaque entier à partir du texte chiffré est utilisé comme pointeur dans la séquence ADN.
- Le déchiffreur lit 4 lettres de la position indiquée et les transforme à une représentation binaire utilisant le même principe réversible présenté dans le tableau 3.1.
- Le texte clair est reconstruit lorsque tous les octets sont extraits de la position indiquée.

II.5 Les aspects d'évaluation de performance

II.5.1 Complexité

La complexité de calcul estime la quantité des ressources nécessaires pour résoudre un certain problème. L'efficacité d'un algorithme peut être évaluée par une analyse théorique, où la théorie de la complexité est impliquée. Une autre façon de tester l'efficacité d'une méthode est de la mettre en œuvre et prendre des mesures du temps écoulé pendant sa performance. Dans la théorie de la complexité, la quantité de ressources utilisées par un algorithme est estimée et calculé sur la base du paramètre d'entrée qui est le nombre d'opérations. Selon la théorie de la complexité, le temps de calcul est une somme de toutes les opérations effectuées. La nombre d'opérations peut être représenté par un nombre fixe ou variable:

- Fixé (paramètre indépendant),
- Exemple: k = 5; a = 9; (Nbr d'opération est 2, O(2))
- Variable (paramètre dépendante)
- Exemple: pour i = 1 $\rightarrow$n {k = k + i; } (Nbr d'opération est n, O (n))

De plus le temps de calcul en théorie de la complexité peut être exprimé par: O(n), Ω(n), ou θ(n). Ils représentent les limites supérieures, inférieures et exactes des ressources nécessaires, et n est le nombre variable d'opérations. Dans la plupart des analyses la notation O(n) est utilisée.

Le temps d'exécution d'un algorithme croît avec la taille de l'entrée, et de sa fonction peut être: logarithmique O($\log_x$n), linéaire O(n), quadratique - O(n^2), cubuque O(n^3), ou exponentielle O(2^n). Une exécution en un taux de croissance logarithmique est la plus optimale. L'exponentielle est le temps à éviter absolument. L'autre méthode pour évaluer le temps d'exécution est de mesurer pendant la performance de l'algorithme. En fonction de l'environnement de développement et le langage de programmation, les fonctions de mesure de temps peuvent varier, mais le principe général est le suivant:

Fonction TempsDeDepart VarDebut;

< Code Algorithme >

Fonction TempsDeFin VarFin;

Duree = VarFin - VarDebut;

Afin de comparer des algorithmes avec cette méthode, ils doivent être testés sur la même machine, développé dans le même langage de programmation et le même environnement. Le taux de croissance du temps de calcul peut être visualisé sur un graphe en calculant l'exécution croissante progressivement des opérations.

II.5.2 Niveau de sécurité

La force de sécurité d'un chiffrement peut être évaluée par différentes techniques comme: mesures statistiques, les attaques de cryptanalyse, l'analyse de l'espace clé et son caractère aléatoire.

Des mesures statistiques comme histogramme, coefficient de corrélation, et l'entropie donnent connaissances sur les tendances dans les informations analysées. La présence de motifs dans le texte chiffré donne l'occasion pour les attaquants de définir une règle par laquelle ils peuvent récupérer des informations utiles sans utiliser la clé. Les techniques statistiques sont importantes en cas d'une attaque de texte chiffré seulement, où un attaquant a l'accès au texte chiffré, mais pas à la clé ou du texte clair correspondant.

L'histogramme est une représentation graphique de la distribution de probabilité statistique du signal. Compte tenu d'une gamme de valeurs discrètes dans un signal, son histogramme montre l'apparition de chaque valeur. Chaque valeur dispose d'un bar dans l'histogramme et il est aussi élevé que la fréquence d'apparition de cette valeur dans le signal. La distribution statistique du texte chiffré peut être analysée par rapport à la distribution du texte clair ou par rapport à lui-même. La distribution du texte clair a généralement certaines formes, motifs, décrivant l'information à l'intérieur. L'histogramme du texte chiffré, si le chiffrement est de sécurité forte, a une distribution aléatoire uniforme, ce qui signifie que les valeurs de texte chiffré ont été générées au hasard dans une distribution uniforme (Fig. 2.20). **[MOV96]**.

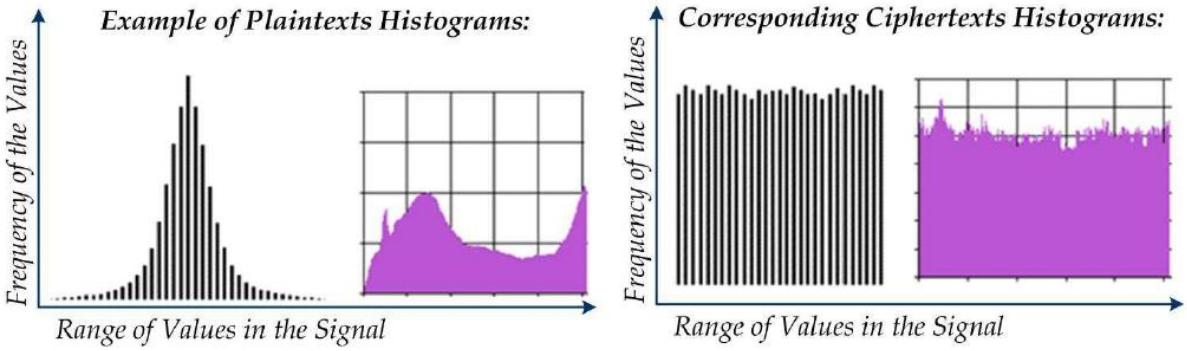


Figure 2.20 Histogrammes des textes clairs et chiffrés

L'entropie mesure l'incertitude ou le caractère aléatoire du signal. Considérant une certaine variable *X* définie par un ensemble de valeurs {x1 , x2 … x3} avec sa probabilité.

1- $P(X = xi) = Pi$
2- $0 \leq Pi \leq 1 , \forall i \in \{ 1 .. n \}$
3- $\sum_{i=1}^{n} P_i = 1$

Considérant la variable X un signal exprimé en binaire, son information propre est

$$I(xi)=-\log_2 Pi \text{ (bits/symboles)}$$

L'entropie de X est la quantité moyen de l'informations. Elle est définie et mesurée comme:

$$H(x) = -\sum_{i=1}^{n} P_i \, log_2 Pi \text{ (bits/symboles)}$$

L'interval des valeurs de l'entropie pour la variable X est :

1- $0 \leq H(X) \leq \log_2 n$
2- $H(X)=0 \Leftrightarrow \exists$! Pi $\in$ P : pi =1 et pj =0 $\forall j \neq i$
3- $H(X) = \log_2 n \Leftrightarrow$ pi = 1/n , $\forall i \in \{1..n\}$

Lorsque l'entropie est égale à zéro, cela signifie qu'il n'y a pas d''incertitude sur la prédiction du signal. Lorsque l'entropie est à sa valeur maximale, cela signifie que tous les symboles du signal sont tout aussi susceptibles de se produire **[Sha48]**. Pour cette raison, l'entropie du texte chiffré doit être aussi élevée que possible. Dans le texte clair, une corrélation entre les différents groupes de bits peut apparaître, en fonction du type de données. Par exemple, dans une image, il y a souvent une forte corrélation entre les pixels voisins. Lorsqu'une image est cryptée, il est souhaitable d'éviter la présence de cette corrélation dans le texte chiffré. La corrélation entre les valeurs de texte chiffré peut être mesurée par le coefficient de corrélation **[RN88]**.

Le coefficient de corrélation de Pearson[1] (r_{xy}) est la formule la plus utilisée à déterminer la corrélation linéaire entre deux variables X et Y. Considérée normalement distribué et linéaire l'une par rapport à l'autres, les variables X et Y définis par des ensembles de valeurs {x1 , x2 .. xn} et {y1 , y2 … yn} respectivement. Leur coefficient de corrélation de Pearson peut être exprimé comme **[You06]**:

$$r_{xy} = \frac{Cov(X,Y)}{\sqrt{Var(X)Var(Y)}} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Quand $\bar{X}$= 1/n - $\sum_{i=1}^{n}$ Xi est la valeur moyenne de X et $\bar{Y}$ est la valeur moyenne de Y.

Quand Les valeurs des coefficients de corrélation proche de zéro signifie qu'il ya une faible relation linéaire entre des valeurs comparées, et quand ils sont proches de un cette relation est forte. Dans l'analyse du texte chiffré, il est souhaitable d'obtenir de faibles valeurs de coefficient de corrélation.

[1] **Karl Pearson** (27 mars 1857–27 avril 1936), mathématicien britannique, est un des fondateurs de la statistique moderne. Il est aujourd'hui principalement connu pour avoir développé le coefficient de corrélation. [www.futura-sciences.com]

L'espace de clé de chiffrement est défini comme étant un ensemble de toutes les clés possibles qui peuvent être formés pour que ce chiffrement. La quantité de toutes les clés possibles est donnée par l'alphabet dans laquelle la clé est représentés élevé à la puissance de sa longueur. Un espace de clé plus importante signifie une meilleure résistance à l'attaque par force brute, d'où une meilleure sécurité.

Exemple:

La clé : 110 – Alphabet : 2 lettres (0 et 1) – Longueur de la clé=3

Espace de clés = (2^3) = 8 : 000, 001, 010, 011, 100, 101, 110, 111

« *Un texte chiffré aléatoirement peut être obtenu à partir d'un texte clair non aléatoire si il est combinée avec une clé de séquence aléatoire* » **[Sch96]**.

« *Une réelle génération de nombres aléatoires nécessite une source naturelle de caractère aléatoire* » **[MOV96]**.

« Un vrai aléatoire ne peut être obtenue par des fonctions mathématiques. Seuls certains processus physiques peuvent garantir vraiment des nombre aléatoire» **[Sch09]**.

L'utilisation de formules mathématiques génère des nombres pseudo-aléatoires: nombres qui apparaissent au hasard, mais sont en réalité prédéterminés. Des vrais nombres aléatoires, proviennent à partir de mesures de phénomènes physiques aléatoires comme le bruit atmosphérique ou des sources radioactives. Les matériaux d'ADN offrent la possibilité de générer des nombres aléatoires de deux façons: grâce à son processus d'hybridation et de sa séquence.

II.6 Conclusion du chapitre

Dans ce chapitre, nous avons présenté la structure basique de la molécule d'ADN. Beaucoup de chercheurs se sont intéressés à la capacité du calcul à l'ADN pour résoudre d'autres problèmes complexes**.** Le développement remarquable du calcul à l'ADN donne la naissance de la cryptographie à l'ADN et ceci en exploitant le parallélisme ainsi que la capacité importante du stockage qu'offre cette molécule. La recherche en cryptographie à l'ADN est dans sa phase initiale et il reste beaucoup de problèmes à résoudre.

Cependant les avantages de l'ADN en stockage, parallélisme ainsi que le développement remarquable des équipements de la biologie moléculaire donnent l'espoir de voir des méthodes cryptographiques à l'ADN s'imposer comme étant des méthodes très puissantes dans les systèmes cryptographiques modernes. Et comme il se fut un temps où un ordinateur occupait une salle entière et aujourd'hui il occupe une surface de quelque centimètres carrés, il arrivera certainement un jour où la cryptographie à l'ADN pourra être pratiquée d'une manière très simple et avec des équipements plus petits et moins onéreux.

Et en attendant ce jour, plusieurs chercheurs ont préféré s'inspirer de l'ADN ou d'autres systèmes biologiques pour concevoir des méthodes cryptographiques offrant un bon niveau de sécurité.

La méthode de « Olga TORNEA » **[OT13]** détaillée dans ce chapitre, est un exemple de ces méthodes. C'est en se basant sur cette méthode que nous proposons dans le chapitre suivant une méthode cryptographique inspirée de l'ADN.

Aussi, nous allons proposer une approche inspirée du chiffre de Vernam et nous allons discuter les points positifs et les points négatifs de chaque approche.

Contenu en bref

III.1 Introduction

III.2 Contribution N° 01

III.3 Contribution N° 02

III.4 Conclusion du chapitre

III.1 Introduction :

Les algorithmes de chiffrement symétriques ont une grande importance dans la sécurité des systèmes informatiques, essentiellement dans la sécurisation des transferts de données via des réseaux non sécurisés. La plupart des systèmes de chiffrement actuels sont des systèmes *bloc cipher* (chiffrement par bloc), car ils offrent un niveau de sécurité très élevé et un temps d'exécution relativement court. Dans ce chapitre, nous proposons de concevoir un algorithme cryptographique symétrique par bloc inspiré de l'ADN.

Nous avons étudiés l'approche de « Olga TORNEA » et on s'est inspiré de cette approche afin de lui apporter des modifications et des améliorations.

Comme nous allons essayer de se rapprocher du chiffrement parfait de Vernam on se basant sur la diversité et la quantité de l'information contenue dans l'ADN.

Tout au long de ce chapitre, nous présenterons en détails les techniques et principes utilisés dans la conception de ces deux algorithmes.

III.2 Contribution N° 01 :

Nous avons baptisé cet algorithme : « Stegano-Adn ». L'Algorithme que nous proposons est un algorithme symétrique (à clef secrète). Il se divise en deux partie : partie brouillage et partie chiffrement. Inspiré de plusieurs autres algorithmes déjà connu

- DES
- ENIGMA
- BOOK CYPHER

La clé de chiffrement et aussi divisée en deux partie : Une partie choisit par le chiffreur et une partie générée par l'algorithme.

Et pour mieux expliquer l'algorithme, on va décrire chaque étape séparément.

Phase 1 : Codage binaire et découpage en blocs

Chaque caractère du texte clair est codé sur 8 bits selon la valeur ascii. Exemple le caractère A soit 65_{Dec} est codé en : 01000001_{Bin}.

Ensuite, le texte clair codé en binaire est découpé en blocs de 64 bits. Et le dernier bloc n'atteignant pas les 64 bits est complété par le caractère « espace » soit 32_{Dec} soit 00100000_{Bin}.

Phase 2 : Codage en base nucléotide

Dans cette étape, on va procéder à une substitution de chaque paire de bits en une base nucléotide selon le tableau 2.2 illustré dans le chapitre 2.

Au résultat comme étant chaque paire de bit est convertit en un seul caractère qui représente une base nucléotide. A partir de chaque bloc de 64 bits, nous obtiendrons un bloc de 32 caractères (bases).

Phase 3 : Brouillage

Cette étape consiste à effectuer un brouillage du texte clair afin d'éliminer l'ordre logique des lettres dans le but de défendre le chiffrement de la cryptanalyse « attaque statistique ».

On s'inspirant de DES, nous avons créé huit (08) boites de substitution de seize (16) bits chacune. (Voire figure 3.2).

Les boites sont conçus d'une façon purement aléatoire, il n'y a aucune logique ni corrélation dans la disposition des indices de substitution et aucun algorithme n'est utilisé pour obtenir cette disposition.

La phase de brouillage est un algorithme itératif de huit (08) tours (par défaut). C'est une inspiration du schéma de Feistel selon la figure 3.1

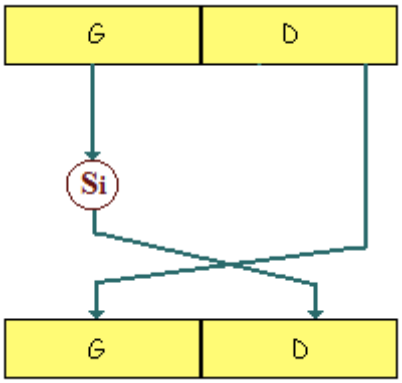


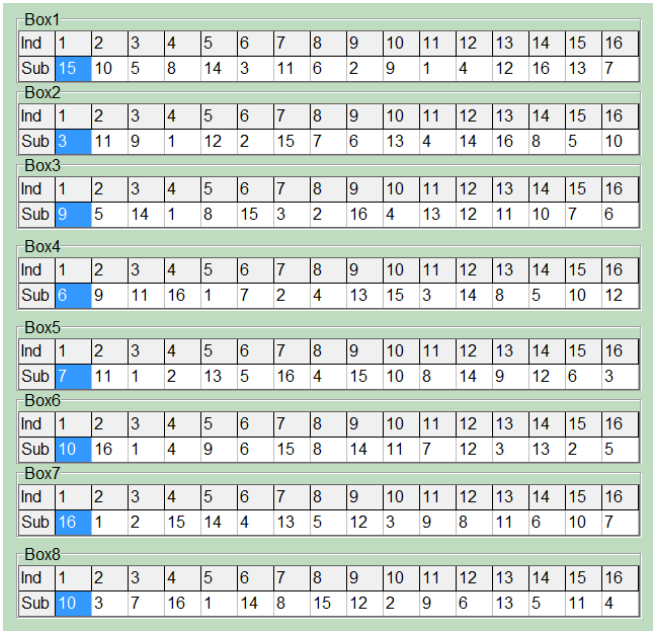
Figure 3.1 : Brouillage inspiré du schéma Feistel

Le Nombre de tours est huit (08) par défaut, il est choisit par le chiffreur, il peut être augmenté comme il peut être réduit.

Pour un brouillage robuste, il est conseillé de ne pas aller au dessous de six (06) tours. **[Pat04]**.

L'ordre des boites est 1.2.3.4.5.6.7.8 par défaut, le chiffreur peut choisir n'importe quelle séquence constituée de 0..8 (0 : pas de substitution). Cette séquence fait partie de la clé de chiffrement.

Ce choix de séquence est inspiré de la fameuse machine à chiffré ENIGMA, la machine standard avait trois (03) rotors, ensuite elle a évoluée pour atteindre six (06) rotors. Les six rotors sont différents les uns des autres. L'ordre des rotors a une influence sur le chiffrement. Cet ordre fait partie de la clé de chiffrement.



Box1

Ind	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Sub	15	10	5	8	14	3	11	6	2	9	1	4	12	16	13	7

Box2

Ind	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Sub	3	11	9	1	12	2	15	7	6	13	4	14	16	8	5	10

Box3

Ind	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Sub	9	5	14	1	8	15	3	2	16	4	13	12	11	10	7	6

Box4

Ind	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Sub	6	9	11	16	1	7	2	4	13	15	3	14	8	5	10	12

Box5

Ind	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Sub	7	11	1	2	13	5	16	4	15	10	8	14	9	12	6	3

Box6

Ind	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Sub	10	16	1	4	9	6	15	8	14	11	7	12	3	13	2	5

Box7

Ind	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Sub	16	1	2	15	14	4	13	5	12	3	9	8	11	6	10	7

Box8

Ind	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Sub	10	3	7	16	1	14	8	15	12	2	9	6	13	5	11	4

Figure 3.2 : Les boites de substitution.

Phase 4 : Chiffrement

Cette étape est une copie type de la méthode de « Olga TORNEA » avec une légère modification.

Le chiffreur doit choisir un brin d'ADN au préalable, le principe de chiffrement est inspiré du chiffrement Book Cypher. Le principe du Book Cypher et de choisir un livre comme clé secrète et de chiffrer chaque mot du message clair avec un triplet qui représente (page, ligne, ordre) dans le livre **[RS10]**.

Le chiffrement est un chiffrement par flux, où l'information est traitée un octet à la fois (04 bases). Le principe consiste à effectuer une recherche de cette courte séquence d'ADN dans la séquence chromosomique qui a été choisie préalablement comme une partie de la clé secrète. Chaque fois qu'une séquence d'octets de texte brouillé est récupérée dans la séquence chromosomique, la position est mémorisée dans un vecteur tel que l'une des valeurs possibles pour le chiffrement par substitution pour cette séquence d'ADN. Les vecteurs de substitutions pour tous les octets sont mémorisés dans un dictionnaire clé. Par conséquent, pour chaque octet du texte brouillé il existe une gamme de valeurs possibles à partir de laquelle est choisi au hasard une pour le chiffrement par substitution. Afin d'obtenir un grand nombre de substitutions pour chaque octet, le brin d'ADN doit être suffisamment long, par exemple 30 000 bases minimum.

La modification apportée à l'approche de « Olga TORNEA » est la suivante :

Mme. Olga TORNEA a effectué une recherche de la courte séquence ADN dans la séquence chromosomique et a mémorisé tout les positions possibles de cette courte séquence alors qu'elle peu n'apparaitre qu'une seule fois dans le texte clair, et comme étant la séquence chromosomique est une base de donnée, il y aura un accès disque de plus en plus fréquent. Et il est connu que les accès disque sont très couteux en temps. Alors dans notre approche pour gérer cette contrainte, nous avons préférer ne mémoriser que le nombre des positions nécessaire. Une courte séquence qui n'apparait qu'une seule fois dans le texte brouillé possède un vecteur avec une seule valeur.

Et pour renforcer la robustesse de notre algorithme une position de départ qui définit la position initiale -soit la position N° 01- est choisie par le chiffreur

Exemple : si la position 100 est définie comme position de départ sa implique que la position 101 deviendra la position N° 02.

Phase 5 : Génération de la clé de chiffrement

Afin de rendre le chiffrement plus robuste et ne pas laisser les positions des courtes séquences lisibles. On à opté pour collé toute les positions ensemble sous la forme d'une suite de numéro ambigu, toute en laissant une trace pour séparer les positions entre elles.

L'astuce est simple. Il s'agit de calculer les sommes des chiffres qui composent les positions et maintenir ces sommes comme clé de chiffrement.

Et pour mieux illustrer ce principe, voici un exemple applicatif.

III.2.1 Exemple applicatif

III.2.1.1 : Chiffrement

M. Meftah envoie un message confidentiel à M. Benyahia : «Un Message Secret .»

Phase 1 : Codage binaire et découpage en bloc de 64 bits

Caractère	Code Ascii	Code binaire
U	85	01010101
n	110	01101110
Espace	32	00100000
M	77	01001101
e	101	01100101
s	115	01110011
s	115	01110011
a	97	01100001
g	103	01100111
e	101	01100101
Espace	32	00100000
S	83	01010011
e	101	01100101
c	99	01100011
r	114	01110010
e	101	01100101
t	116	01110100
Espace	32	00100000
.	46	00101110

Tableau 3.1 : Conversion caractère / Ascci / Binaire

Voici la séquence binaire claire :
01010101 01101110 00100000 01001101 01100101 01110011 01110011 01100001 01100111 01100101 00100000 01010011 01100101 01100011 01110010 01100101 01110100 00100000 00101110

Ensuite, le texte clair codé en binaire est découpé en blocs de 64 bits. Et le dernier bloc n'atteignant pas les 64 bit est complété par le caractère « espace » soit 00100000_{Bin}.

01010101 01101110 00100000 01001101 01100101 01110011 01110011 01100001
01100111 01100101 00100000 01010011 01100101 01100011 01110010 01100101
01110100 00100000 00101110 00100000 00100000 00100000 00100000 00100000

On obtient trois (03) blocs de 64 bits

Observation : L'espace entre les octets n'existe pas, il est inséré juste pour une meilleur lisibilité.

Phase 2 : Codage en base nucléotide

Selon le codage suivant : 00 : A 01 : C 10 : G 11 : T

Prenons le caractère « U » soit « 01010101 » en binaire

01 : C 01 : C 01 : C 01 : C

Donc le caractère « U » est codé en « cccc » comme base nucléotide

Prenons le caractère « n » soit « 01101110 » en binaire

01 : C 10 : G 11 : T 10 : G

Donc le caractère « n » est codé en « cgtg » comme base nucléotide

Prenons le caractère « espace » soit « 00100000 » en binaire

00 : A 10 : G 00 : A 00 : A

Donc le caractère « espace » est codé en « agaa » comme base nucléotide

On obtient les trois (03) blocs suivant de 32 caractères chacun :

```
cccc cgtg agaa catc cgcc ctat ctat cgac

cgct cgcc agaa ccat cgcc cgat ctag cgcc

ctca agaa agtg agaa agaa agaa agaa agaa
```

Phase 3 : Brouillage

Pour cette étape, on va illustrer juste la 1ère boite de substitution

Box1

Ind	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Sub	15	10	5	8	14	3	11	6	2	9	1	4	12	16	13	7

Voici le 1er bloc : `cccccgtgagaacatccgccctatctatcgac`

En utilisant le principe illustré à la figure 3.1, en divise le bloc en deux partie, partie gauche G et partie droite D. (Voir figure 3.3).

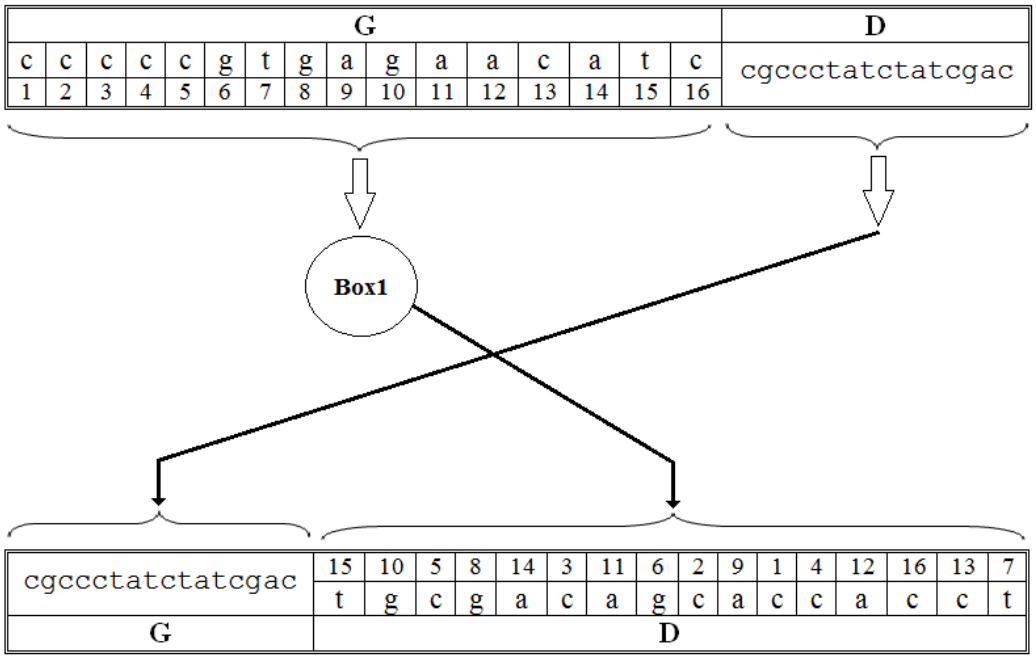


Figure 3.3 : Illustration de la procédure de brouillage avec la boite N° 01

On obtient le bloc 01 brouillé suivant :

`cgccctatctatcgactgcgacagcaccacct`

Et en finalisant les autres blocs, voici le résultat :

```
cgccctatctatcgactgcgacagcaccacct
cgcccgatctagcgccagccccaggactatcc
agaaagaaagaaagaaagaagctgtacagaaa
```

Phase 4 : Chiffrement

En divise le bloc en séquence de quatre bases, on effectue une recherche de chaque séquence dans la séquence chromosomique choisis au préalable, et on sauvegarde les positions de chaque séquence dans un vecteur. Dans notre cas, on à choisis le chromosome n° 09 du génome humain. La séquence chromosomique à été extraite du site NCBI (National Centre for Biotechnology Information).

L'étape suivante consiste à choisir aléatoirement une position pour représenter une séquence de quatre bases.

IT°	Nombre d'apparition	Bloc 1		Position Retenue
1	3	**cgcc**	765 ,784,809	809
2	2	**ctat**	912 , 1379	1379
3	2	**ctat**	Déjà traitée : Itération 2	912
4	1	**cgac**	1020	1020
5	1	**tgcg**	4711	4711
6	1	**acag**	64	64
7	1	**cacc**	492	492
8	1	**acct**	131	131
		Bloc 2		
9	3	**cgcc**	Déjà traitée : Itération 1	765
10	1	**cgat**	707	707
11	1	**ctag**	98	98
12	1	**cgcc**	Déjà traitée : Itération 1	765
13	1	**agcc**	32	32
14	1	**ccag**	30	30
15	1	**gact**	271	271
16	1	**atcc**	237	237
		Bloc 3		
17	5	**agaa**	55 , 142 , 172 , 198 , 298	198
18	5	**agaa**	Déjà traitée : Itération 17	55
19	5	**agaa**	Déjà traitée : Itération 17	172
20	5	**agaa**	Déjà traitée : Itération 17	198
21	5	**agaa**	Déjà traitée : Itération 17	298
22	1	**gctg**	44	44
23	1	**taca**	16	16
24	1	**gaaa**	56	56

Tableau 3.2 : Dictionnaire des index

Voici le résultat de l'étape 5 :

809,1379,912,1020,4711,64,492,131,765,707,98,765,32,30,271,237,198,55,172,198,298,44,16,56

Phase 5 : Génération de la clé de chiffrement

On calcul la somme de chaque position est le résultat est une partie de la clé de chiffrement.

Avec le résultat obtenu à l'étape 4 :
809,1379,912,1020,4711,64,492,131,765,707,98,765,32,30,271,237,198,55,172,198,298,44,16,56

On obtient la séquence suivante :

17,20,12,3,13,10,15,5,18,14,17,18,5,3,10,12,18,10,10,18,19,8,7,11

Et le résultat final du chiffrement est :

809137991210204711644921317657079876532302712371985517219829844 1656

Et la clé du chiffrement est :

Chromosome N° 9 du génome humain

Position de départ : 1

Nombre et ordre des boites de brouillage : 1

Clé générée : 17,20,12,3,13,10,15,5,18,14,17,18,5,3,10,12,18,10,10,18,19,8,7,11

III.2.1.2 Déchiffrement

On reçoit le texte chiffré :

809137991210204711644921317657079876532302712371985517219829844 1656

Et la clé du chiffrement :

Chromosome N° 9 du génome humain

Position de départ : 1

Nombre et ordre des boites de brouillage : 1

La clé : 17,20,12,3,13,10,15,5,18,14,17,18,5,3,10,12,18,10,10,18,19,8,7,11

Phase 1 : Déchiffrer les positions

On déchiffre les positions du texte chiffré à l'aide de la clé, chaque entier de la clé correspond à la somme des chiffres qui composent une position.

809	**1379**	**912**	**1020**	**4711**	**64**	**492**	**131**	**765**	**707**	**98**	**765**
17	20	12	3	13	10	15	5	18	14	17	18

32	**30**	**271**	**237**	**198**	**55**	**172**	**198**	**298**	**44**	**16**	**56**
5	3	10	12	18	10	10	18	19	8	7	11

Tableau 3.3 : Processus de déchiffrement des positions

On obtient le résultat suivant :

809,1379,912,1020,4711,64,492,131,765,707,98,765,32,30,271,237,198,55,172,198,298, 44,16,56

Phase 2 : Bases correspondants

On effectue un accès direct aux positions obtenue dans la séquence ADN clé, on prenant on considération la position de départ choisie. (ici position N° 01)
Voici le résultat :

Position	Courte séquence correspondante
809	cgcc
1379	ctat
912	ctat
1020	cgac
4711	tgcg
64	acag
492	cacc
131	acct
765	cgcc
707	cgat
98	ctag
765	cgcc
32	agcc
30	ccag
271	gact
237	atcc
198	agaa
55	agaa
172	agaa
198	agaa
298	agaa
44	gctg
16	taca
56	gaaa

Tableau 3.4 : Indexation des bases nucléotides

Voici les blocs obtenus :

```
cgccctatctatcgactgcgacagcaccacct
cgcccgatctagcgccagccccaggactatcc
agaaagaaagaaagaaagaagctgtacagaaa
```

Phase 3 : Enlever le brouillage

En utilisant la boite de substitution N° 01 choisie dans la clé de chiffrement, on doit enlever le brouillage. (Voire figure 3.4).

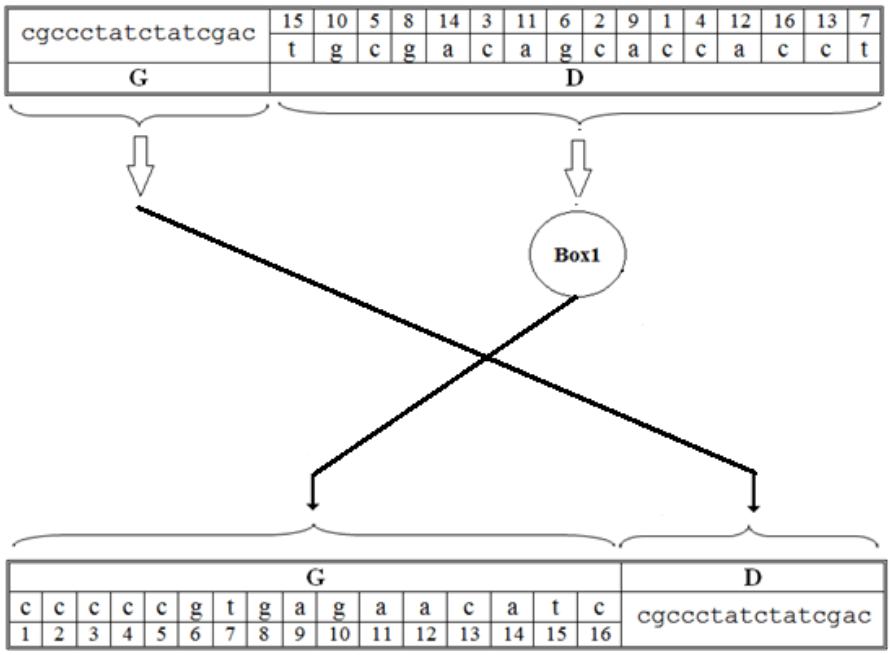


Figure 3.4 : Processus pour enlever le brouillage

Voici le résultat obtenu : `cccccgtgagaacatccgccctatctatcgac`
`cgctcgccagaaccatcgcccgatctagcgcc`
`ctcaagaaagtgagaaagaaagaaagaaagaa`

Phase 4 : Conversion en binaire

Selon le codage suivant : 00 : A 01 : C 10 : G 11 : T

Bloc 1 :

c	c	c	c	c	g	t	g	a	g	a	a	c	a	t	c	c	g	c	c	c	t	a	t	c	t	a	t	c	g	a	c
0	0	0	0	0	1	1	1	1	1	0	0	0	0	1	0	0	1	0	0	0	1	0	1	0	1	0	1	0	1	0	0
1	1	1	1	1	0	1	0	1	0	0	0	1	0	1	1	1	0	1	1	1	1	0	1	1	1	0	1	1	0	0	1

Bloc 2 :

c	g	c	t	c	g	c	c	a	g	a	a	c	c	a	t	c	g	c	c	c	g	a	t	c	t	a	g	c	g	c	c
0	1	0	1	0	1	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0	1	0	1	0	1	0	1	0	1	0	0
1	0	1	1	1	0	1	1	0	0	0	0	1	1	0	1	1	0	1	1	1	0	0	1	1	1	0	0	1	0	1	1

Bloc 3 :

c	t	c	a	a	g	a	a	a	g	t	g	a	g	a	a	a	g	a	a	a	g	a	a	a	g	a	a	a	g	a	a
0	1	0	0	0	1	0	0	0	1	1	1	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tableau 3.5 : Conversion base / code binaire

Voici le résultat obtenu :

0101010101101110001000000100110101100101011100110111001101100001
0110011101100101001000000101001101100101011000110111001001100101
0111010000100000001011100010000000100000001000000010000000100000

Phase 5 : Conversion en texte

On effectue la conversion binaire-décimal en suite décimal-Ascii

Code binaire	Code Ascii	Caractère
01010101	85	U
01101110	110	n
00100000	32	Espace
01001101	77	M
01100101	101	e
01110011	115	s
01110011	115	s
01100001	97	a
01100111	103	g
01100101	101	e
00100000	32	Espace
01010011	83	S

Code binaire	Code Ascii	Caractère
01100101	101	e
01100011	99	c
01110010	114	r
01100101	101	e
01110100	116	t
00100000	32	Espace
00101110	46	.
00100000	32	Espace
00100000	32	Espace
00100000	32	Espace
00100000	32	Espace
00100000	32	Espace

Tableau 3.6 : Conversion Binaire / Ascii / Caractère

III.3 Contribution N° 02 :

Nous avons baptisé cet algorithme : « Vern-Adn ». L'Algorithme que nous proposons est un algorithme symétrique (à clef secrète). Il se divise en deux partie : partie brouillage et partie chiffrement. Inspiré de plusieurs autres algorithmes déjà connu

- DES
- ENIGMA
- VERNAM

Cette approche à en commun avec la précédente les trois premières phases :

Phase 1 : Codage binaire et découpage en blocs de 64 bits
Phase 2 : Codage en base nucléotide
Phase 3 : Brouillage
Phase 4 : Chiffrement

Le chiffreur doit choisir un brin d'ADN au préalable, le principe de chiffrement est inspiré du chiffrement de VERNAM. Il s'agit de choisir une clé aussi longue que le texte claire. Cette clé doit avoir un usage unique. Ensuite effectuer un ou exclusif logique entre le texte clair et la clé.

L'algorithme génère une suite de chiffre aléatoire correspondant au nombre de bloc résultants de la phase 3.

Chaque chiffre correspond au début d'une séquence ADN de 32 bases du brin d'ADN choisi au préalable.

Ensuite, en effectue un XOR biologique entre le bloc résultat de l'étape 3 et la séquence ADN qui débute à la position du le chiffre de la clé.

Comme pour la contribution précédente, pour renforcer la robustesse de notre algorithme, une position de départ qui définit la position initiale -soit la position N° 01- est choisie par le chiffreur

Et pour mieux illustrer ce principe, voici un exemple applicatif.

III.3.1. Exemple applicatif

III.3.1.1 : Chiffrement

M. Meftah envoie un message confidentiel à M. Benyahia : «Un Message Secret .»

Phase 1 : Codage binaire et découpage en bloc de 64 bits :

Caractère	Code Ascii	Code binaire
U	85	01010101
N	110	01101110
Espace	32	00100000
M	77	01001101
E	101	01100101
S	115	01110011
S	115	01110011
A	97	01100001
g	103	01100111
e	101	01100101
Espace	32	00100000
S	83	01010011
e	101	01100101
c	99	01100011
r	114	01110010
e	101	01100101
t	116	01110100
Espace	32	00100000
.	46	00101110

Tableau 3.7 : Conversion caractère / Ascci / Binaire

Voici la séquence binaire claire :

01010101 01101110 00100000 01001101 01100101 01110011 01110011 01100001 01100111 01100101 00100000 01010011 01100101 01100011 01110010 01100101 01110100 00100000 00101110

Ensuite, le texte clair codé en binaire est découpé en blocs de 64 bits. Et le dernier bloc n'atteignant pas les 64 bit est complété par le caractère « espace » soit 00100000_{Bin}.

01010101 01101110 00100000 01001101 01100101 01110011 01110011 01100001
01100111 01100101 00100000 01010011 01100101 01100011 01110010 01100101
01110100 00100000 00101110 00100000 00100000 00100000 00100000 00100000

On obtient trois (03) blocs de 64 bits

Phase 2 : Codage en base nucléotide

Selon le codage suivant : 00 : A 01 : C 10 : G 11 : T

On obtient les trois (03) blocs suivant de 32 caractères chacun :

```
cccc cgtg agaa catc cgcc ctat ctat cgac
cgct cgcc agaa ccat cgcc cgat ctag cgcc
ctca agaa agtg agaa agaa agaa agaa agaa
```

Phase 3 : Brouillage

Pour cette étape, on va illustrer juste la 1ère boite de substitution

Box1

Ind	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Sub	15	10	5	8	14	3	11	6	2	9	1	4	12	16	13	7

Voici le 1er bloc :

```
cccccgtgagaacatccgccctatctatcgac
```

En utilisant le principe illustré à la figure 3.1, en divise le bloc en deux partie, partie gauche G et partie droite D. (Voir figure 3.3). Et exactement comme détaillé à la phase 3 du chiffrement de la 1ère contribution.

On obtient le bloc 01 brouillé suivant :

```
cgccctatctatcgactgcgacagcaccacct
```

Et en finalisant les autres blocs, voici le résultat :

```
cgccctatctatcgactgcgacagcaccacct
cgcccgatctagcgccagccccaggactatcc
agaaagaaagaaagaaagaagctgtacagaaa
```

Phase 4 : Chiffrement

L'algorithme génère une séquence de chiffres aléatoires correspondant au nombre de bloc de 32 bases obtenus à l'étape 3.

On extrait les séquences ADN de 32 bases du brin d'ADN choisi au préalable à partir des positions de la séquence aléatoire. Dans notre cas, on à choisis le chromosome n° 09 du génome humain. La séquence chromosomique à été extraite du site NCBI (National Centre for Biotechnology Information).

Voici l'opération XOR biologique entre les bases nucléotides :

x	y	x xor y
a	a	a
c	a	c
g	a	g
t	a	t

x	y	x xor y
a	c	c
c	c	a
g	c	t
t	c	g

x	y	x xor y
a	g	g
c	g	t
g	g	a
t	g	c

x	y	x xor y
a	t	t
c	t	g
g	t	c
t	t	a

Tableau 3.8 : XOR Biologique

Chiffre aléatoire	Bloc ADN 32 bases nucléotides
87	atcagcgtcctctagagtttacgtatattttg
1443	agaaagcagaagcgtactcttctacgccagaa
37310	acgactggacttctcaaggctcccaggtcctg

Tableau 3.9 : Séquence aléatoires générée

Ensuite on effectue un XOR biologique avec les blocs de l'étape 4

Bloc	Séquence
Bloc 1 Phase 3	cgccctatctatcgactgcgacagcaccacct
Bloc 1 Phase 4	atcagcgtcctctagagtttacgtatattttg
Bloc 1 Résultat	ccactggaagtggggcccgcaagcctcgtggc
Bloc 2 Phase 3	cgcccgatctagcgccagccccaggactatcc
Bloc 2 Phase 4	agaaagcagaagcgtactcttctacgccagaa
Bloc 2 Résultat	cacccactttaaaagcccaggatgtgagaccc
Bloc 3 Phase 3	agaaagaaagaaagaaagaagctgtacagaaa
Bloc 3 Phase 4	acgactggacttctcaaggctcccaggtcctg
Bloc 3 Résultat	atgaccggatttcccaaagccagttgtttctg

Tableau 3.10 : Résultat de l'opération XOR

Voici le résultat de la phase 4 :

```
ccactggaagtggggcccgcaagcctcgtggc
cacccactttaaaagcccaggatgtgagaccc
atgaccggatttcccaaagccagttgtttctg
```

Et le résultat final du chiffrement est :

```
ccactggaagtggggcccgcaagcctcgtggc
cacccactttaaaagcccaggatgtgagaccc
atgaccggatttcccaaagccagttgtttctg
```

Et la clé du chiffrement est :

Chromosome N° 9 du génome humain

Position de départ : 1

Nombre et ordre des boites de brouillage : 1

Clé générée : 87,1443,37310

III.3.1.2 Déchiffrement

On reçoit le texte chiffré :

```
ccactggaagtggggcccgcaagcctcgtggc
cacccactttaaaagcccaggatgtgagaccc
atgaccggatttcccaaagccagttgtttctg
```

Et la clé du chiffrement :

Chromosome N° 9 du génome humain

Nombre et ordre des boites de brouillage : 1

La clé : 87,1443,37310

Phase 1 : Extraire les blocs de bases nucléotide correspondantes :

Chaque chiffre de la clé correspond à une séquence ADN de 32 bases.

La clé	Bloc ADN 32 bases nucléotides
87	atcagcgtcctctagagtttacgtatattttg
1443	agaaagcagaagcgtactcttctacgccagaa
37310	acgactggacttctcaaggctcccaggtcctg

Tableau 3.11 : Correspondance index-Bloc ADN

Ensuite on effectue un XOR biologique avec les blocs du texte chiffré

Bloc	Séquence
Bloc 1 texte chiffré	ccactggaagtggggcccgcaagcctcgtggc
Bloc 1 Bloc extrait ADN	atcagcgtcctctagagtttacgtatattttg
Bloc 1 Résultat	cgccctatctatcgactgcgacagcaccacct
Bloc 2 texte chiffré	cacccactttaaaagcccaggatgtgagaccc
Bloc 2 Bloc extrait ADN	agaaagcagaagcgtactcttctacgccagaa
Bloc 2 Résultat	cgcccgatctagcgccagccccaggactatcc
Bloc 3 texte chiffré	atgaccggatttcccaaagccagttgtttctg
Bloc 3 Bloc extrait ADN	acgactggacttctcaaggctcccaggtcctg
Bloc 3 Résultat	agaaagaaagaaagaaagaagctgtacagaaa

Tableau 3.12 : Résultat de l'opération XOR

Phase 2 : Enlever le brouillage

En utilisant la boite de substitution N° 01 choisie dans la clé de chiffrement, on doit enlever le brouillage. (Voire figure 3.4). Exactement comme illustré à la phase 3 du déchiffrement de la 1ère contribution. Voici le résultat obtenu :

ccccgtgagaacatccgccctatctatcgac
cgctcgccagaaccatcgcccgatctagcgcc
ctcaagaaagtgagaaagaaagaaagaaagaa

Phase 3 : Conversion en binaire

Selon le codage suivant : 00 : A 01 : C 10 : G 11 : T Voici le résultat obtenu :
01010101011011100010000001001101011001010111001101110011011000 01
01100111011001010010000001010011011001010110001101110010011001 01
01110100001000000010111000100000001000000010000000100000001000 00

Phase 4 : Conversion en texte

On effectue la conversion binaire-décimal en suite décimal-Ascii

Code binaire	Code Ascii	Caractère
01010101	85	U
01101110	110	n
00100000	32	Espace
01001101	77	M
01100101	101	e
01110011	115	s
01110011	115	s
01100001	97	a
01100111	103	g
01100101	101	e
00100000	32	Espace
01010011	83	S

Code Binaire	Code Ascii	Caractère
01100101	101	e
01100011	99	c
01110010	114	r
01100101	101	e
01110100	116	t
00100000	32	Espace
00101110	46	.
00100000	32	Espace
00100000	32	Espace
00100000	32	Espace
00100000	32	Espace
00100000	32	Espace

Tableau 3.13 : Conversion Binaire / Ascii / Caractère

III.4 Conclusion du chapitre

Dans ce chapitre, nous avons présenté nos contributions cryptographiques, ainsi que les détails de chaque étape les constituants. Nos algorithmes sont inspirés de techniques déjà connues qui ont prouvé leur efficacité.

Stégano-ADN : Cet algorithme est un croisement de chiffrement et de stéganographie. Il consiste d'effectuer un brouillage du message secret à l'aide de boite de substitution en combinant DES et ENIGMA ensuite de chercher la présence du texte dans une séquence ADN comme si le texte à été tatoué sur l'ADN.

VernADN : Cet algorithme est un chiffrement qui consiste en bref d'effectuer un brouillage du message secret à l'aide de boite de substitution en combinant DES et ENIGMA, ensuite d'extraire aléatoirement des séquences ADN pour appliquer le chiffre de VERNAM.

Contenu en bref

IV.1 Introduction

IV.2 Environnement de développement

IV.3 Les classes principales

IV.4 Tests et résultats expérimentaux de Stegano-ADN

IV.5 Tests et résultats expérimentaux de VernADN.

IV.6 Conclusion du chapitre

IV.1 Introduction

Dans ce chapitre nous allons présenter la mise en œuvre des algorithmes proposés, ainsi que les résultats des différents tests effectués dans le but d'évaluer leurs performances. Les tests des méthodes cryptographiques s'effectuent sur deux axes importants :

Axe temps d'exécution :

Ces tests ont pour objectifs d'évaluer les performances de l'algorithme de point de vue temps de chiffrement/déchiffrement et ceci en variant certains de ses paramètres (nombre de tours, longueur de la séquence ADN … etc)

Axe sécurité :

Ces tests ont pour objectifs d'évaluer la sécurité de l'algorithme, Autrement dit, évaluer la résistance de l'algorithme devant les attaques des cryptanalyste.

IV.2 Environnement de développement :

Le système d'exploitation choisi pour la réalisation de notre application est le système Windows 7. Nous avons utilisé un ordinateur hp pro3500 i7-3ème gen / 8Go ram.

Nous avons choisi le langage Pascal pour développer notre système cryptographique.

Le langage de programmation Pascal dont le nom vient du mathématicien français Blaise Pascal a été inventé par « Niklaus Wirth » dans les années 1970. Il a été conçu pour servir à l'enseignement de la programmation de manière rigoureuse mais simple, il se caractérise par une syntaxe claire, rigoureuse et facilitant la structuration des programmes.

En dehors de sa syntaxe et de sa rigueur, le langage Pascal possède de nombreux points communs avec le C comme les pointeurs. Le langage Pascal de base était conçu à usage purement éducatif et était assez limité. Par exemple, les chaînes de caractères, absentes du langage d'origine, ont rapidement été intégrées. Les développements qu'il a connus par la suite en ont fait un langage complet et efficace.

Les implémentations actuelles de Pascal, utilisées hors du monde éducatif, sont des extensions telles que Turbo Pascal, Pascal Objet, et Delphi (EDI). Il existe des versions libres comme Free Pascal et Lazarus (EDI). On peut programmer en Pascal sous DOS, Windows, Mac OS ou encore sous Linux/Unix ou Palm OS.

Le système d'exploitation des ordinateurs Apollo, ainsi qu'une partie du système du Macintosh ont été écrits en Pascal. La première version d'Adobe Photoshop également. Le compilateur GCC a été développé par Richard Stallman à partir d'un compilateur du LLNL, qui était écrit en langage Pastel, une extension du langage Pascal.

Outre son orientation objet, le langage pascal a l'avantage rigoureux, la plupart des erreurs se produisent à la compilation et non à l'exécution.

En 1995, pour contrecarrer Microsoft et la programmation visuelle du Visual Basic, Borland sort Delphi. On voit également apparaître la bibliothèque VCL servant d'interface aux bibliothèques système de Windows, facilitant grandement le développement.

Au début des années 2000, Borland produit Kylix, l'équivalent de Delphi pour le monde Linux. Au début des années 2010, Embarcadero, qui a repris les activités d'outils de développement de Borland, produit Delphi XE, dont les versions récentes sont compatibles avec Windows, OS X et iOS.

Lazarus, est un logiciel libre de développement intégré RAD légèrement différent, permettant de compiler sur différents systèmes d'exploitation tel que Windows, GNU/Linux, OS X, Unix,OS/2, ReactOS, Haiku et plateformes telles que x86, x86-64, ARM, SPARC, PowerPC, IA-64.

Les environnements de développement intégrés (EDI), sont des logiciels regroupant un ensemble d'outils nécessaires au développement logiciel dans un ou plusieurs langages de programmation. Parmi tous les environnements de développement existant notre choix a été porté sur Delphi 2010 à la faveur de sa rapidité à mettre en place une application.

C'est un environnement de développement intégré (IDE) pour Pascal. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, éditeur graphique d'interfaces et de pages web, intégration de solution pour les smartphones et tablettes ANDROID et IOS, support de solutions de gestion de versions, intégration de débogueur…). **[NET1]** Delphi 2010 est disponible sous Windows, Linux, Mac OS.

IV.3. Les classes principales :

L'ensemble des classes et leurs méthodes respectives pour les deux algo sont :

- ***ChiffrementBloc :*** Cette classe regroupe les différentes procédures et fonctions nécessaires pour le chiffrement d'un bloc (64 bits).
- ***DéchiffrementBloc :*** Cette classe regroupe les différentes procédures et fonctions nécessaires pour le déchiffrement d'un bloc (64 bits).
- ***BoiteAOutil :*** Cette classe contient plusieurs procédures et fonctions qu'on aura besoin dans notre application.

IV.4. Tests & résultats expérimentaux de Stegano-ADN

Dans ce qui suit, nous allons présenter les différents tests et évaluations effectués ainsi que les résultats obtenus sur l'algorithme Stégano-ADN.

On tien à préciser que les tests ont été effectués sur une machine HP Pro 3500 dotée d'un processeur i7-3770 3ème génération de 3.40 Ghz avec 8 Go de RAM, et d'un système d'exploitation Windows 7 Professionnel 64 bits

IV.4.1 Complexité de l'algorithme

L'analyse de la complexité d'un algorithme est importante car elle révèle son efficacité pour les applications en temps réel. Dans ce travail, le temps de calcul de l'algorithme a été analysé à l'aide des méthodes de la théorie de la complexité. Les conclusions obtenues ont été testées pour être vrai à travers les résultats de mise en œuvre. Le temps d'exécution d'un algorithme est considéré comme la somme de toutes les opérations. Le nombre d'opérations peut être constant ou variable et dépend des paramètres d'entrée.

Selon les approximations de la théorie de la complexité, la limite inferieur de la fonction est utilisée pour exprimer le taux de croissance de l'exécution de l'algorithme **[Has88]**. Par conséquent, si le nombre d'opérations est par exemple 1 + 2n, la complexité serait O(n); si le nombre d'opérations est 4 + n + n3, alors la complexité serait $O(n^3)$.

Dans ce travail, la complexité analysé pour 3 opérations importantes de l'indexation de l'ADN : Calcul du dictionnaire de clé, le cryptage et le décryptage. La phase computation du dictionnaire des index est calculée en m opérations, m est la longueur de la séquence ADN utilisée. La phase cryptage et le décryptage sont exécutés en n opérations, où n est le nombre de mots de texte en clair et de texte chiffré. La complexité résultante :

Pour le chiffrement : Le calcul du dictionnaire des index est O(m) et pour le processus de chiffrement est O(n). ça implique que la complexité de l'indexation et chiffrement est O(m+n). Cela signifie que le taux croissant du temps de calcul est linéaire en fonction de la longueur du la séquence ADN et la taille du texte claire.

Pour le déchiffrement : le processus de déchiffrement est O(n). Cela signifie que le taux croissant du temps de calcul est linéaire en fonction de la taille du texte claire uniquement et que la longueur du la séquence ADN n'a pas d'influence sur le temps d'exécution.

IV.4.2. Variation du temps d'exécution

IV.4.2.1 Selon la longueur de la séquence ADN clé :

Dans ce test, nous avons un choisis un fichier texte avec une taille fixe. Ensuite, nous avons effectué le test plusieurs fois avec 8 tours de brouillage, tout en variant la longueur de la séquence ADN. Puis nous avons évalué le temps moyen de chiffrement/déchiffrement.

Test N°	Taille du fichier	Longueur ADN	Temps chiffrement	Temps déchiffrement
Test N° 1	2 042 octets	1000	252 ms	2 ms
Test N° 2		5000	1s 80 ms	2 ms
Test N° 3		10000	2s 600 ms	3 ms
Test N° 4		20000	5s 69 ms	3 ms
Test N° 5		40000	10s 125 ms	4 ms

Tableau 4.1 Mesure du temps d'exécution selon la longueur de la séquence ADN

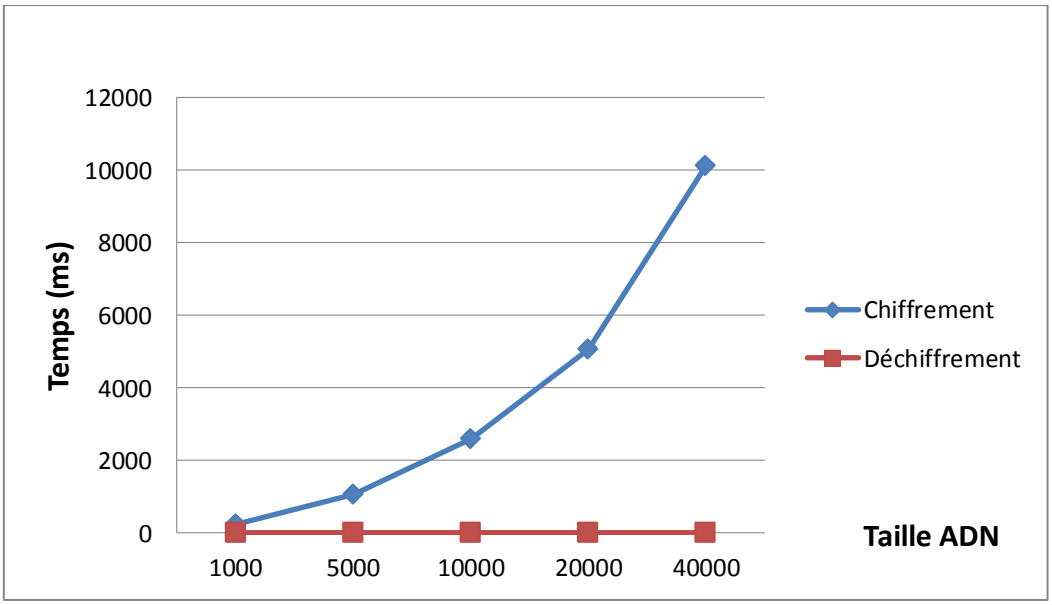


Figure 4.1 : Variation du temps d'exécution selon la longueur de la séquence ADN

Analyse:

Avec la variation de la longueur de la séquence ADN clé, le temps de chiffrement évolue d'une façon linéaire. Par conséquent, le temps de déchiffrement est presque constant du faite que cette variante n'a pas d'influence sur le déchiffrement.

IV.4.2.2 Selon la longueur de taille du texte clair :

Dans ce test, nous avons fixé la longueur de la séquence ADN clé à 30000 bases. Ensuite, nous avons effectué le test plusieurs fois avec 8 tours de brouillage, tout en variant la taille du texte clair. Puis nous avons évalué le temps moyen de chiffrement/déchiffrement.

Taille du texte clair	Temps chiffrement	Temps déchiffrement
2 Ko	7s 700 ms	4 ms
12 Ko	46s 250 ms	14 ms
30 Ko	117 s 900 ms	65 ms
60 Ko	240 s 333 ms	145 ms
100 Ko	401 s 52 ms	216 ms

Tableau 4.2 Mesure du temps d'exécution selon la taille du texte claire

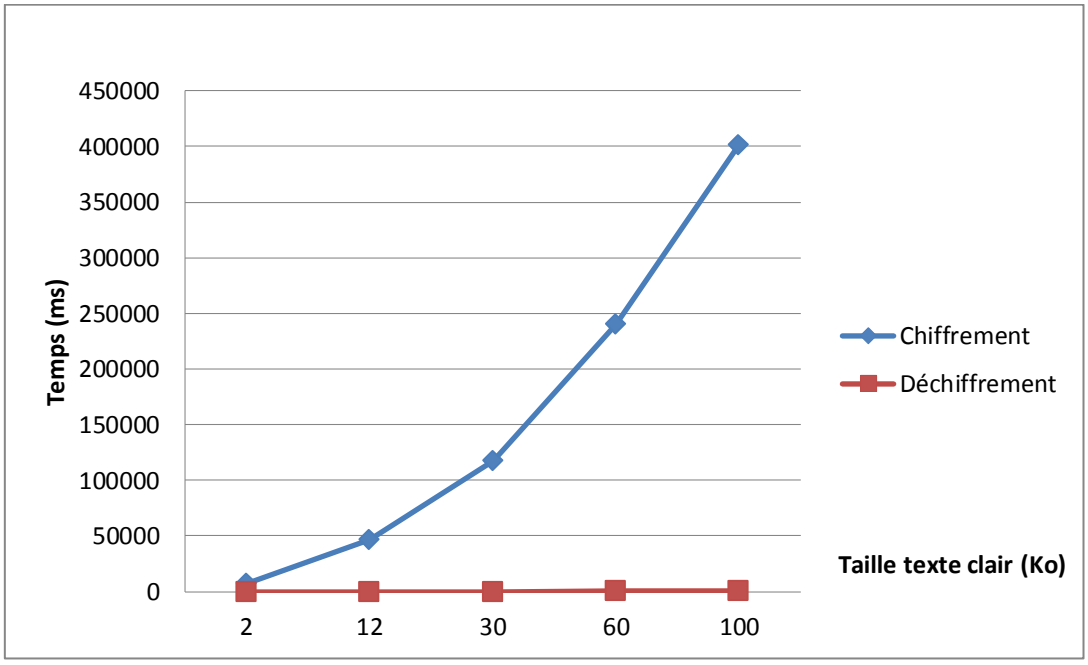


Figure 4.2 : Variation du temps d'exécution selon la taille du texte clair

Analyse:

On remarque que le temps de déchiffrement est beaucoup plus inferieur au temps de chiffrement. Cela est du au faite que le chiffrement contient la phase computation du dictionnaire des index qui prend beaucoup de temps. Par contre, dans le processus du déchiffrement il s'agit d'un accès indexé direct. C'est pour ça qu'il est beaucoup plus rapide.

On remarque aussi que le temps de chiffrement (respectivement déchiffrement) augmente d'une façon linéaire avec l'augmentation de la taille du texte clair.

IV.4.3 Analyse de l'espace clé :

L'espace clé d'un algorithme de chiffrement doit être aussi grand que possible, afin de résister à des attaques par force brute. Si la clé est une séquence d'alphabet, son espace sera : nombre d'alphabet$^{Long(Clé)}$.

Essayer toutes les clés possibles donnera nombre d'alphabet$^{Long(Clé)}$ nombre de tentatives pour obtenir une clé réussie. En moyenne, la bonne réponse peut être trouvée dans la moitié de ce nombre d'essais.

Dans le cas de l'indexation ADN, la clé se compose de 4 parties : La séquence génétique, ordre des boites, position de départ et la clé générée.

- Par défaut le nombre des boites de substitution est 8, incluant le 0 implique 9^8 combinaison = 43 046 721.
- Si comme déjà mentionné au chapitre 2, on choisit une séquence ADN de 30000 bases implique 30 000 positions de départ possible
- La séquence ADN clé est une séquence génétique. Dans ce cas, d'essayer toutes les clés possibles signifie essayer toutes les séquences génétiques d'une base de données. L'une des bases de données peut être la base de la séquence génétique du NIH nommé GenBank. C'est une collection de toutes les séquences ADN publiquement disponibles **[wncbi]**. Elle contient environ 135440924 enregistrements de séquence ADN. En cas d'utilisation d'une base de données privée (non publiée), l'espace clé tend vers l'infini.
- L'espace clé = 43 046 721*30 000*135 440 924 = 1, 7490863002230612 * 10^{20}

Comme indiqué au tableau 4.2, un message de 100Ko avec une séquence ADN de 30000 bases, le déchiffrement dure 216ms

Avec 1749086300223061200000 clés possible, une attaque par force brute durera : 1749086300223061200000 * 216 = ?

37 780 264 084 818 121 920 000	Ms
37 780 264 084 818 121 920	Seconds
629 671 068 080 302 032	Minutes
10 494 517 801 338 367	Heures
437 271 575 055 765	Jours
1 198 004 315 221	Années
11 980 043 152	Siècles

Ceci sans compter l'espace de la clé générée.

IV.4.4 Comparaison entre notre pseudo code et le pseudo code de Olga TORNEA

Calcul du dictionnaire des index	Chiffrement	Déchiffrement
Pour X = 1 à Long(SeqDNA) DicCle[SeqDNA(x:x+3)] = DicCle[SeqDNA(x:x+3)] + x Fin Pour	Pour X = 1 à Long(TClaire) Index = Random(1 , Long(DicCle[x])) TChiffré[x] =DicCle[x][Index] Fin Pour	Pour X = 1 à Long(TChiffré) Index = TChiffré[x] TClaire = TClaire + SeqDNA(Index:Index+3) Fin Pour

Key Dictionary Computation:	Encryption:	Decryption:

m = length(SeqDNA)
next = 0
FOR X = 1 to 256
FOR Y = 1 to m
IF X = SeqDNA(y:y+3)
KeyDic[x][next] = y
next++;
END IF
END FOR
END FOR
	n = length(Plaintext)
	FOR X = 1 to n
	Index = RandomNo(1,
	length(KeyDic[x]))
	Ciphertext[x] =
	KeyDic[x][Index]
	END FOR
		n = length(Ciphertext)
		FOR X = 1 to n
		Index = Ciphertext[x]
		ByteOfPlaintext =
		SeqDNA(Index:Index+3)
		END FOR

Tableau 4.3 : Pseudo code Stegano-ADN VS DNA Indexation Cipher de OLGA

Analyse :

En remarque que pour les deux opérations chiffrement et déchiffrement, le pseudo code des deux algorithmes sont exactement identique. Sauf que pour l'opération calcul du dictionnaire des index, l'algorithme de Dr OLGA fait 256*m opérations telle que m est la longueur de la séquence ADN clé utilisée. Alors que notre algorithme ne fait que m opérations. Et comme la longueur de la séquence ADN est importante (minimum 30000 bases), l'impacte est palpable. Entre 256 * 30000 = 7 680 000 op et 30 000 op la différence est frappante. En d'autres termes, un texte qui se chiffre en 1mn avec notre algo, se chiffre en 256mn avec celui de OLGA

IV.5. Tests & résultats expérimentaux de VernADN

Dans ce qui suit, nous allons présenter les différents tests et évaluations effectués ainsi que les résultats obtenus sur l'algorithme VernADN.

On tien à préciser que les tests ont été effectués sur une machine HP Pro 4530s dotée d'un processeur i3 de 3.30 Ghz avec 4 Go de RAM, et d'un système d'exploitation Windows 7 Professionnel 64 bits.

IV.5.1. Evaluation du taux de chiffrement :

Dans ce test, nous avons un choisis un fichier texte. Ensuite, nous avons effectué le test plusieurs fois avec 8 tours de brouillage, puis nous avons évalué le temps moyen de chiffrement/déchiffrement.

Test N°	Taille du fichier	Nombre de bloc 64 bits	Temps chiffrement	Temps déchiffrement
Test N° 1	2 042 octets	256	78 ms	54 ms
Test N° 2			66 ms	52 ms
Test N° 3			72 ms	48 ms
Test N° 4			69 ms	46 ms
Test N° 5			64 ms	51 ms

Tableau 4.4 : Temps d'exécution chiffrement/déchiffrement du fichier texte.

Temps moyen de chiffrement : 69,8 ms

Temps moyen de déchiffrement : 50,2 ms

A partir de ces tests et plusieurs d'autres tests, nous avons estimé le taux de chiffrement et déchiffrement :

Taux chiffrement = taille de fichier / temps de chiffrement de fichier = 29,25 o / ms

Taux déchiffrement = taille de fichier / temps de déchiffrement de fichier = 40,67 o / ms

IV.5.2. Variation du temps de chiffrement/déchiffrement selon la taille de l'entrée :

Dans ce test, nous avons suivi la variation de temps chiffrement/déchiffrement en augmentant la taille de fichier en clair, et nous avons reporté les résultats dans le tableau suivant :

Taille	Temps chiffrement	Temps déchiffrement
2042 o	66 ms	48 ms
12 Ko	492 ms	456 ms
132 Ko	36 s 120 ms	29 s 65 ms
40 Mo	15 h 33 m	13 h 23 m
80 Mo	49 h 52 m	44 h 13 m

Tableau 4.5 : Relation temps d'exécution avec la variation de la taille de fichier.

Le graphe suivant représente la relation entre la taille du texte clair et le temps de chiffrement et déchiffrement.

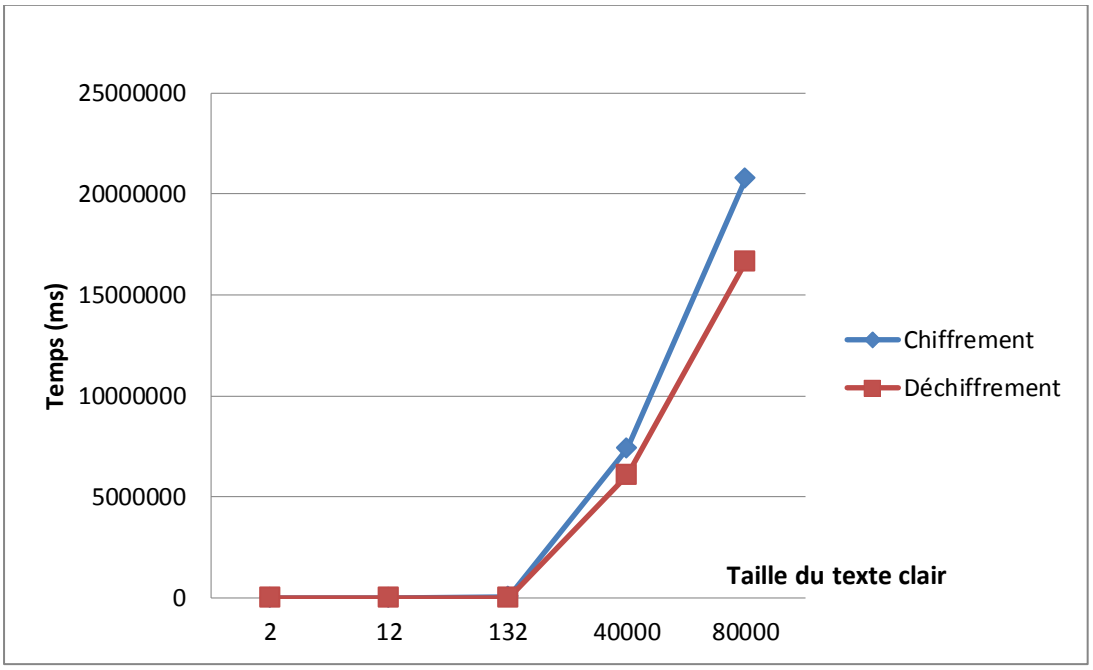


Figure 4.3 : Relation temps chiffrement - déchiffrement / taille du texte clair.

Analyse:

On remarque que le temps de déchiffrement est inferieur au temps de chiffrement. Cela est du au faite que le chiffrement contient la phase génération des chiffres aléatoire qui est absente dans le processus de déchiffrement.

On remarque aussi que le temps de chiffrement (respectivement déchiffrement) augmente d'une façon linéaire avec l'augmentation de la taille du texte clair.

Cela s'explique que à chaque fois que le texte claire augmente de taille, le temps de chiffrement (respectivement déchiffrement) augmente autant de fois qu'il ya de phase. Il se multiplie par cinq dans le processus de chiffrement et par quatre dans le processus de déchiffrement.

IV.5.3. Variation de la taille du texte chiffré selon la taille du texte clair:

Dans ce test, nous avons suivi la variation de la taille du texte chiffré en augmentant la taille de texte clair, et nous avons reporté les résultats dans le tableau suivant :

Taille du texte claire	Taille du texte chiffré
2 Ko	8 Ko
8 Ko	32 Ko
16 Ko	64 Ko
32 Mo	128 Mo
64 Mo	256 Mo

Tableau 4.6 : Relation taille du texte clair / taille du texte chiffré.

Le graphe suivant représente la relation entre la taille du texte clair et la taille du texte chiffré.

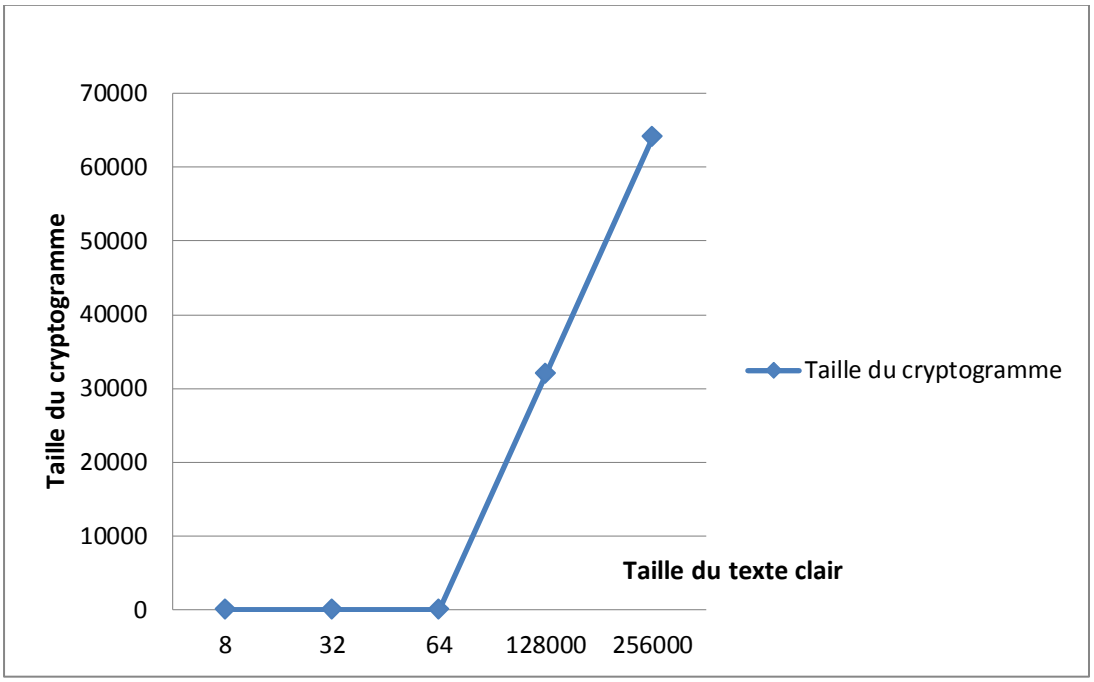


Figure 4.4 : Relation taille du texte clair / taille du texte chiffré.

Analyse:

On remarque que la taille du texte chiffré et toujours quatre fois supérieure à la taille du texte clair. Tout simplement parce que chaque caractère et chiffré en quatre bases nucléotides.

IV.5.4. Comparaison en temps d'exécution entre l'algorithme proposé et AES:

Dans ce test, nous avons effectué une étude comparative entre notre algorithme et le standard connu AES en temps de chiffrement / déchiffrement, ce dernier à été implémenté dans le même environnement de développement (Delphi 2010) et la même machine. Cette implémentation est téléchargeable gratuitement sur « www.code-source.com ».

Taille du fichier texte (kb)	AES		VernADN	
	Chiffrement	Déchiffrement	Chiffrement	Déchiffrement
Fichier 1 : 105 kb	15 ms	14 ms	26 s 2 ms	21 s 14 ms
Fichier 2 : 199 kb	17 ms	16 ms	48 s 18 ms	39 s 12 ms
Fichier 3 : 475 kb	46 ms	45 ms	2 m 3 s 14 ms	1 m 60 s 9 ms
Fichier 4 : 786 kb	62 ms	61 ms	4 m 30 s 5 ms	3 m 9 s 14 ms
Fichier 5 : 1510 kb	140 ms	140 ms	10 m 14 s 85 ms	8 m 45 s 12 ms

Tableau 4.7 : AES vs VernADN.

Analyse :

Ces tests montrent que AES est beaucoup plus rapide que notre algorithme exécuté dans les mêmes conditions. Cette différence est du à :

Dans AES, on trouve quatre procédures : SubByte (Substitution), ShiftRow (Décalage ligne), MixColomn (Permutation) et un XOR. Il est évident que ces quatre procédures consomment moins de temps parce qu'il s'agit de manipulation de chaine de caractère. Alors que notre algorithme contient dans le processus de chiffrement cinq procédures :

1 - Texte2Bin : Conversion du texte en binaire.
2 - Bin2Base : Conversion du code binaire en bases nucléotides.
3 - SubBase : Substitution des blocs de bases.
4 - KeyGen : Génération de la clé de chiffrement.
5 - XorADN : Ou exclusif avec des séquences ADN.

Les quatre premières procédures sont très rapides. C'est la cinquième procédure qui est très couteuse en temps d'exécution parce qu'il s'agit d'un accès disque à la base de donnée génétique afin d'extraire les séquences ADN désignées par la clé générée.

De même pour le déchiffrement notre algorithme contient quatre procédures :

1 - ADNXor : Ou exclusif avec des séquences ADN avec le texte chiffré.
2 - BaseSub : Enlever le brouillage à l'aide des boites de substitution.
3 - Base2Bin : Conversion des base nucléotide en binaire.
4 - Bin2Texte : Conversion du code binaire en texte.

Uniquement la première procédure est très couteuse en temps d'exécution parce qu'il s'agit d'un accès disque afin d'extraire les séquences ADN désignées par la clé.

IV.5.5 Analyse de l'espace clé :

Dans le cas de VernADN, la clé se compose de 4 parties : La séquence génétique, ordre des boites, position de départ et la séquence aléatoire.

Les 3 premières parties, ont le même espace clé que Stegano-ADN. Ajoutant la séquence aléatoire on doit multiplier le résultat par (longueur de la séquence ADN élevé à la puissance du nombre de bloc du texte clair.

IV.6. Comparaison Stegano-ADN & VernADN

IV.6.1. Temps d'exécution :

Test	Taille du fichier	ADN	Stegano-ADN		VernADN	
			Chiff	Déchiff	Chiff	Déchiff
1	2 042 octets	1000	252 ms	2 ms	78 ms	54 ms
2		5000	1s 80 ms	2 ms	66 ms	52 ms
3		10000	2s 600 ms	3 ms	72 ms	48 ms
4		20000	5s 69 ms	3 ms	69 ms	46 ms
5		40000	10s 125 ms	4 ms	64 ms	51 ms

Tableau 4.8 : Stegano-ADN VS VernADN

L'étude de la complexité et les tests sur Stegano-ADN ont démontré que ce crypto-système et très rapide. Dans son processus de déchiffrement, ne dépondant que de la taille du texte clair, et s'agissant d'un accès direct à l'information, il croit linéairement et dépasse de loin le temps de déchiffrement de VernADN. Alors que le processus de chiffrement, dépondant et de la taille de la séquence ADN clé et de la taille du texte clair croit linéairement mais d'une façon très rapide dépassant d'une façon assez remarquable le temps de chiffrement de VernADN.

Pour conclure, on peu dire que Stegano-ADN remporte la partie dans le processus de déchiffrement et VernADN la remporte dans le processus de chiffrement.

IV.6.1. Robustesse :

L'étude de l'espace clé de Stegano-ADN démontre que cet algorithme est très robuste et qu'il est théoriquement incassable.

Aussi, la robustesse de VernADN à été démontré, sauf que ce dernier à une faiblesse, c'est la fonction de génération de nombres aléatoire ou plus exactement pseudo-aléatoire. Si elle est mise à nu, l'espace clé peut diminuer et malgré ça il reste très important.

Autre avantage de VernADN, on n'a pas besoin de transmettre une clé (OTP) aussi long que le texte, il suffit juste de transmettre les index des séquences ADN.

IV.7. Conclusion du chapitre

Dans ce chapitre, nous avons vu l'environnement de développement de nos algorithmes. Ensuit, nous avons effectué plusieurs tests afin de bien évaluer leurs performance, notamment son comportement quand les paramètres se varient.

On à présenté un chiffrement symétrique basé sur des séquences en utilisant l'ADN biologique numérique à partir des bases de données génétiques publiques afin de récupérer des séquences et de les utiliser pour le chiffrement.

L'algorithme Stegano-ADN représente une application pratique de la cryptographie de l'ADN et apporte un grand avantage d'utiliser des bases de données génomiques; une caractéristique qui n'a pas été beaucoup exploitée en cryptographie.

Le temps d'exécution, et le niveau de sécurité ont été analysés pour cet algorithme. Le Runtime est linéaire, et des mesures statistiques ont montré une bonne sécurité du cryptogramme.

Comme nous avons essayé d'implémenter le chiffrement de Vernam réputé être le chiffrement parfait en exploitant la quantité d'information incluse dans l'ADN et sa diversité.

Cependant, les tests de la sécurité des algorithmes proposés n'étaient pas suffisants, et ceci à cause de manque des outils de cryptanalyse universelles mais aussi la difficulté de réaliser l'une des attaques connues.

CONCLUSION GENERALE

Ce travail est une recherche scientifique sur la cryptographie ADN. C'est une direction émergente et prometteuse en cryptographie.

L'avènement du calcul à l'ADN proposé par Adleman en 1994 a ouvert les portes à l'utilisation de cette molécule comme outil de calcul et de résolution de problèmes complexes.

Ensuite, les cryptographes ont vu en l'ADN un outil très puissant pour concevoir des méthodes de chiffrements.

Ces méthodes peuvent utiliser l'ADN dans sa structure biologique dans un laboratoire avec des outils biologique adéquats. Comme ils peuvent utiliser l'ADN dans sa structure numérique avec des ordinateurs comme outils de travail.

Aperçu des Contributions

Dans le chapitre 3, une méthode de chiffrement symétrique grâce à l'indexation de l'ADN, est présentée. Elle a été conçue pour utiliser des bases de données génétiques afin de récupérer des séquences d'ADN et les utiliser pour les opérations de substitution comme une clé secrète. Et pour cela, la succession des bases nucléotide et la capacité de stockage de la molécule ADN, ont été explorées.

L'ADN est un alphabet de 4 lettres et tout type d'information peut être écrit –ou plus exactement, existe– dans l'ADN en utilisant cet alphabet. Un schéma de correspondance entre l'alphabet ADN 4 lettres et l'alphabet binaire 2 lettres a été établi comme table de conversion.

Un brin ADN suffisamment long, peu contenir n'importe quel texte, comme si le texte est déjà tatoué dans l'ADN, il suffi de trouver une corrélation entre la succession des lettres et les bases nucléotide. C'est le principe de notre 1ère contribution STEGANO-ADN.

Aussi Dans le même chapitre une autre contribution à été présentée. Inspiré du chiffre de Vernam, qui stipule que la clé doit être aléatoire, aussi longue que le texte et à usage unique. Dans cette méthode, il n'est pas nécessaire de générer une clé aléatoire aussi longue que le texte, il suffit de générer des chiffres correspondant à des blocs de séquences ADN de 32 bases. De ce fait, la clé (OTP) est 32 fois inferieur à la longueur du texte claire et c'est un grand avantage pour la transmission de la clé de chiffrement.

Sauf que les fonctions mathématiques peuvent générer des nombres pseudo-aléatoires à partir d'une graine. La génération de nombres purement aléatoires est une tâche difficile pour les ordinateurs. C'est pour cela que certains événements naturels (température, humidité, bruit …) sont utilisés comme sources.

Le temps de calcul a été estimé par une analyse pratique. Basé sur les mesures obtenues et une vue graphique de différente variation a été établi pour chaque opération.

La sécurité des deux algorithmes, a été mesurée. Une analyse théorique des attaques de cryptanalyse par force brute (espace de clé) a été réalisée. Une comparaison à un autre algorithme d'un principe similaire a été réalisée en utilisant la théorie de la complexité.

Une description détaillée d'un exemple d'une façon visuel à été présentée et une étude de la longueur des séquences d'ADN a été effectuée en raison de son influence sur la vitesse du chiffrement et du déchiffrement.

REFERENCES BIBLIOGRAPHIQUE

[AD58] Adolphe Delahays « L'art d'écrire en chiffre » Libraire Editeur, Rue Voltaire. Paris 1858.

[ADL94] Leonard Adlman, « Molecular computation of solution to combinatorial problems » Science : 266 : 1021-1024, 1994.

[ADL98] Leonard Adlman, « Calculer avec de l'ADN » : informatique et biologie pour la science N° 252 » octobre 1998.

[ANT03] Hélène Antaya, Isabelle Ascach-Coalilier, « l'ordinateur à l'ADN ». License informatique 202-2003. Université de Nice Sophia-Antipolis.

[ANT11] M. Antonini, « Techniques de compression pour le codage des images et des vidéos », lecture notes, 2011.

[AUG83] Auguste Kerckhoffs «La cryptographie militaire, Journal des sciences militaire», Vol IX pp 5-83, Janv. 1983, pp 161-191, Fev 1983

[BDH04] S. Block, D. Donoho, T. Hwa, et al., « DNA Barcodes and Watermarks », *MITRE Corporation*, 2004.

[BLE10] G. E. Blelloch, « Introduction to Data Compression », Carnegie Mellon University, 2010.

[BOC09] I. Bocharova, « Compression for Multimedia », *Cambridge University Press*, 2009.

[Bor11] M. Borda, « Fundamentals in Information Theory and Coding », *Springer*, May 2011.

[BTT13] M. Borda, O. Tornea, R. Terebes, R. Malutan, « Method and cryptographic OTP system based on DNA random sequences », Gold Medal for Patent request at ProInvent 2013, No. of patent application: A10003/14.02.2013.

[CL00] H. Cheng, X. Li, « Partial encryption of compressed images and videos », IEEE Transactions on Signal Processing, Vol. 48(8), pp. 2439–2451, 2000.

[DAN10] Daniel Barsky & Ghislain Dartois, « Cours de cryptographie » 2010.

[DBV03] J. Dolezel, J. Bartos, H. Voglmayr, J. Greilhuber, « Nuclear DNA content and genome size of trout and human », *Cytometry, Wiley-Liss, Inc.*, Vol. 51, No. 2,pp. 127–128, 2003.

[DEN07] Tom St Denis, « Cryptography for Developers », Syngress Publishing, Inc., 2007.

[DH76] W. Diffie, and M. Hellman, « New Directions in Cryptography. » Proceedings of the AFIPS National Computer Conference, June 1976.

[DUO05] Duong Hieu Phan, « Sécurité et efficacité des schémas cryptographiques », Doctorat de l'école polytechnique, Ecole normale supérieur de l'informatique, Paris-France, 2005.

[ElG85] T. ElGamal, « A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms », *IEEE Transactions on Information Theory*, Vol. 31 (4), pp. 469–472. 1985.

[FIPS01] FIPS 197, Advanced Encryption Standard (AES), 2001.

[FIPS93] FIPS 46-2, Data Encryption Standard, 1993.

[FIPS94] « Digital Signature Standard (DSS) », *Federal Information Processing Standards Publication 186*, May 1994.

[GEH99] Ashish Gehani, Tomas H. LaBean, John H. Reif, « DNA based cryptography » 5th annual DIMACS meeting on DNA based computers (DNA 5), MIT Cambridge, MA, June 1999.

[GG10] Carine Giovannangeli « Cibler l'ADN : pour la compréhension du vivant » EDP Sciences, 2010,

[GG91] A. Gersho, R.M. Gray, « Vector Quantization and Signal Compression », *Kluwer Academic Publishers*, 1991.

[GLR04] A. Gehani, T. LaBean, J. Reif, « DNA-Based Cryptography », *Springer*, Vol. 2950, pp. 167-188, 2004.

[HAS88] J. Hastad, « Lower bounds in computational complexity theory », *Notices of the AMS*, Vol. 35, No 5, pp. 677–683, 1988.

[KNU94] L.R. Knudsen, « Block Ciphers - Analysis, Design, Applications, » Ph.D. dissertation, Aarhus University, Nov 1994.

[KOH08] David R. Kohel, « Cryptography » 11th July, 2008.

[LAM02] Constanza Lampasona, « DNA Computers Applications Cryptography », Innovative Computer Architecture and Concepts Computer Architecture. University of Stuttgart June 2002.

[LIA08] S. Lian, « Multimedia Content Encryption: techniques and applications », *CRC Press*, 2008.

[LM91] X. Lai, and J. L. Massey, « A Proposal for a New Block Encryption Standard », *Springer-Verlag, Lecture Notes on Computer Science (LNCS)*, Vol. 473, pp. 389–404, 1991.

[MEL01] H.X.Mel, Doris Baker, « La cryptographie décryptée ». Compus Press 2001.

[MOV96] A. Menezes, P. Van Oorschot, S. Vanstone, « Handbook of applied cryptography », *CRC Press*, 1996.

[MSTM06] Dr. Makri-Mokrane Samira, Pr. Tazir Meriem. Département de médecine. « Introduction à la génétique moléculaire », Année 2006.

[NAS14] Nasser Yassine, Ouyous Mina « Rapport sur l'étude et l'implémentation de quelques algorithmes de chiffrement et de signature ». Université Mohamed V, AGDAL 2013-2014

[NIST01] D.P. Leech, M.W. Chinworth, « The Economic Impacts of NIST's Data Encryption Standard (DES) Program », *TASC, Inc.*, October 2001.

[OT13] Olga Tornea. « Contributions to DNA cryptography : applications to text and image secure transmission ». Université Nice Sophia Antipolis; Technical University of Cluj-Napoca (Roumanie), 2013.

[PAR03] C. Parisot, « Model-Based allocations and scan-based discrete wavelet transform for image and video coding », PhD Thesis, *University of Nice-Sophia Antipolis*, 2003.

[PGP04] PGP Corporation, Phil Zimmermann, « An introduction to cryptography », 2004.

[REN07] Renaud Dument, « Introduction à la cryptographie et à la sécurité informatique ». Université de Liège, faculté des sciences appliquées, 2007.

[RJ91] M. Rabbani, P. W. Jones, « Digital Image Compression Techniques », Tutorial texts in optical engineering, Vol TT7, *SPIE Press*, 1991.

[RN88] J.L. Rodgers, W.A. Nicewander, « Thirteen ways to look at the correlation coefficient »,*The American Statistician*, Vol. 42, No. 1, pp. 59 – 66, 1988.

[RS10] O.S. Rao, S.P. Setty, « Efficient mapping methods for Elliptic Curve Cryptosystems », *International Journal of engineering Science and Technology*.Vol. 2, No. 8, pp. 3651-3656, 2010.

[RSA78] R. Rivest, A. Shamir, and L. Adleman, « A Method for Obtaining Digital Signatures and Public-Key Cryptosystems », *Communications of the ACM*, Vol. 21 Nr. 2 February 1978.

[RUM09] A. H. Rumpf, « A Brief Introduction to Data Compression and Information

Theory », *Ripon College Summation*, pp. 15-18, 2009.

[SAY03] K. Sayood, « Data Compression », Encyclopedia of Information Systems, Vol. 1, pp. 423 – 444, *Elsevier Science*, 2003.

[SB98] C. Shi, B. Bhargava, « A fast MPEG video encryption algorithm », In *Proceedings 6th ACM International Multimedia Conference*, pp. 81–88, 1998.

[SCH02] B. Schneier, « Crypto-Gram Newsletter », *Counterpane Internet Security, Inc.*, October 2002.

[SCH09] B. Schneier, « The History of One-Time Pads and the Origins of Sigaba », Blog post, 2009.

[SCH96] B. Schneier, « Applied Cryptography: Protocols, Algorithms, and Source Code in C », *John Wiley & Sons Inc.*, 1996.

[SEE81] N.C. Seeman, « Nucleic Acid Junctions: Building Blocks for Genetic Engineering in Three Dimensions »,*R.H. Sarma, Adenine Press*, pp. 269-277, 1981.

[SFP02] B. Shimanovsky, J. Feng, M. Potkonjak, « Hiding Data in DNA », *Proceeding IH '02 International Workshop onInformation Hiding*, pp. 373-386, 2002.

[SHA48] C.E. Shannon, « A Mathematical Theory of Communication », *Bell System Technical Journal*, Vol. 27, No. 3,pp. 379–423, 1948.

[SHA49] C.E. Shannon, « Communication Theory of Secrecy Systems », *Bell System Technical*

[SK05] P. Salama, B. King, « Efficient secure image transmission: compression integrated with encryption », *Proc. SPIE*, Vol. 5681, pp. 47-58, 2005.

[STA11] W. Stallings, « Cryptography and Network Security: Principles and Practice », (5th Ed.), *Prentice Hall*, 2011.

[TB09] O. Tornea and M. E. Borda, « DNA Cryptographic Algorithms », *IFMBE Proc.*, Vol. 26, pp. 223-226, 2009.

[TRB99] C. T. Taylor, V. Risca, and C. Bancroft, « Hiding messages in DNA microdots », *Nature*, Vol. 399, pp. 533-534, 1999.

[VER26] G. S. Vernam, « Cipher Printing Telegraph Systems », *Journal of* the *American Institute of Electrical Engineers*, Vol. XI.V, pp. 109-115, 1926.

[WC53] J. Watson, F. Crick, « Molecular structure of nucleic acids; a structure for deoxyribose nucleic acid », *Nature*, Vol. 171, No. 4356, pp. 737–738, 1953.

[YOU06] W.R. Yount, « Research Design and Statistical Analysisin Christian Ministry » (chapter 22: Correlation Coefficients), 4th ed.*Napce Organisation*, 2006.

[ZL78] J. Ziv and A. Lempel, « Compression of Individual Sequences via Variable-Rate Coding », *IEEE Transactions on Information Theory*, Vol. 24, No. 5, pp. 530–536, 1978.

WEBOGRAPHIQUE

[IHGM] Institute of Human Genetics, Munich - http://ihg.gsf.de/ihg/databases.html
Consulté : le 06/01/2016 à 22:30

[NET1] www.phidel.fr/environnement.php
Consulté : le 12/04/2016 à 23:58

[Sanger] www.sanger.ac.uk
Consulté : le 19/02/2016 à 10:49

[wncbi] http://www.ncbi.nlm.nih.gov/genbank
Consulté : le 28/12/2016 à 21:13

LISTE DES FIGURES

LISTE DES TABLEAUX