REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE HAOUARI BOUMEDIENNE

FACULTE DE MATHEMATIQUES

DEPARTEMENT D'ALGEBRE ET THEORIE DES NOMBRES

Mémoire de fin d'études

Pour l'obtention du diplôme de PGS en cryptologie

THEME

Cryptanalyse d'un crypto-système basé sur le problème SAT

Présenté par :

Mr Belouad Mohamed

Soutenu le: 24 Mars 2004

Devant le jury:

Mr Aï ssani Amar (Président).

Mme Drias Habiba (Directeur de Thèse).

Mr Betina Kamel (Examinateur).

Mr Faci Hocine (Examinateur).

Mr Marmoul Atef (Examinateur).

Dédicaces.

Avec l'aide de Dieu le tout puissant, j'ai pu achevé ce modeste travail que je tiens à dédier solennellement à ma très chère femme Nacera.

A mes chères parents en témoignage de leur amour, du respect et de la gratitude que je leurs porte et à ma grand mère Meriem.

A mes frères et sœurs.

A toute mes familles ;

-Belouad.

-Terchi.

-Benaouali.

-Qadiri.

-Sassi.

-Gharbi.

A tous mes amis.

Mohamed.

Remerciements.

Mes remerciements s'adressent tout d'abord à Madame Drias Habiba, ma directrice du projet, pour l'assistance qu'elle m'a prêtée afin de mener à mieux ce projet.

Aux hauts responsables de notre institution, qui m'ont donné cette opportunité de formation.

Aux hauts responsables de la DGSCT, qui grâce à eux a été mené à bien cette formation.

A tous les professeurs, qui ont veillé à nous enseigner leur savoir tout au long de notre formation.

A tous mes collègues du PGS.

Enfin, sans oublier mes amis, qui de près ou de loin m'ont aidé dans mon travail.

Mes remerciements solennels à mes beaux parents.

Abstract

Cryptanalyse of a crypto-system based on problem SAT

The genetic algorithms constitute an interesting alternative to the traditional techniques for optimization problem solving. In this work we present a cryptanalyse of a crypto-system based on a function with a single direction, which transforms a 2-SAT instance into a 3-SAT data. The cryptanalyse consists in solving the 3-SAT problem. Since the latter is NP-Complete, the approach of the genetic algorithms is used for that purpose.

Key Word : Cryptanalyse, 3-SAT, NP-Complet, Genetic Algorithms.

Résumé

Cryptanalyse d'un crypto-systéme base sur le problème SAT

Les algorithmes génétiques constituent une alternative intéressante aux techniques classiques pour la résolution des problèmes d'optimisation. Nous présentons dans ce travail la cryptanalyse d'un crypto-système basé sur une fonction à sens unique, qui transforme une donnée 2-SAT en une donnée 3-SAT. La cryptanalyse revient à résoudre le problème 3-SAT. Puisque 3-SAT est NP-Complet, nous optons pour la résolution du problème 3-SAT par l'approche des algorithmes génétiques.

Mots Clés : Cryptanalyse, 3-SAT, NP-Complet, Algorithmes Génétiques.

SOMMAIRE

CHAPITRE II : NOTION SUR LE PROBLEME SAT

CHAPITRE III : LES SYSTEMES CRYPTOGRAPHIQUES ET ATTAQUES DES ALGORITHMES

CHAPITRE VI : ALGORITHMES GENETIQUES

LE PREAMBULE

La cryptographie

C’est l'étude des principes, méthodes et techniques mathématiques reliée aux aspects de sécurité de l'information telles la confidentialité, l'intégralité des données, l’authentification d'entités et l'authentification de l'originalité des données. C'est un ensemble de techniques qui fournit la sécurité de l'information.

La cryptographie vous permet de stocker des informations sensibles ou de les transmettre à travers des réseaux non sûrs (comme Internet) de telle sorte qu'elles ne peuvent être lues par personne à l'exception du destinataire convenu.

La cryptanalyse

Elle étudie la sécurité des procédés de chiffrement utilisés en cryptographie. Elle consiste alors à casser des fonctions cryptographiques existantes, c'est-à-dire à démontrer leur sécurité.

La cryptanalyse mêle une intéressante combinaison de raisonnement analytique, d'application d'outils mathématiques, de découverte de redondances, de patience, de détermination et de chance.

La cryptologie

Elle embrasse à la fois la cryptographie et la cryptanalyse.

Un crypto-système

Il est constitué d'un algorithme cryptographique ainsi que toutes les clés possibles et tous les protocoles qui le font fonctionner.

Le cryptographe

Il a comme travail de fournir des outils pour éliminer les risques, afin de rendre les échanges d'information confidentiels, infalsifiables, authentiques et inaltérables. L'information est chaque jour échangé d'un point à un autre et se trouve susceptible d'être lue, copiée, supprimée, altérée ou falsifiée.

Terminologie

Informatique

Un bit

C’est la valeur « 0 » ou « 1 » (valeurs booléennes).

Un octet

Est constitué de 8 bits.

Cryptographie

Le message clair

C’est le message d'origine (*plaintext*, *cleartext*).

Le chiffrement

C’est la transformation effectuée sur le texte clair (*encryption*).

Le texte chiffré ou cryptogramme

C’est le message transformé par un algorithme de chiffrement (*ciphertext*).

Le déchiffrement

C’est la transformation de reconstitution sur un texte (*decryption*).

Critères d'évaluation

Les algorithmes de chiffrement peuvent être évalués à l'aide de plusieurs mesures, cinq critères principaux ont été relevés.

1- Le Niveau de sécurité

Il est défini comme étant la quantité maximale de travail pour venir à bout de l'objectif.

2- Les fonctionnalités

Elles sont déterminées par les propriétés de base des outils cryptographiques (primitives)

3- Les modes de fonctionnement

Ce sont les possibilités des primitives à agir différemment selon la manière ou les entrées utilisées.

4- La performance

Elle consiste en l’efficacité de la puissance de calcul par rapport au temps, par exemple compter le nombre de bits chiffrés par seconde.

5- La facilité d’implémentation

Elle est définie par la complexité selon l'environnement (logiciel et matériel).

INTRODUCTION GENERALE

Depuis longtemps, la transmission de données sensibles a nécessité l'utilisation d'un système de sécurisation performant.

Les services secrets des grandes puissances économiques et politiques, de tous temps très impliqués, ont développé, tout d'abord, des codages alphabétiques et numériques simples, puis des techniques cryptographiques plus poussées, grâce à l'outil mathématique pour rendre inviolables et inexploitables directement leurs données sensibles.

La cryptologie, véritable science régissant le codage de l'information, a connu une réelle explosion avec le développement des systèmes informatiques, passant d'une première artisanale et confidentielle à des systèmes de très hautes technologies nécessitant une importante puissance de calcul. Elle a connu un plus large essor encore avec l'arrivée des systèmes de communications modernes (Internet, etc...) où il y a une nécessité absolue de protéger les données échangées pour respecter les individus.

Ce travail porte sur le domaine de la cryptanalyse et plus particulièrement sur la cryptanalyse du crypto-système basé sur le problème de satisfiabilité ou SAT en abrégé, proposé par monsieur Ben-Aziz Sid-Ali dans le cadre de la même PGS.

Les crypto-systèmes basés sur le problème de satisfiabilité constituent une nouvelle tendance dans le domaine de la cryptographie. Cet intérêt vient d'une partie de l'étroite relation entre la cryptologie et la complexité, et d'une autre part de pouvoir bénéficier de tous les travaux déjà effectués autour du problème SAT et des problèmes NP-complets en général.

Le crypto-système considéré est basé sur une fonction à sens unique, qui transforme une donnée 2-SAT en une donnée 3-SAT, la cryptanalyse de ce dernier consiste à la résolution du problème 3-SAT, nous optons à l'approche des algorithmes génétiques pour la résolution du problème.

Notre travail a été jalonné en cinq chapitres comme suit :

Dans le premier chapitre, on a présenté un historique succinct sur la cryptologie et les méthodes de cryptologie actuelle, ainsi que la cryptanalyse actuelle.

Le second chapitre est consacré aux notions de base concernant le problème de satisfiabilité (SAT), ainsi que les algorithmes de résolution de ce dernier.

Le troisième chapitre décrira une étude générale sur les crypto-systèmes, ainsi que les différents types d'attaque des algorithmes.

Le quatrième chapitre, portera sur une étude des algorithmes génétiques. Les différents opérateurs génétiques utilisés dans un AG sont décrits, ainsi que le fonctionnement d'un AG, suivi d'un exemple qui le met en évidence. Comme on a abordé les différentes approches théoriques sur les algorithmes génétiques.

Le cinquième et dernier chapitre, repose sur la cryptanalyse du crypto-système existant, basé sur une fonction à sens unique qui transforme une donnée 2-SAT en une donnée 3-SAT, puis que 3-SAT est NP-Complet, la cryptanalyse revient à résoudre le problème 3-SAT, on a choisit l'approche des algorithmes génétiques pour la résolutions du problème.

Chapitre I

Généralités sur la Cryptologie

I.1. Introduction

Le passage de la préhistoire à l'Histoire **[1,2,3,4],** s'est fait dès la création de l'écriture. En effet à partir du moment où l'on a pu conserver les grands faits de l'homme, l'histoire a débuté. Sans l'écriture, la cryptologie n'aurait jamais existé. Cependant son élaboration ne s'est pas faite au hasard du temps. On peut penser comme David Kahn l'a dit : "que ce doit être dès que la culture a atteint un certain niveau, mesuré par sa littérature, que la cryptographie apparaît spontanément (comme ses parents l'écriture et le langage, l'ont probablement fait) ". Les besoins multiples de l'homme demandant la confidentialité entre deux ou plusieurs personnes, au milieu de la société conduit inévitablement à la cryptographie.

Dans le monde actuel, le déchiffrement, opération qui consiste à rétablir le texte clair d'un document chiffré dont on ne connaît pas la clé, est la source la plus importante de renseignements secrets. Les informations qu'il procure, beaucoup plus nombreuses et plus sûres que celles fournies par l'espionnage, exercent une influence sur la politique des gouvernements. Cependant l'édification de cette science qu'est la cryptologie ne s'est pas faite en un jour. Elle regroupe la cryptographie et la cryptanalyse. Le mot "*cryptologie*" du grec *kruptos* (caché) et *graphein* (écrire) peut être assimilé à "étude des écritures secrètes". La cryptographie, c'est l'art de dissimuler ses intentions ou ses instructions à ses ennemis et partant de les transmettre à ses amis au moyen d'un texte chiffré. En face, chez l'adversaire, il s'agit de briser le code, de trouver le système qui préside à son élaboration : c'est la cryptanalyse.

La cryptologie est apparue dans de nombreux domaines tels que l'armée, le commerce, la religion... Elle a donc énormément influencé le cours de l'histoire même si elle est restée dans l'ombre. Mais quels sont les principaux faits historiques que la cryptologie a bouleversé jusqu'à nos jours et par quels moyens ?

Ce chapitre portera sur la cryptologie, un historique succinct sera présenté.

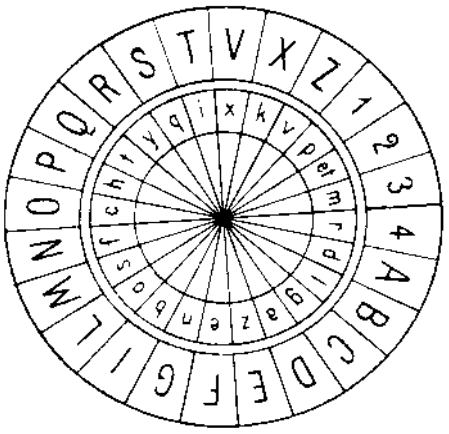
I.2. La naissance de la cryptologie

La cryptographie est une discipline ancienne. Elle a évolué suivant différentes civilisations, l'Egypte,la Chine, l'Inde et la civilisation de l'antiquité, la Grèce.

Polybe, écrivain grec est à l'origine du premier procédé de chiffrement par substitution. C'est le carré de 25 cases :

	1	2	3	4	5
1	a	b	c	d	e
2	f	g	h	ij	k
3	l	m	n	o	p
4	q	r	s	t	u
5	v	w	x	y	z

En 1467, l'italien Leon Batista Alberti inventa la substitution polyalphébetique



Cadran chiffrant d'Alberti

La substitution polyalphabétique évolua encore sous l'impulsion de Giovanni Batista Belaso, qui inventa la notion de clé littérale qu'il appela « mot de passe ».

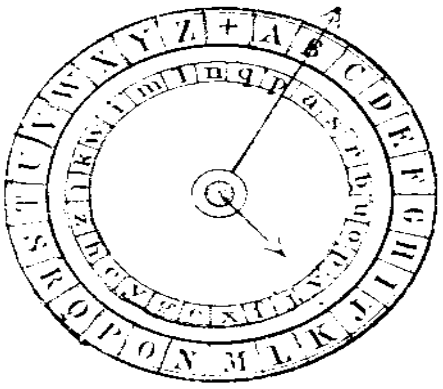
Clé littérale : BEL ASOBELA SOB ELASOB
Texte clair : LES ITALIENS ONT TROUVE

L'inventeur du second procédé est un français du nom de Blaise de vigenère. Il utilise un tableau, c'est le « carré de vigenère »

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Tableau carré, dit « Carré de Vigenère »

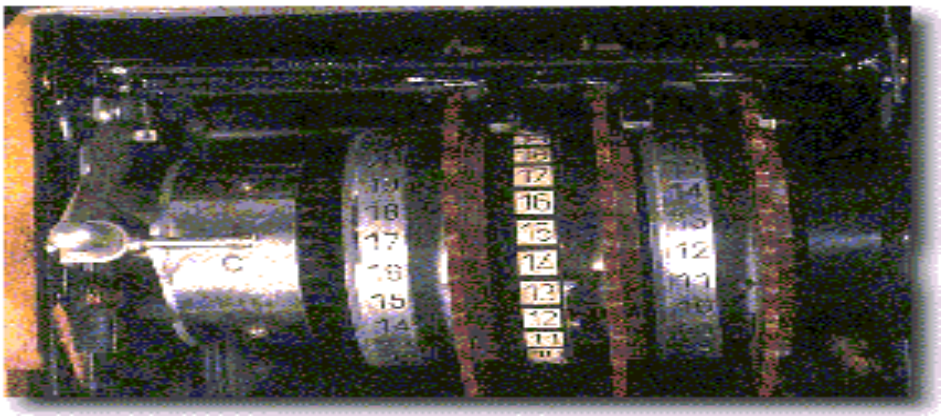
Un savant anglais du nom de Wheastone, enrichissait la cryptographie d'un nouveau procédé. Il s'agissait d'un cryptographe de type Alberti mais avec deux aiguilles semblables à celle d'une montre.



Le cryptographe de Wheatstone (alphabet clair à l'extérieur, cryptographique à l'intérieur)

Pendant les deux guerres mondiales, le savoir en cryptographie et cryptanalyse étaient très important dans le domaine militaire.

En 1923 fût le début de l'histoire de la machine de codage ENIGMA modèle A, par la suite, trois autres modèles apparurent B, C, D.



I.3. La Cryptologie actuelle

I.3.A. Les besoins actuels en cryptographie

De tous temps, les services secrets ont utilisé toutes sortes de codages et de moyens cryptographiques pour communiquer entre agents et gouvernements, de telle sorte que les "ennemis" ne puissent pas comprendre les informations échangées. La cryptologie a alors évolué dans ces milieux fermés qu'étaient les gouvernements, les services secrets et les armées. Ainsi, très peu de gens, voire personne n'utilisait la cryptographie à des fins personnelles. C'est pourquoi, pendant tant d'années, la cryptologie est restée une science discrète.

De nos jours en revanche, il y a de plus en plus d'informations qui doivent rester secrètes ou confidentielles. En effet, les informations échangées par les banques ou les mots de passe ne doivent pas être divulgués et personne ne doit pouvoir les déduire. C'est pourquoi ce genre d'informations est chiffré. L'algorithme de cryptographie DES par exemple, est utilisé massivement par les banques pour garantir la sécurité et la confidentialité des données circulant sur les réseaux bancaires. Le système d'exploitation Unix, lui aussi, utilise ce procédé pour chiffrere ses mots de passe.

Finalement, la cryptologie est de plus en plus utilisée sur le réseau mondial Internet. Avec l'apparition du commerce en ligne, c'est-à-dire la possibilité de commander des produits directement sur Internet, la cryptographie est devenue nécessaire. En effet, si les différents ordinateurs branchés sur Internet sont sécurisés par des mots de passe, c'est-

à-dire à priori inaccessibles par un ennemi, les transactions de données entre deux ordinateurs distants via Internet sont, quant à elles, facilement interceptées. C'est pourquoi lorsque l'on commande un produit sur Internet en payant avec notre carte bancaire, il est beaucoup plus sûr d'envoyer notre numéro de carte bancaire une fois chiffré, celui-ci ne pourra à priori, être déchiffré que par la société à laquelle on a commandé ce produit.

C'est pour ces mêmes raisons d'insécurité sur Internet, et par un besoin humain d'intimité que la cryptographie à des fins purement personnelles s'est développée sur le réseau : pour la messagerie électronique. En effet lorsque l'on envoie un message électronique par Internet, on peut préférer qu'il reste discret vis à vis de la communauté Internet, voire qu'il ne soit compréhensible que par le destinataire du message. En d'autres termes, la cryptographie peut servir si l'on veut envoyer un message confidentiel, ou un message intime à quelqu'un. Cela est aujourd'hui possible grâce à la formidable distribution de logiciels gratuits permettant d'utiliser de la cryptographie "forte" très facilement. C'est le cas du logiciel PGP (Pretty Good Privacy = "assez bonne confidentialité") qui est distribué gratuitement sur Internet, développé par Philip R. Zimmerman seul, en 1991. Ce sont pour toutes ces raisons que tout d'abord la cryptologie s'est énormément renforcée, et que finalement elle est passée d'un monde fermé comme les armées ou les services secrets à un monde ouvert à tout utilisateur.

I.3.B. Les méthodes actuelles de cryptographie

I.3.B.1. Le chiffrement actuel

Le chiffrement est l'action de transformer une information claire, compréhensible par tout le monde, en une information chiffrée, incompréhensible. Le chiffrement est toujours associé au déchiffrement, l'action inverse. Pour ce faire, le chiffrement est opéré avec un algorithme à clé publique ou avec un algorithme à clé privée.

I.3.B.2. Les algorithmes à clé privée ou à clé secrète

Les algorithmes à clé privée sont aussi appelés algorithmes symétriques. En effet, lorsque l'on chiffre une information à l'aide d'un algorithme symétrique avec une clé secrète, le destinataire utilisera la même clé secrète pour déchiffrer. Il est donc nécessaire que les deux interlocuteurs se soient mis d'accord sur une clé privée auparavant, par courrier, par téléphone ou lors d'un entretien privé. La cryptographie à clé publique, quant à elle, a été inventée par Whitfield Diffie et Martin Hetlman en 1976 pour éviter ce problème d'échange de clé secrète préalable.

I.3.B.3. Les algorithmes à clé publique

En effet, les algorithmes à clé publique sont aussi appelés algorithmes asymétriques. C'est à dire que pour chiffrer un message, on utilise la clé publique (connue de tous) du destinataire, qui sera à priori le seul à pouvoir le déchiffrer à l'aide de sa clé privée (connue de lui seul).

I.3.B.4. La préparation au chiffrage

Une information de type texte, ou n'importe quel autre type d'information a besoin d'être codée avant d'être chiffrée à l'aide d'un algorithme à clé publique ou privée. En d'autres termes, il faut fixer une correspondance entre une information et un nombre, puisque les algorithmes à clé (publique ou privée) ne peuvent chiffrer que des nombres. Le problème se résout facilement, puisque la plupart du temps, ce type de cryptographie est essentiellement utilisé sur des machines. Et comme de toute façon les informations sur une machine sont une suite de nombres, le problème est déjà très simplifié.

I.3.B.4.1. La préparation au chiffrage avec DES [5]

L'algorithme DES ne chiffre que des blocs de 64 bits. Il nous suffira donc de diviser nos informations à chiffrer en blocs de 8 octets.

I.3.B.4.2. La préparation au chiffrage avec RSA [6]

L'algorithme RSA, lui, ne chiffre que des nombres inférieurs au nombre *n* qui est un élément de sa clé publique.On pourra utiliser le standard ASCII, plus communément appelé "table ASCII" qui code chaque octet (ou chaque caractère) de 000 à 255, pour transformer partie par partie l'information à chiffrer en nombres (tous inférieurs à n).

I.3.C. L'algorithme DES

I.3.C.1. Histoire de DES

D.E.S., pour Data Encryption Standard ("standard de chiffrage de données"), est un algorithme très répandu à clé privée crée à l'origine par IBM en 1977. Il sert à la cryptographie et l'authentification de données. Il a été jugé si difficile à percer par le gouvernement des Etats-Unis qu'il a été adopté par le ministère de la défense des Etats-Unis qui a contrôlé depuis lors son exportation. DES a été pensé par les chercheurs d'IBM pour satisfaire la demande des banques. Il a été conçu pour être implémenté directement en machine. En effet puisque les étapes de l'algorithme étaient simples, mais nombreuses, il était possible à IBM de créer des processeurs dédiés, capables de chiffrer et de déchiffrer rapidement des données avec l'algorithme DES. Cet algorithme a donc été étudié

intensivement depuis les 15 dernières années et est devenu l'algorithme le mieux connu et le plus utilisé dans le monde à ce jour.

Bien que DES soit très sûr, certaines entreprises préfèrent utiliser le "triple-DES". Le triple-DES n'est rien d'autre que l'algorithme DES appliqué trois fois, avec trois clés privées différentes.

I.3.C.2. Description de l'algorithme DES

L'algorithme DES est un algorithme de cryptographie en bloc. En pratique, il sert à chiffrer une série de blocs de 64 bits (8 octets).

I.3.C.2.1. Le chiffrage avec l'algorithme DES

DES utilise une clé secrète de 56 bits, qu'il transforme en 16 "sous-clés" de 48 bits chacune. Le chiffrage se déroule sur 19 étapes.

1ère étape

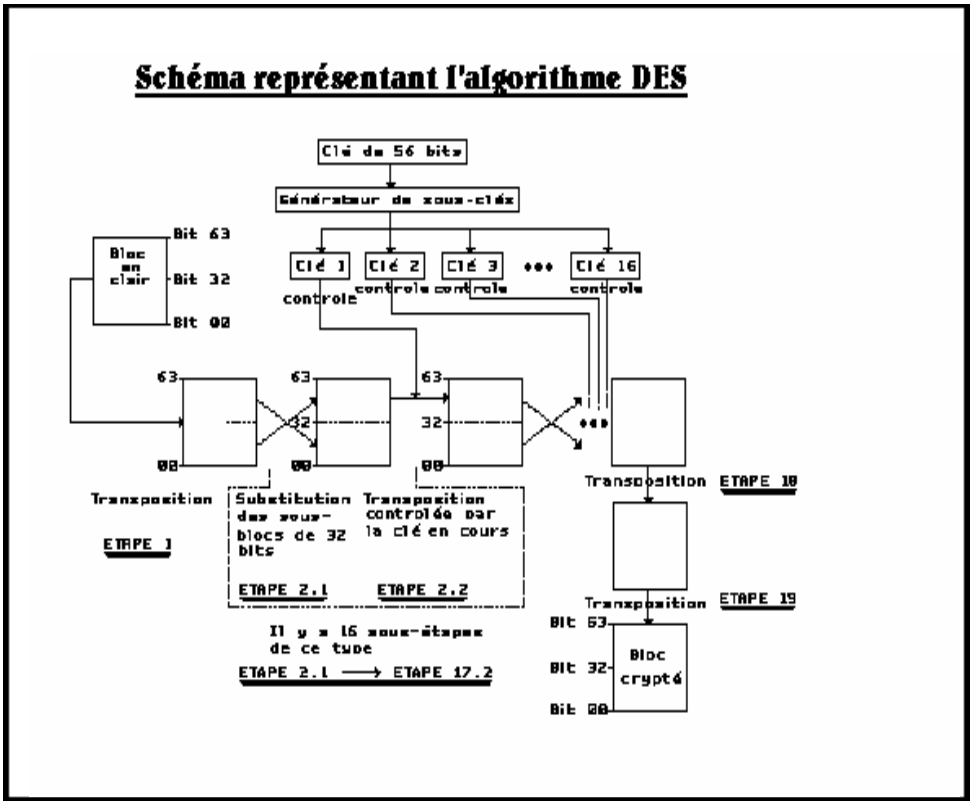
La première étape est une transposition fixe (standard) des 64 bits à chiffrer.

16 étapes suivantes

Les 16 étapes suivantes peuvent être divisées en 2 "sous-étapes" chacune. Dans un premier temps, Le bloc de 64 bits est découpé en 2x32 bits, et une substitution est effectuée entre ces deux blocs, en fait, ces deux blocs seront tout simplement échangés l'un avec l'autre. Dans un second temps, le bloc de 32 bits ayant le poids le plus fort (le bloc qui va du bit n°32 au bit n°63) subira une transposition contrôlée par la sous-clé correspondant à l'étape en cours.

Etape 18 et 19

Les deux dernières étapes sont deux transpositions.



I.3.C.2.2. Le déchiffrage avec l'algorithme DES

Pour déchiffrer un document auparavant chiffré avec DES, il suffit d'effectuer l'algorithme à l'envers avec la bonne clé. En effet, il n'est pas nécessaire d'utiliser un algorithme différent ou une dé différente puisque DES est comme nous l'avons vu un algorithme symétrique. Il est donc totalement et facilement réversible, si l'on possède la clé secrète.

I.3.C.2.3. Les modes opérationnels utilisés avec DES

Comme nous l'avons vu, l'algorithme DES ne permet de chiffrer que des blocs de 64 bits. Pour chiffrer ou déchiffrer un document complet, il faut donc utiliser DES en série dans un "mode opérationnel". Il existe plusieurs modes opérationnels, nous présenterons le mode ECB et le mode CBC.

I.3.C.2.3.a. Le mode opérationnel ECB

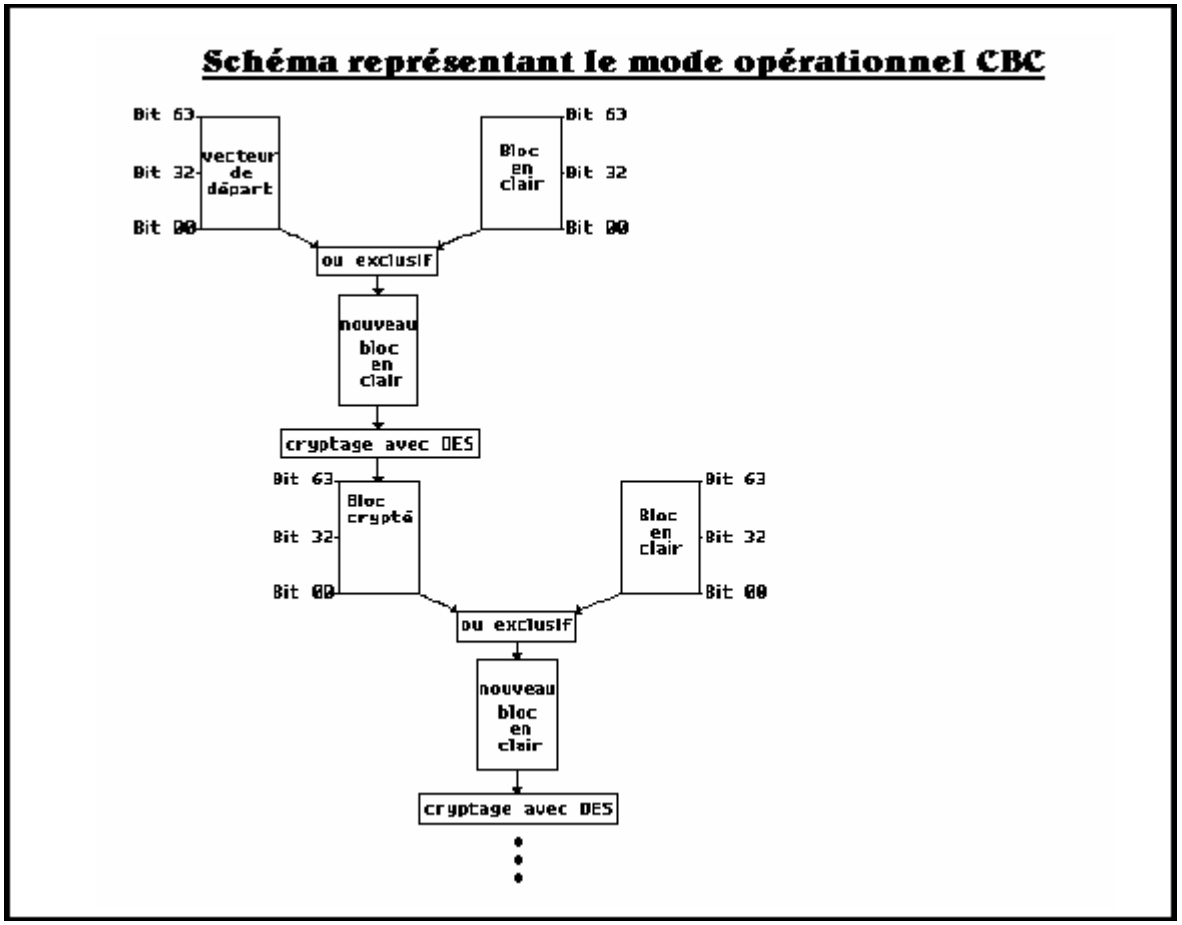
ECB signifie Electronic Code Book ("catalogue électronique de codes"). Dans ce mode, on découpe le document à chiffrer ou à déchiffrer en blocs de 64 bits qu'on chiffre les uns indépendamment des autres. Puisque, à chaque bloc en clair correspond un bloc chiffré, pour une clé donnée, cela peut faire penser à un "catalogue de codes".

I.3.C.2.3.b. Le mode opérationnel CBC

CBC signifie Chain Block Cipher ("Cryptogramme à blocs chaînés"). Comme nous l'avons vu précédemment, le mode opérationnel ECB ne protège pas contre la présence de blocs redondants, puisqu'ils sont chiffrés indépendamment les uns des autres. La seconde faiblesse est qu'un bloc en clair, hors contexte, et codé toujours avec la même clé, produira toujours le même bloc chiffré.

Le CBC lui, répond à ces deux problèmes. Pour ce faire, avant de chiffrer un bloc en clair, on va effectuer un "ou-exclusif" entre ce bloc en clair et le bloc précédemment chiffré. Cela nous donnera un nouveau bloc en clair que l'on chiffrera.

En plus de posséder une clé secrète en commun, les deux interlocuteurs doivent dorénavant se mettre d'accord sur un bloc de 64 bits de départ qu'on appellera "vecteur de départ", ou "vecteur initial".



I.3.C.3. AES [7] : L'AES (Advanced Encryption Standard) est le nouveau standard de chiffrement à clef secrète. Il a été choisi en octobre 2000 parmi les 15 systèmes proposés en réponse à l'appel d'offre lancé par le NIST (National Institute of Standards and Technology). Cet algorithme, initialement appelé RIJNDAEL, a été conçu par deux cryptographes belges, V. Rijmen et J. Daemen. Il opère sur des blocs de message de 128 bits et est disponible pour trois tailles de clef différentes : 128, 192 et 256 bits.

I.3.D. L'algorithme RSA

I.3.D.1. Histoire de RSA

R.S.A. provient des noms Rivest-Shamir-Adleman, en l'honneur à ses inventeurs : Ron Rivest, Adi Shamir et Leonard Adleman qui l'ont inventé en 1977. Le brevet de cet algorithme appartient à la société américaine RSA Data Security, qui fait maintenant partie de Security Dynamics et aux Public Key Parteners, (PKP à Sunnyvale, Californie, Etats-Unis) qui possèdent les droits en général sur les algorithmes à clé publique. RSA est un algorithme à clé publique qui sert aussi bien à la cryptographie de documents, qu'à l'authentification. Grâce au fait qu'il était à clé publique, et au fait qu'il était très sûr, l'algorithme RSA est devenu un standard de facto dans le monde.

I.3.D.2. Description de l'algorithme RSA

Tout le principe de RSA repose sur le fait (qui n'a toujours pas été prouvé) qu'il est très difficile et très long de factoriser un très grand nombre en deux facteurs premiers.

I.3.D.2.1. La génération des clés publiques et privées

Pour commencer, il nous faut choisir deux nombres premiers p et q très grands (de l'ordre de 100 chiffres). Il y a des algorithmes de génération aléatoire de nombres premiers qui existent. Ensuite on trouve le nombre n facilement : $n=p.q$. Ensuite il nous faut trouver un entier e compris entre 2 et $\varphi(n)$. $\varphi(n)$ est la fonction indicatrice d'Euler, c'est en fait le nombre d'entiers inférieurs à n qui sont premiers avec lui, on a $\varphi(n)=(p-1)(q-1)$. $\varphi(n)$ se calcule très facilement ici, puisque l'on a p et q. Maintenant que l'on a n et e, nous sommes prêts à chiffrer. Les nombres n et e forment ici notre clé publique que l'on notera $[n,e]$. Il nous faut calculer le nombre d qui sera nécessaire au déchiffrage. Selon la théorie de RSA, nous devons avoir d tel que $(e.d-1)$ soit divisible par $\varphi(n)$. Pour trouver d nous devons alors résoudre l'équation diophantienne $d+k.\varphi(n)=1$ à l'aide de l'arithmétique. Comme e et $\varphi(n)$ sont premiers entre eux, le théorème de Bézout prouve qu'il existe d et k dans $\mathbf{Z}$ tel que $e.d+k.\varphi(n)=1$.

On pourra résoudre l'équation grâce à l'algorithme d'Euclide. Après résolution, on arrivera à une classe de solution de la forme $d=r.\varphi(n)+d_0$ (où r appartient à **Z**) puisque e a été choisi premier avec $\varphi(n)$. L'ensemble des solutions d à l'équation diophantienne $e.d+k.\varphi(n)=1$ est une classe de congruence modulo $\varphi(n)$, il y a donc une unique solution d comprise entre 2 et $\varphi(n)$, donc $d=d_0$. Nous voilà prêts à déchiffrer. Le nombre d est notre clé privée.

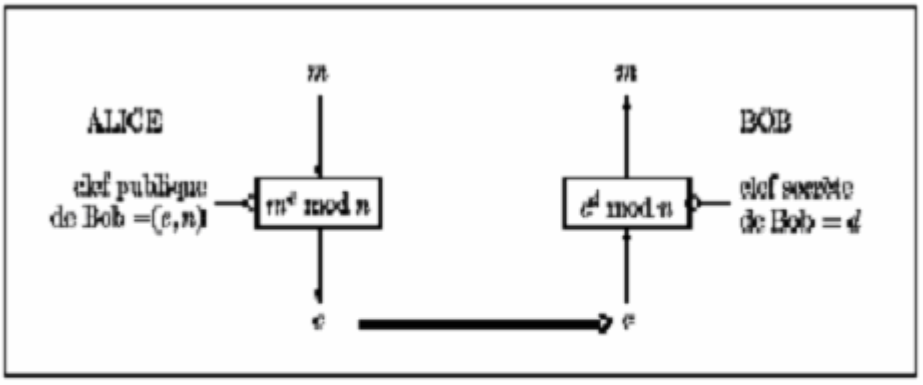
Nous pouvons à présent rendre publique notre clé publique $[n,e]$ et garder secrète notre clé privée. Quant aux nombres p, q, et $\varphi(n)$, on doit, soit les conserver secrets, soit les détruire car ils ne serviront plus.

I.3.D.2.2. Le chiffrage avec l'algorithme RSA

Pour chiffrer un document que l'on aura auparavant transformé en un nombre m inférieur à n il nous faut effectuer l'opération $c=m^e \bmod n$. c est ici notre nombre n une fois chiffré. La première opération peut être très longue à effectuer à la main, l'utilisation d'un ordinateur et d'un programme spécial est fortement conseillée.

I.3.D.2.3. Le déchiffrage avec l'algorithme RSA

Pour déchiffrer un document c, il nous faut effectuer l'opération $m=c^d \bmod n$. m sera bel et bien notre nombre déchiffré, qu'il ne restera plus qu'à retransformer en texte ou en autre chose. La preuve de cet algorithme de chiffrement est faite avec le théorème de Fermat et le théorème chinois des restes connus depuis quelques siècles.



Le chiffrement par RSA

I.3.E. L'authentification de documents

L'authentification d'un document, c'est le fait d'être sûr de l'identité de l'auteur d'un document. Cette authentification peut s'avérer indispensable pour la justice lors d'un litige sur un contrat par exemple. L'authentification se fait toujours sur un contrat papier par une signature manuscrite, à priori infalsifiable. Le problème de l'authentification d'un document "informatique", est l'impossibilité physique d'y apposer une signature manuscrite à sa fin. On va donc y apposer une signature "digitale". Pour ne pas être falsifiable, on va chiffrer cette signature par exemple avec l'algorithme RSA.

I.3.E.1. Les signatures digitales avec RSA

Pour bien prouver qu'un document a été composé par nous, il nous suffira de chiffrer par exemple notre Nom, Prénom et fonction ou n'importe quoi d'autre, avec notre clé privée (en théorie connue de nous seul). Ainsi, quiconque voudra vérifier l'auteur de ce document, n'aura qu'à utiliser notre clé publique pour le déchiffrage. Et si le déchiffrage fonctionne, cela veut bien dire que la signature a été "forgée" avec notre clé privée.

I.3.E.2. Tableau récapitulatif de la gestion des clés avec RSA

Pour ...	on utilise ...	de qui ?
Envoyer un document chiffré à quelqu'un	la clé publique	du destinataire
Envoyer une signature chiffrée à quelqu'un	la clé privée	de l'expéditeur
Déchiffrer un document	la clé privée	du destinataire
Déchiffrer une signature	la clé publique	de l'expéditeur

I.4. La cryptanalyse actuelle

La cryptanalyse est l'étude des procédés de déchiffrage. Où, plus généralement la science qui étudie la sécurité des procédés cryptographiques. Le cryptologue est toujours cryptanalyste puisque qu'il doit en créant un algorithme de cryptographie s'assurer de sa sécurité, et pour ce faire, il a besoin de la cryptanalyse. La cryptanalyse tente de tester la résistance d'un algorithme de cryptographie en simulant différents types "d'attaques", qu'un ennemi pourrait effectuer s'il interceptait le document chiffré. Un ennemi, en cryptologie, est une personne qui tentera, une fois le document chiffré intercepté d'opérer une attaque passive, ou une attaque active.

I.4.A. Les attaques passives

Faire une attaque passive est le fait de tenter de déchiffrer un document dans le but d'en prendre connaissance uniquement, sans l'altérer.

I.4.B. Les attaques actives

Faire une attaque active est le fait de tenter de déchiffrer un document dans le but de pouvoir en prendre connaissance d'une part, et d'autre part dans le but de le modifier, ou d'en modifier la signature pour le falsifier, en général dans son intérêt.

I.4.C. L'attaque d'un document crypté avec DES

La seule méthode connue à ce jour pour décrypter un message crypté avec DES, est la méthode dite "brute" qui consiste à tester la totalité des différentes clés de 56 bits possibles. Le problème majeur est qu'il y en a 2^56, soit exactement 72 057 595 037 927 936 différentes. Cela peut prendre un temps considérable. Cependant, les services secrets peuvent avoir les moyens matériels de briser de tels codes, il leur suffit d'avoir une ou des machines extrêmement puissantes, ce qui pourrait tout à fait être possible pour des nations importantes...

I.4.D. L'attaque d'un document chiffré avec RSA

Comme on l'a vu précédemment, la résistance d'un document chiffré avec l'algorithme RSA s'appuie sur le fait qu'il est extrêmement difficile de factoriser en deux facteurs premiers un très grand nombre. L'attaque va donc consister à utiliser des algorithmes de factorisation les plus rapides, et les plus puissants possibles, pour factoriser le nombre n extrêmement grand de la clé publique visée. L'attaque d'un tel document est encore beaucoup plus longue (pour une taille du nombre n raisonnable) que l'attaque d'un document chiffré avec DES. C'est pourquoi, de grandes recherches en mathématiques sur des algorithmes de factorisation de plus en plus rapides sont effectuées partout dans le monde. La méthode RSA, réputée pour sa quasi-invulnérabilité (quand elle est utilisée avec une très grande clé) pourrait s'écrouler si quelqu'un parvenait un jour à écrire un tel algorithme.

Car RSA repose sur un principe qui a l'air évident mais qui n'a jamais été prouvé. Actuellement, il n'y a aucun algorithme/méthode connu, capable de factoriser dans un temps convenable une très grande clé. Avec les algorithmes de factorisation actuels, il faudrait au briseur de code une puissance beaucoup plus importante pour arriver à ses fins. Mais avec une puissance de calcul plus importante, l'utilisateur peut aussi agrandir la taille

de la clé de un bit ou deux, par exemple. Or l'augmentation de la taille de la clé de un/deux bits signifie une multiplication par deux/quatre le nombre maximum que peut être la clé. Par exemple *RSA Labs* a mis sur le marché il y a quelques mois un processeur dédié à la méthode RSA comportant des instructions dites *"de haut niveau"* directement implémentées sur le processeur, comme une instruction permettant de calculer le modulo d'un grand nombre avec un autre grand nombre rapidement, et une instruction permettant de factoriser un grand nombre. Ce processeur factorise en effet beaucoup plus vite qu'un ordinateur normal, puisque sur l'un, l'algorithme de factorisation est implémenté en *hardware* alors que sur l'autre, il est implémenté en *software*. On peut remarquer que ce processeur avantage plus ou moins également le chiffreur que le briseur de code.

I.5. Le principe de PGP [8]

PGP est de loin le logiciel de cryptographie le plus utilisé dans la communauté Internet par les particuliers. Et ce, parce qu'il est à la fois rapide, très sûr, pratique et gratuit. Philip R. Zimmermann a même eu des problèmes avec le gouvernement des Etats Unis qui a voulu interdire l'exportation de ce produit.

Ce logiciel utilise le principe de cryptographie à clé publique. Alors, avant d'utiliser ce logiciel, vous devrez créer vos clés privées et publiques. Une fois ces deux choses (parfois délicates ...) faites, tout est prêt pour chiffré et déchiffré.

Pour chiffré et déchiffré, PGP utilise deux algorithmes distincts : IDEA (algorithme à clé privé) et RSA.

L'opération de chiffrage se fait donc en deux étapes principales :

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

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

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

C'est la combinaison *algorithme symétrique (IDEA pour chiffrer les données) / algorithme asymétrique (RSA pour chiffré la clé IDEA)* qui confère à PGP sa vitesse et sa grande sécurité.

I.6. Conclusion

La cryptologie a donc connu une rapide évolution à notre époque et c'est en partie dû à deux facteurs essentiels : l'irruption des mathématiques et de l'informatique et le développement des moyens de télécommunication, qui a eu pour effet de multiplier les activités où intervient la cryptologie.

Le rôle de la cryptologie a évolué au fil des siècles. Avant, elle n'avait pour rôle que de protéger un texte écrit ; maintenant, la cryptologie s'étend dans différents domaines tels que la téléphonie, le télétraitement, le stockage des données, communication via satellites...

Depuis une vingtaine d'années, la cryptologie s'est enrichie de nouvelles techniques de chiffrement électronique. Il est vraisemblable que la cryptologie ne disparaîtra pas du fait des nouvelles techniques. Si elle devait disparaître, ça ne pourrait être que par suite d'une nouvelle conception des rapports humains. Le chiffre, lui non plus, n'est pas prêt de disparaître puisqu'il a été et reste encore le moyen le plus sérieux d'assurer la sécurité des correspondances. Il tend de plus en plus vers une structure mathématique.

Il y a aujourd'hui une perpétuelle remise en cause de la cryptologie par la cryptanalyse. C'est une sorte de véritable combat. Les progrès de la cryptanalyse entraînent nécessairement des progrès en cryptologie et vice-versa. C'est une évolution sans fin. On peut se demander si la confidentialité des messages ne se trouve pas alternée dans ces progressions.

Chapitre II

NOTION SUR LE PROBLEME SAT

II.1. Introduction

Le problème **SAT**, qui est le premier problème NP-Complet, démontré par Stephen COOK **[9]**, admet aujourd'hui de plus en plus d'applications, notamment dans la résolution d'encodages de problèmes « difficiles » réels, comme par exemple la cryptanalyse, ou encore la vérification de protocoles et de circuits. Les solveurs actuels de plus en plus performants et gérant les spécificités des problèmes réels nous semblent adaptés pour résoudre des problèmes comme le problème de la cryptanalyse « à la Masscci », qui consiste en un encodage du problème de cryptanalyse en problème **SAT**, ou comme les problèmes issus du Bounded Model Checking qui permettraient de vérifier des circuits liés à la cryptographie. Cependant la seule approche complète du problème **SAT** incarnée par la procédure de DAVID et PUTNAM **[10.11]** reste encore très coûteuse en temps processeur.

Avant de se lancer dans la définition du problème SAT, quelques définitions issues de la logique mathématique paraissent nécessaires pour la compréhension et seront utilisées ultérieurement :

- **Connecteurs :** On appelle connecteurs propositionnels, les symboles suivants :

  Ã : Et exclusif, **Å** : Où exclusif, **« :**Equivalence,

  ® : Implication, **Ù :** Conjonction, **Ú :** Disjonction.
- **Atomes :** Sont le énoncés dont on ne connaît pas (et on ne cherche pas à connaître) les structures internes. On les appelle aussi **variables propositionnelles.**
- **Littéral :** Un littéral est un atome (littéral positif), ou la négation d'un atome.
- **Formule booléenne :** On appelle formule booléenne, toute formule construite à partir des opérateurs logiques (**, « , ®, Ù, Ú**) et des variables propositionnelles. Nous avons les définitions suivantes :
  - Les atomes sont des formules.
  - Si α et β sont des formules alors (α **®** β), (α **«** β), (α **Ú** β) et (α **Ù** β) sont des formules.
  - Si α est une formule alors α est une formule.
  - α **Ù** α : est une formule qu'on note $\varnothing$ appelée clause vide.
- **Clause :** Une clause est une disjonction de littéraux $X_1 \vee X_2 \vee \ldots \vee X_n$.
- **Forme Normale Conjonctive (FNC) :** C'est une conjonction de clauses :

$$C_1 \wedge C_2 \wedge \ldots \wedge C_m.$$

Instanciation : Soit E = { $X_1, X_2, X_3, ..., X_n$ }un ensemble de variables propositionnelles. On appelle instance de E, une fonction **I** telle que : **I**(E) : {faux ou vrai}.

Théorème : Toute formule booléenne admet une forme normale conjonctive qui lui est logiquement équivalente.

Le problème **SAT**, consiste à trouver une instanciation de X pour satisfaire la formule suivante.

$$\mathbf{F} = \wedge \mathbf{C_i} \quad \text{avec } 1 \quad i \quad m$$
$$\mathbf{C_i} = \vee \mathbf{X_j} \quad \text{avec } 1 \quad j \quad n$$

C_i : Une clause.

X_j : Un littéral.

Une formule booléenne écrite sous FNC est dite satisfiable, s'il existe une instanciation qui met à vrai toutes les clauses la composant.

II.2. Définition

Une instance du problème de satisfaisabilité **SAT** est définie par un ensemble de variables booléennes et un ensemble de clauses formés à partir de ces variables. La question associée est de déterminer s'il existe une affectation de valeurs qui satisfait simultanément toutes les clauses. Le problème de satisfaisabilité appartient à la classe des problèmes NP-Complets.

Parmi les applications de problème **SAT,** la détection des défauts dans les systèmes (de communication, d'ordinateurs,etc..) de grande taille, ainsi qu'a la cryptologie.

II.3. Complexité

La distinction entre les "bons" algorithmes et les "mauvais" peut être faite en exécutant les programmes correspondants sur machine, c'est la complexité empirique **[12]**. Une meilleure façon de faire cette distinction est la complexité théorique, elle est définie comme une fonction mathématique qui permet de donner le temps d'exécution des algorithmes en déterminant le nombre maximum d'instructions exécutées en fonction de la taille de la donnée du problème.

Un algorithme dont la fonction de complexité peut être bornée par un polynôme est un algorithme à temps polynomial, sinon il est dit à temps exponentiel.

II.4. Classification des problèmes

On peut classer les problèmes de décision en trois catégories différentes et ceci selon leurs niveaux de difficulté :

1. Problèmes polynomiaux.

2. Problèmes non polynomiaux.

3. Problèmes NP-Complet.

II.4.1. Les problèmes polynomiaux

Un problème est dit polynomial s'il existe un algorithme permettant de trouver une solution optimale pour toutes ses instances en un temps polynomial par rapport à la taille de l'instance. Un tel algorithme est dit *efficace* pour le problème en question.

Il existe une sous-classe des problèmes polynomiaux qui sont les problèmes linéaires (**P** est une fonction linéaire), qui constitue une classe de problèmes plus simple que ceux de **P**, appartiennent à cette classe des problèmes triviaux comme le calcul de la somme de deux nombres de *p* chiffres…

Exemples de problèmes polynomiaux

La classe polynomiale comprend quelques problèmes classiques dont voici des exemples :

a. Tri d'un ensemble de *n* nombres;
b. Recherche des composants connexes d'un graphe ;
c. Recherche d'une chaîne qui passe par toutes les arêtes d'un graphe.

La classe des problèmes polynomiaux est à la fois importante et limitée. Importante parce que nombre de problèmes pratiques indispensables au bon fonctionnement de l'informatique ont des solutions polynomiales, limitées parce que pour nombre de problèmes intéressants, personne n'a réussi à montrer leur caractère polynomial.

II.4.2. Les problèmes NP

Il reste tout un ensemble de problèmes que nous ne pouvons pas classer dans P, car on ne connaît pas d'algorithme polynomial permettant de les résoudre, et que nous ne pouvons pas non plus les considérer comme intraitables, car on n'a pas prouvé qu'il n'existe pas d'algorithme polynomial permettant de les résoudre. L'introduction de la classe NP permet de pallier en partie à l'absence de ces résultats.

II.4.3. Les problèmes NP-Complets

Un problème NP-Complet est un problème dont l'algorithme exact de résolution (qui procède par énumération de toutes les solutions possibles) prend un temps exponentiel par rapport à la taille des données.

Exemple

La classe des problèmes NP-Complets est très importante et nous ne citerons que quelques exemples considérés comme étant les plus importants ou les plus intéressants:

- Le problème **SAT**.
- Le problème du sac à dos.
- Le problème du voyageur de commerce.
- Planification de tâches indépendantes (ordonnancement).

II.5. La NP-Complétude du problème SAT

Cook a démontré qu'effectivement le problème **SAT** est NP- Complet et a présenté cela sous forme d'un théorème :

Théorème de Cook [9]

La satisfiabilité d'une forme booléenne sous forme normale conjonctive est un problème NP-Complet.

II.6. Algorithmes de résolution [13]

Il existe deux classes d'algorithmes de résolution du problème de satisfiabilité ; les algorithmes complets et les algorithmes incomplets. On appelle algorithme incomplet un algorithme qui ne parcourt pas tout l'espace de recherche, ces algorithmes sont très efficaces, mais il se peut qu'ils n'aboutissent pas à une solution même dans le cas où le problème admet une solution. Un algorithme complet quant à lui est basé sur l'exhaustivité du parcours de l'espace de recherche et fournit donc une réponse exacte dans tous les cas.

Si l'on veut obtenir très rapidement un modèle d'une formule proportionnelle on peut donc tout d'abord rechercher une solution avec un algorithme incomplet, si celui-ci ne donne pas de résultat on pourra alors rechercher une solution avec un algorithme complet, par contre si l'on cherche à montrer qu'une formule est non-satisfiable, les algorithmes incomplets ne nous sont d'aucune utilité, seul un algorithme complet pourra y

parvenir, sachant que pour montrer qu'une formule est non-satisfiable, il est nécessaire de parcourir tout l'espace de recherche.

De manière plus précise les méthodes complètes sont toutes basées sur le principe général du « Frist Fail » qui vise à rechercher une contradiction le plus rapidement possible plutôt alors que les méthodes incompletes sont toutes basées sur un principe général de « Réparation Locale » qui vise à améliorer une fonction objective.

II.6.1. Algorithmes incomplets

Les algorithmes incomplets sont principalement basés sur l'idée de la recherche locale : on part d'un certain point dans l'espace de recherche (une instanciation complète des variables), et on se déplace dans le voisinage du point de départ en essayant toujours d'améliorer au maximum la solution courante, citons par exemple GSAT, Recherche tabou, Recuit Simulé.

II.6.2. Algorithmes complets

- **Présentation générale**

Les algorithmes complets sont basés sur un parcours implicite de tout l'espace de recherche associé à la formule de départ. Dans le cas d'une formule à n variables le parcours explicite reviendrait à évaluer la formule sur les 2^n interprétations possibles (méthode dite de la table de vérité Wittgenstein).

Les principaux algorithmes complets sont bien entendu plus rapides que la simple énumération de toutes les interprétations, mais ils souffrent encore de problèmes de performances (problème en général NP-Complet) face aux algorithmes incomplets dans le cas où le problème est satisfiable. Pour le cas non-satisfiable, ils sont les seuls algorithmes à répondre de manière certaine. Les algorithmes complets les plus courants sont les algorithmes basés sur les travaux fondateurs de David et Putnam, les deux principes de base : Saturation et Simplification.

II.7. P = NP ?

Malgré des recherches intensives, aucune solution polynomiale n'a jusqu'à présent pu être trouvée pour un problème NP-complet.

Notons qu'il suffit de trouver une telle solution pour n'importe lequel des problèmes NP-complets connus pour pouvoir dériver des solutions polynomiales pour tous ces problèmes.

Dans l'état actuel des connaissances, on ne sait donc pas si les problèmes de la classe NP sont polynomiaux dans leur ensemble, ou qui sont intrinsèquement difficiles.

Cette question se résume par la formulation lapidaire: **P = NP ?**

L'opinion généralement partagée est que:

• soit $P \subset NP$, i.e. les problèmes NP-complets ne sont pas tous polynomiaux;

• soit P = NP ? n'est pas décidable .

II.8. Conclusion

Personne n'a réussi à trouver un algorithme polynomial pour résoudre un des problèmes NP-Complet. Si un tel algorithme était trouvé, il pourrait d'office s'appliquer à tous les autres problèmes NP-Complet. Une personne qui ne trouve pas de solution polynomiale à un problème NP-Complet a donc une bonne excuse; personne d'autre n'a réussi. Par ailleurs, personne n'a pu prouver, à l'inverse, qu'une telle solution n'existait pas.

Le fait qu'aucune solution polynomiale n'existe, ne signifie pas pour autant qu'il faille renoncer à trouver des solutions acceptables à ces problèmes. Ceci suggère seulement une approche différente lorsqu'il est établi que le problème étudié est NP-Complet. Il existe souvent des solutions heuristiques qui mènent fréquemment à la solution optimale en un temps raisonnable. Il peut aussi exister des solutions qui mènent toujours rapidement à une solution qui est souvent quasi-optimale. Les humains réussissent souvent assez bien à trouver une solution quasi-optimale à de tels problèmes complexes. Les concepteurs essaient alors de reproduire le raisonnement humain pour atteindre des résultats comparables. Cette approche ne rencontre pas toujours beaucoup de succès. En effet, il ne faut pas oublier de profiter des capacités particulières (rapidité et mémoire) des ordinateurs au profit de la difficile tâche d'analyser le comportement humain pour la solution de tels problèmes.

Chapitre III
LES SYSTEMES CRYPTOGRAPHIQUES ET ATTAQUES DES ALGORITHMES

III.1. Les systèmes cryptographiques [14]

Les systèmes cryptographiques, ou crypto-systèmes, ont pour but la transformation d'un message clair en texte codé.

Il existe plusieurs types de systèmes cryptographiques, du simple au plus complexe, chacun possédant un degré de complexité de déchiffrement. Malheureusement aucun, même le plus sophistiqué, n'est inviolable.

III.1.1. Les systèmes cryptographiques à usage restreint

Les cryptosystèmes à usage restreint font reposer leur sécurité sur la confidentialité de leurs méthodes de chiffrement et de déchiffrement. Cette technique est devenue complètement obsolète avec l'arrivée de réseaux dont le nombre d'utilisateurs est très élevé.

De plus, ce crypto-système est d'une efficacité très limitée, car il est assez aisé de le déchiffrer.

III.1.2. Les systèmes cryptographiques à usage général

L'usage d'une clé rend inutile la connaissance des opérations de chiffrement et de déchiffrement. Chaque utilisateur génère sa propre clé qui, elle-même, génère son propre chiffrement. De cette façon, même les concepteurs du programme de chiffrement ne peuvent déchiffrer le cryptogramme.

Dans des cas précis d'impérieuse confidentialité (informations militaires ou diplomatiques), il est possible d'augmenter la sécurité de ce type de systèmes en conservant secret les algorithmes des créateurs du système. Cette précaution supplémentaire n'apporte qu'une sécurité additionnelle et ne saurait être considérée à elle seule comme un crypto-système. En effet, seule la clé constitue, comme son nom l'indique, le sésame permettant d'aboutir à la solution. C'est ce qu'énonce le principe de Kerckhoff (cryptologue néerlandais du XIXème siècle) : la sécurité doit être mesurée en considérant que, hormis la clé, le cryptanalyste connaît dans le détail (il pourrait en être le concepteur) le système de chiffrement.

Il en découle que la sécurité de tout cryptosystème à usage général est fonction du nombre de clés. En effet, un système à clé unique ou peu nombreuses peut permettre au(x) déchiffreur(s) de "casser" le système en un temps relativement peu élevé (à savoir

financièrement raisonnable) en essayant systématiquement toutes les clés possibles sur le texte codé jusqu'à obtention du message clair.

Il existe, cependant, une exception à ce postulat : un crypto-système à substitution mono-alphabétique simple offre un nombre de clé énorme (26 ! permutations possibles des lettres de l'alphabet, soit des possibilité de l'ordre de 4.10^{26}) dont la recherche exhaustive est quasiment irréalisable. Cependant, en utilisant les variations de fréquence des lettres dans les langues naturelles, il est relativement aisé de cryptanalyser ce type de système cryptographique. Il existera donc des systèmes avec un nombre de clés beaucoup moins important mais qui seront, paradoxalement, beaucoup plus sûrs.

III.1.2.1. Les crypto-systèmes généraux à clé secrète

Dans ce cas particulier, il y a existence d'une clé connue uniquement des personnes désirant échanger des informations confidentielles. Il y a donc une part de l'information secrète (en l'occurrence, la clé) qui doit faire l'objet d'une entente préalable entre les utilisateurs de ce système de communication.

Il apparaît évident que pour des réseaux multi-utilisateurs comme il en existe actuellement, le nombre de clés que chacun devrait connaître est absolument impensable. Avec la prépondérance des grands réseaux, l'utilisation de systèmes à clé publique s'est tout naturellement généralisée.

III.1.2.2. Les crypto-systèmes généraux à clé publique

Ce concept a été créé par Ronald Rivest, Léonard Adleman et Adi Shamir. Il a enfin permis à des individus ne partageant pas de clé secrète d'échanger des informations confidentielles avec une sécurité tout à fait acceptable.

Il repose entièrement sur le principe que la personne qui chiffre un message (l'expéditeur) n'a en aucun cas besoin d'être en mesure de le déchiffrer, seul le destinataire doit en être capable.

Chaque utilisateur choisit une clé personnelle à partir de laquelle seront générés deux algorithmes :

- un algorithme public de chiffrement qui sera diffusé auprès de tous les utilisateurs du système.
- un algorithme privé de déchiffrement qui restera secret.

Ainsi, si un utilisateur A désire envoyer un message codé à un utilisateur B, il codera son message à l'aide de l'algorithme de chiffrement que l'utilisateur B aura rendu

public lors de sa création. A la réception, ce même utilisateur B déchiffrera le message grâce à son algorithme de déchiffrement privé.

Toutes les communications confidentielles entre utilisateurs du système seront réalisées de la même façon :

III.1.3. Les dernières évolutions des crypto-systèmes

Toujours dans le souci d'améliorer la sécurité, l'inviolabilité n'existant pas, de nouveaux systèmes, toujours plus sophistiqués ont été mis en place.

III.1.3.1. Le chiffrement probabiliste

Il s'agit plus d'une intéressante amélioration de la cryptographie à clé publique que d'un crypto-système propre.

L'intervention du chiffrement probabiliste complique le " travail " du déchiffreur en ce sens que deux chiffrements d'un même message clair avec une même clé donne lieu à des textes codés complètement différents.

III.1.3.2. La cryptographie sans clé

Imaginée par Bowen Alpern et Fred Schneider, la cryptographie sans clé repose quant à elle non pas sur la protection du contenu du document, mais sur celle de l'identité de l'expéditeur. Elle part du principe très simple qu'un message militaire hautement confidentiel, par exemple, n'est intéressant pour des espions qu'à partir du moment où ils savent qu'il s'agit d'un message militaire hautement confidentiel.

III.1.3.3. La cryptographie quantique

Elaborée par Charles Bennett et Gilles Brassard, la cryptographie quantique se différencie de manière originale des autres types de systèmes cryptographiques puisque ses méthodes de chiffrement s'appuient, comme son nom l'indique, sur les notions de physique quantique.

III.2. Les attaques des algorithmes [11]

Les différents algorithmes ont des niveaux de sécurité divers, plus ou moins difficiles à casser. Si le coût nécessaire pour casser un algorithme dépasse la valeur de l'information chiffrée, alors cet algorithme est probablement sûr. Si le temps nécessaire pour casser un algorithme est plus long que le temps durant lequel l'information chiffrée doit rester secrète, alors cet algorithme est probablement sûr. On dira 'probablement' car il est toujours possible qu'une avancée soit faite en cryptanalyse. De plus, nous ne connaissons pas tout : les cryptanalystes ont souvent intérêt à garder les résultats de leurs travaux et méthodes. De cette manière, ce qui est connu du public en matière de cryptanalyse n'est très probablement qu'une petite partie de ce que connaissent les gouvernements, les industries. Pour exemple, d'après les cryptographes, le DES est un algorithme que l'on devait pouvoir casser sans problèmes à l'aide de la cryptanalyse différentielle inventée en 1990. On a remarqué que les tables-S du DES contenaient des valeurs choisies de manière à rendre la cryptanalyse différentielle la plus inefficace possible (mais elle est toujours possible). Pourtant le DES a été inventé 13 ans avant la cryptanalyse différentielle.

Il existe un seul algorithme qui soit inconditionnellement sûr : celui du masque jetable (se référer au chapitre dédié à cet algorithme à clé secrète). Etant donné des ressources infinies on ne peut pas le casser, pour peu que le protocole et la clé ont été faits de manière correcte. Tous les autres cryptos-ystèmes sont vulnérables au moins à une attaque exhaustive, c'est-à-dire à une attaque qui essaie toutes les combinaisons de clé jusqu'à trouver un texte qui semble correct.

La cryptographie se préoccupe plus particulièrement de crypto-systèmes invulnérables par calculs. Un algorithme est considéré comme invulnérable par calculs, ou parfois qualifié de fort, s'il ne peut pas être cassé avec les ressources disponibles actuellement et dans un futur n'étant pas inférieur au temps que les données doivent rester confidentielles.

Il existe peu d'algorithmes sûrs, et vouloir en inventer un soi-même est inconsidéré. Vouloir en modifier un est hasardeux. On peut utiliser des méthodes de surchiffrement, en chiffrant en cascade avec des algorithmes et des clés non liées. Le résultat est au moins aussi difficile à casser que le plus résistant des algorithmes utilisés. Si on utilise la même clé, il peut en résulter un affaiblissement général. La cryptographie est capricieuse.

Il n'existe pas de moyens fiables de savoir lors de la conception d'un algorithme, s'il sera

résistant ou non. Une solution raisonnable, est de partir sur quelques bonnes bases connues, concevoir l'algorithme, et de le faire tester par d'autres cryptographes. Les consultants en sécurité ne font qu'exploiter les algorithmes qui existent déjà, et ils ont bien raison.

On peut mesurer la complexité d'une attaque de l'une ou l'autre des manières suivantes :

1- Complexité en information : la quantité d'information nécessaire en entrée pour casser l'algorithme. Par exemple on aura peut-être besoin de connaître quelques mots du texte en clair, ou de connaître le type de fichier chiffré.

2- Complexité en temps : le temps nécessaire pour achever l'attaque. Ceci est aussi appelé effort.

3- Complexité en espace : la quantité de mémoire nécessaire pour l'attaque.

Comme règle de base, la complexité d'une attaque est prise comme le minimum de ces 3 facteurs. Pour certaines attaques il faut jongler entre les 3 complexités : une attaque peut être plus rapide au prix de besoins en mémoire plus importants. Exemple, *pkcrck12* qui est un programme de cryptanalyse de fichiers Zip chiffrés, demande 20 Mo de ram pour tourner. (Logiciel fourni).

Durant ce dernier demi-siècle, il y a eu des progrès phénoménaux en puissance de calcul et il n'y a pas de raison de penser que cela s'arrêtera si tôt. Nombre d'attaques sont très bien adaptées aux machines parallèles : la tâche peut être morcelée en plusieurs petites tâches. Annoncer qu'un algorithme est sûr simplement parce qu'on ne peut pas le casser avec la technologie d'aujourd'hui est hasardeux. Les bons cryptosystèmes sont conçus pour être invulnérables même avec les puissances de calcul prévues d'ici à de nombreuses années.

III.2.1. Attaque exhaustive

L'attaque la plus simple qui soit, est l'attaque exhaustive : elle consiste à essayer toutes les combinaisons d'une clé. Une clé de 32 bits demande donc 2^32 4.2 milliards d'essais. Comme on a autant de chances de trouver la bonne clé au début des essais, qu'à la fin, on devra faire en moyenne 4.2/2 = 2.1 milliards d'essais. Les programmes (ou en jargon " moulinettes") qui utilisent ce procédé sont dits "à force brute". Le terme est assez explicite. Ils ont besoin en entrée du nom du fichier à casser et une expression régulière, par exemple [A-Za-z]*[0-9]2. La moulinette essaie toutes les combinaisons possibles de clés satisfaisant l'expression régulière. Le programme *zipcrk2*

(fourni avec le rapport) fonctionnant sur Pc et qui casse les fichiers chiffrés avec Pkzip fonctionne de cette manière.

III.2.2. Attaque par dictionnaires

Une autre attaque assez courante, et plus fine, est l'attaque par "dictionnaires " : on essaie de retrouver la clé à l'aide de mots courants de la langue, et par combinaison de mots significatifs. En effet, il s'avère qu'un grand nombre d'utilisateurs utilisent des clés facilement mémorisables, autrement dit, des clés ayant un sens, telles 'toto' ou 'Jurassic Park' etc... La plupart des 'moulinettes' que l'on trouve sur l'Internet utilisent cette technique. Les clés sont donc tirées d'une base contenant les mots de la langue par exemple. En plus des mots courants on peut ajouter des expressions, des noms de héros, de films et dessins animés, de chansons, des phrases philosophiques, noms des systèmes informatiques, de programmes, les expressions vulgaires, les syllabes chinoises, les noms des 12 nerfs crâniens, les dialogues de films etc... La moulinette tente de déchiffrer le texte en connaissant l'algorithme et en essayant ces mots comme clés, à l'endroit, à l'envers, en minuscules, majuscules... Le programme PGPCRACK qui tente de casser les textes chiffrés avec l'algorithme IDEA de Pgp utilisé en mode clé secrète, fonctionne ainsi.

III.2.3. Attaque à texte clair connu

L'attaque à texte clair connu est le moyen le plus réaliste pour un cryptanalyste de trouver une clé. Elle suppose qu'il dispose d'un message chiffré et du message clair correspondant ; le rapprochement des deux fournit parfois beaucoup d'éléments. C'est l'art de la cryptanalyse linéaire, inventée par Matsui. Elle est plus récente que la cryptanalyse différentielle, et peut encore connaître de grandes améliorations d'ici les prochaines années. Pour casser les mots de passe Unix - variante DES par crypt(3), ce type de cryptanalyse est tout à fait adéquat, puisque l'on connaît le résultat du chiffrement, et on connaît ce qui à été chiffré (c'est un bloc de 64 bits a 0). On tente alors de déduire la clé par des approximations linéaires entre l'entrée et la sortie. Cela ne fonctionne pas systématiquement, et la méthode est suffisamment compliquée pour ne pas être utilisée par des non-spécialistes.

Mais si l'on connaît déjà le texte en clair, quel est l'intérêt de faire du déchiffrage et de trouver la clé ?

D'une part, si la même clé est utilisée pour d'autres messages, on peut les déchiffrer de manière triviale, avec la clé. D'autre part, il n'est pas indispensable de connaître tout le texte en clair, car notons que l'on peut souvent supposer l'existence de

certaines parties. Par exemple, les musulmans pieux commencent tout ce qu'ils écrivent par "Au nom de Dieu, le Clément, le Miséricordieux", et les fonctionnaires français par "République Française - Ministère de...", etc. Les formules de politesses, les dates, sont d'autres exemples de parties de textes en clairs dont l'existence est presque toujours une hypothèse plausible. Cela est encore d'autant plus intéressant pour le cryptanalyste, que mode de chiffrement utilisé est l'ECB.

III.2.4. Attaque à texte choisi

Les systèmes à clés publiques sont sensibles et vulnérables si l'entropie du massage à chiffrer est faible. En effet, plus **H** est faible, moins il existe de combinaisons de textes. S'il existe peu de combinaisons, le cryptanalyste va tenter de chiffrer tous les n messages possibles, (rappelez-vous que la clé est publique, et que donc il dispose de cette clé). Il compare ensuite ce qu'il a chiffré, avec le texte chiffré à découvrir. Quand il y a identicité, le cryptanalyste a trouvé le message en clair. Dans la pratique, il ne va pas chiffrer tous les n messages possibles, mais des parties de messages que l'on suppose chiffrées. Cela donne déjà des indices très précieux au cryptanalyste. Plus n est petit, plus le travaille est rapide.

La cryptanalyse différentielle, s'attaque avec des textes clairs choisis sur des algorithmes à clé secrète. On utilise 2 blocs de textes clairs, puis on étudie l'évolution des différences, des 2 textes lors du processus de chiffrement à travers les rondes de l'algorithme. Ce type de cryptanalyse a été inventé par Biham (cryptanalyste) et Shamir (cryptographe) en 1990.

III.2.5. Attaque à texte chiffré

On ne dispose que du texte chiffré et de l'algorithme de chiffrement. Une attaque exhaustive ou par dictionnaire est une attaque à texte chiffré.

III.3. Conclusion

Si le déchiffrement est un concept très ancien, ce n'est qu'avec l'intérêt tout particulier que lui ont porté quelques scientifiques depuis le début du siècle qu'il est devenu une science exacte et efficace.

L'arrivée des ordinateurs personnels, des réseaux internes et mondiaux (notamment Internet) n'ont fait qu'accroître cet état de fait et intensifier, encore, les recherches sur le sujet de la part des cryptologues comme des cryptanalystes.

A l'instar des techniques, les courants de pensée ont également évolué : auparavant la cryptologie reposait sur la théorie de Shannon, qui part du principe que le cryptanalyste ne possède pas assez d'informations pour le déchiffrement. L'arrivée sur le marché des ordinateurs et de leur puissance de calcul a fait, inévitablement, évoluer les esprits vers la théorie de la complexité du calcul : on ne considère plus que le calcul est irréalisable mais que le déchiffreur n'aura pas le temps nécessaire pour l'effectuer.

La toute dernière approche de la cryptologie, même si elle ne fait pas encore l'unanimité, fait reposer la sécurité du système non plus sur les mathématiques mais sur les propriétés de la physique quantique. Seul l'avenir dira si cette nouvelle technique avortera dans l'oeuf ou si elle formera les bases de la 3^{éme} génération de systèmes cryptographiques.

Quoiqu'il en soit, même les récentes évolutions n'ont pu rendre aucun crypto-système définitivement inviolable. Si cela devait arriver un jour, d'autres problèmes, inhérents à la nature humaine et sans rapport direct avec le déchiffrage, pointeraient à l'horizon : le vol pur et simple du texte clair ou de la clé, le vandalisme (destruction des documents codés, virus,...), les chevaux de Troie (simulation de déchiffrage), trahison du destinataire,...

La cryptologie n'est donc qu'un maillon dans la lutte contre l'espionnage.

Chapitre VI
Algorithmes Génétiques

VI.1. Introduction

Il arrive très souvent que les sciences et les techniques imitent les mécanismes de la nature. Par exemple, le sonar ou le radar sont inspirés des techniques d'écholocation des dauphins ou des chauve-souris, la texture de la peau de requin a été imitée pour fabriquer des matériaux hydrodynamique, les algorithmes génétiques (AG) proposent de reproduire les mécanismes d'évolution et d'adaptation génétique de la vie pour optimiser des problèmes complexes dont les données peuvent varier au cours du temps.

La vie a besoin en permanence de s'adapter à un nouvel environnement, à un nouveau prédateur, ou à de nouvelles proies. Le vocabulaire et les intuitions des algorithmes génétiques appartiennent en partie au vocabulaire et aux intuitions de la biologie. Ces algorithmes fabriquent des chromosomes qui codent chacun une solution potentielle à un problème donné. A chaque étape (appelée génération), ces chromosomes se combinent, mutent, et sont sélectionnés en fonction de leur qualité à répondre au problème. De même, dans la nature, selon la thèse darwinienne, seule la succession de croisements et de mutations aléatoires suffisent à expliquer l'adaptation des êtres vivants à leur milieu naturel (problème qui serait trop complexe à comprendre dans son ensemble).

Dans les AG, la succession des croisements et des mutations permet d'arriver à une solution, qui sans être nécessairement optimale, peut être très satisfaisante.

Les premiers travaux sur les algorithmes génétiques ont commencé dans les années cinquante lorsque plusieurs biologistes américains ont simulé des structures biologiques sur ordinateur. Puis entre 1960 et 1970, John Holland **[15]**, sur la base des travaux précédents, développa les principes fondamentaux des algorithmes génétiques dans le cadre de l'optimisation mathématique. Malheureusement, les ordinateurs de l'époque n'étaient pas assez puissants pour envisager l'utilisation des algorithmes génétiques sur des problèmes réels de grande taille. L'ouvrage de Goldberg **[16]** qui décrit l'utilisation des algorithmes génétiques dans le cadre de résolution de problèmes concrets a permis de mieux faire connaître ces derniers et a marqué le début d'un nouvel intérêt pour ces techniques.

Les applications des AG sont multiples : la cryptographie, optimisation de fonctions numériques difficiles (discontinues, multimodales, bruitées…), traitement d'image (alignement de photos satellites, reconnaissance de suspects…), optimisation d'emplois du temps, contrôle de systèmes industriels, apprentissage des réseaux de neurones, etc.

Les AG peuvent être utilisés pour contrôler un système évoluant dans le temps (chaîne de production, centrale nucléaire…) car la population peut s'adapter à des conditions changeantes. En particulier, ils supportent bien l'existence de bruit dans la fonction à optimiser. Ils peuvent aussi servir à déterminer la configuration d'énergie minimale d'une molécule ou à modéliser le comportement animal.

VI.2. Principes

VI.2.1. Introduction

Les algorithmes génétiques sont des procédures qui s'inspirent des mécanismes de sélection naturelle et des phénomènes génétiques. Le principe de base consiste à simuler le processus d'évolution naturelle dans un environnement hostile. Ces algorithmes utilisent un vocabulaire similaire à celui de la génétique, cependant, les processus auxquels ils font référence sont beaucoup plus complexes.

On parlera ainsi d'individu dans une population. L'individu est composé d'un ou plusieurs chromosomes. Les chromosomes sont eux-mêmes constitués de gènes qui contiennent les caractères héréditaires de l'individu. Les principes de *sélection*, de *croisement*, de *mutation* introduits dans ce cadre artificiel, s'appuient sur les processus naturels du même nom.

Pour un problème d'optimisation donné, un individu représente un point de l'espace d'état. On lui associe la valeur du critère à optimiser. L'algorithme génère ensuite de façon itérative des populations d'individus sur lesquelles on applique des processus de sélection, de croisement et de mutation. La sélection a pour but de favoriser les meilleurs éléments de la population, tandis que le croisement et la mutation assurent une exploration efficace de l'espace d'état.

VI.2. 2.But

Le but des AG est de déterminer les extrêmes d'une fonction, $f : X \longrightarrow R$ où X est un ensemble quelconque appelé espace de recherche et f est appelée fonction d'adaptation ou fonction d'évaluation ou encore fonction *fitness*. La fonction agit comme une «boite noire»pour l'AG. Aussi des problèmes très complexes peuvent être approchés par programmation génétique sans avoir de compréhension particulière du problème.

VI.2. 3.Gestion d'une population

Initialement l'ensemble des solutions est généré aléatoirement. A partir de cet ensemble la recherche sera lancée. Le fait que les solutions soient dispersées à travers l'espace de recherche constitue un avantage très important qui nous assure le balayage de toutes les régions susceptibles de contenir la solution optimale.

La taille de la population (N) : Elle constitue un des paramètres des AG car le choix de la taille de la population peut pénaliser le temps de traitement (si N est trop grand) comme peut-il pénaliser la diversité (si N est trop petit).

Le taux de renouvellement (G) : Désigne le pourcentage de la population qui doit être renouveler à chaque génération.

Le taux de diversité : C'est une valeur qui nous donne une idée sur la diversité de la population et on s y intéresse plus particulièrement durant l'initialisation car après et au fil des générations elle a tendance à diminuer, chose due à la convergence de la population. Une manière de le calculer consiste à sommer toutes les distances de Hamming entres les individus pris deux à deux.

VI.2. 4.Codage d'une population

Le premier pas dans l'implantation des algorithmes génétiques est de créer une population d'individus initiaux. En effet, les algorithmes génétiques agissent sur une population d'individus, et non pas sur un individu isolé. Par analogie avec la biologie, chaque individu de la population est codé par un *chromosome(ensemble de gènes)* ou *génotype* (Holland, **[15]**). Une population est donc un ensemble de chromosomes. Chaque chromosome code un point de l'espace de recherche. L'efficacité de l'algorithme génétique va donc dépendre du choix du codage d'un chromosome.

Dans l'algorithme canonique de Holland (AGC)**[15]**, un chromosome était représenté sous forme de chaînes de bits contenant toute l'information nécessaire à la description d'un point dans l'espace, ce qui permettait des opérateurs de mutation et de croisements simples. L'inconvénient, c'est que deux individus voisins en terme de distance de Hamming ne codent pas nécessairement des éléments proches dans l'espace de recherche. La solution est d'utiliser un code de Gray.

On aboutit à une structure présentant cinq niveaux d'organisation (figure 1), d'où résulte le comportement complexe des AG :

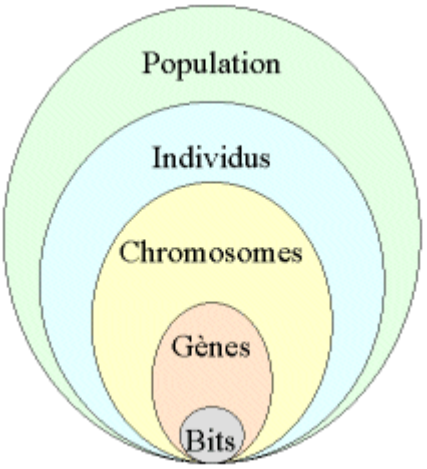
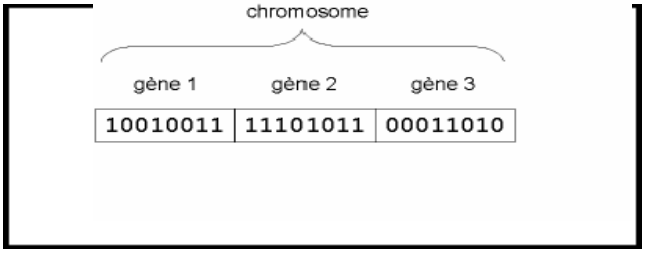


Figure 1: les cinq niveaux d'organisation de notre Algorithme Génétique.

D'un point de vue informatique, nous utilisons dans notre algorithme un codage binaire. C'est-à-dire qu'un gène est un entier long (32 bits).



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

VI.2.5. Fonctions objectif et fonction d'adaptation

La ou les grandeurs à optimiser peuvent être par exemple une consommation, un rendement, un facteur de transmission, un profit, la faisabilité technologique, un coût, une durée de développement, etc. Un algorithme d'optimisation nécessite généralement la définition d'une fonction rendant compte de la pertinence des solutions potentielles, à partir des grandeurs à optimiser. Nous la nommerons *fonction d'adaptation f* (ou *fitness function* en terminologie anglo-saxonne). L'algorithme convergera vers un optimum de cette fonction, quelle que soit sa définition. La pertinence de la solution dépendra donc de la pertinence de la " question " posée à l'ordinateur. La fonction *f* doit donc exprimer le plus fidèlement possible le désir de l'utilisateur sous forme mathématique. C'est une

fonction des variables $x_1,\ldots,x_n$. Sa définition peut être simplement analytique, ou elle peut faire appel à un modèle numérique du dispositif étudié, ou elle peut éventuellement faire appel au jugement de l'utilisateur, etc.

VI.2.5.a. Objectif unique

Dans le cas d'un *objectif unique*, la définition de *f* ne pose généralement pas de problème. Par exemple, si l'on se fixe l'objectif de trouver un dispositif dont le rendement est maximum, *f* sera égale au rendement. Dans le cas où l'on utilise un modèle numérique, on commence par évaluer les caractéristiques des solutions potentielles en utilisant le modèle. Puis on calcule la fonction d'adaptation à partir de ces caractéristiques.

VI.2.5.b. Objectifs multiples

Certains problèmes d'optimisation doivent satisfaire des *objectifs multiples*, souvent concurrents, ce qui implique un compromis. La méthode classique consiste à définir plusieurs *fonctions objectif* f_i, traduisant chaque objectif à atteindre, et à les combiner au sein de la fonction d'adaptation.

D'où on peut définir :

VI.2.5.c. Fonction d'adaptation et fonction *fitness*

Pour calculer le coût d'un point de l'espace de recherche, on utilise une *fonction d'adaptation*. L'adaptation d'un individu ne dépendant pas de celle des autres individus, le résultat fourni par la fonction d'adaptation va permettre de sélectionner ou de refuser un individu pour ne garder que les individus ayant le meilleur coût en fonction de la population courante : c'est le rôle de la fonction *fitness*. Cette méthode permet de s'assurer que les individus performants seront conservés, alors que les individus peu adaptés seront progressivement éliminés de la population.

VI.2.6. La Sélection

La sélection permet d'identifier statistiquement les meilleurs individus d'une population et d'éliminer les mauvais. On trouve dans la littérature un nombre important de principes de sélection plus ou moins adaptés aux problèmes qu'ils traitent.

VI.2.6.1. Les différentes méthodes de Sélection

VI.2.6.1.a. La sélection par roulette (*wheel*)

Les parents sont sélectionnés en fonction de leur performance. Meilleur est le résultat codé par un chromosome, plus grandes sont ses chances d'être sélectionné. Il faut imaginer une sorte de roulette de casino sur laquelle sont placés tous les chromosomes de la population, la place accordée à chacun des chromosomes étant en relation avec sa valeur d'adaptation. Cette roulette est représentée par la figure 2.

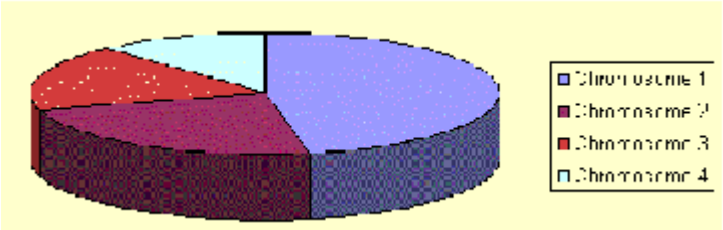


Figure 2: Exemple de sélection par roulette

Ensuite, la bille est lancée et s'arrête sur un chromosome. Les meilleurs chromosomes peuvent ainsi être tirés plusieurs fois et les plus mauvais ne jamais être sélectionnés. Cela peut être simulé par l'algorithme suivant :

1. On calcule la somme $S1$ de toutes les fonctions d'adaptation d'une population.
2. On génère un nombre r entre 0 et $S1$.
3. On calcule ensuite une somme $S2$ des adaptations en s'arrêtant dès que r est dépassé.
4. Le dernier chromosome dont la fonction d'adaptation vient d'être ajoutée est sélectionné.

VI.2.6.1.b. La sélection par rang

La sélection précédente rencontre des problèmes lorsque la valeur d'adaptation des chromosomes varie énormément. Si la meilleure fonction d'adaptation d'un chromosome représente 90% de la roulette, alors les autres chromosomes auront très peu de chance d'être sélectionnés et on arriverait à une stagnation de l'adaptation.

La sélection par rang trie d'abord la population par fitness. Ensuite, chaque chromosome se voit associé un rang en fonction de sa position. Ainsi le plus mauvais chromosome aura le rang 1, le suivant 2, et ainsi de suite jusqu'au meilleur chromosome qui aura le rang N (pour une population de N chromosomes). La sélection par rang d'un chromosome est la même que par roulette, mais les proportions sont en relation avec le rang plutôt qu'avec la valeur de l'adaptation. Le tableau 1 fournit un exemple de sélection par rang. Avec cette méthode de sélection, tous les chromosomes ont une chance d'être sélectionnés. Cependant, elle conduit à une convergence plus lente vers la bonne solution. Ceci est dû au fait que les meilleurs chromosomes ne diffèrent pas énormément des plus mauvais.

Tableau 1: Exemples de sélection par rang pour 6 chromosomes.

Chromosomes	1	2	3	4	5	6	Total
Probabilités initiales	85 %	5 %	1 %	4 %	3 %	2 %	100 %
Rang	6	5	1	4	3	2	21
Probabilités finales	29 %	24 %	5 %	19 %	14 %	9 %	100 %

VI.2.6.1.c. La sélection steady-state

Ce n'est pas une méthode particulière de sélection des chromosomes parents. L'idée principale est qu'une grande partie de la population puisse survivre à la prochaine génération. L'algorithme génétique marche alors de la manière suivante. A chaque génération sont sélectionnés quelques chromosomes (parmi ceux qui ont le meilleur coût) pour créer des chromosomes fils. Ensuite les chromosomes les plus mauvais sont retirés et remplacés par les nouveaux. Le reste de la population survit à la nouvelle génération.

VI.2.6.1.d. La sélection par tournoi

Sur une population de m chromosomes, on forme m paires de chromosomes. Dans les paramètres de l'AG, on détermine une probabilité de victoire du plus fort. Cette probabilité représente la chance qu'a le meilleur chromosome de chaque paire d'être sélectionné. Cette probabilité doit être grande (entre 70% et 100%). A partir des m paires, on détermine ainsi m individus pour la reproduction.

Elitisme

A la création d'une nouvelle population, il y a de grandes chances que les meilleurs chromosomes soient perdus après les opérations de croissement et de mutation. Pour éviter cela, on utilise la méthode d'élitisme. Elle consiste à copier un ou plusieurs des meilleurs chromosomes dans la nouvelle génération. Ensuite, on génère le reste de la population selon l'algorithme de reproduction usuel. Cette méthode améliore considérablement les algorithmes génétiques, car elle permet de ne pas perdre les meilleures solutions.

VI.2.7. Le croisement (*crossover*)

A partir de deux individus, on obtient deux nouveaux individus (enfants) qui héritent de certaines caractéristiques de leurs parents. Le croisement sélectionne des gènes parmi deux individus appelés parents. A partir de ces gènes sont générés les enfants. La probabilité de croisement représente la fréquence à laquelle les croissements sont appliqués.

- S'il n'y a pas de croisement, les fils sont l'exacte copie des parents.
- S'il y a croisement, les fils sont composés d'une partie de chacun de leurs parents.
- Si la probabilité est de 0%, la nouvelle génération est la copie de la précédente.
- Si la probabilité est fixée à 100%, tous les descendants sont générés par croisement.

Le croisement est mis en place pour que les nouveaux chromosomes gardent le meilleur parti des chromosomes anciens. Ceci dans le but d'obtenir, peut-être, de meilleurs chromosomes. Néanmoins, il est quand même important qu'une partie de la population survive à la nouvelle génération.

VI.2.7.1. Les différentes méthodes de croisement

VI.2.7.1.a. Le croisement unipoint [17]

On choisit au hasard un point de croisement, pour chaque couple (Figure 3). Notons que le croisement s’effectue directement au niveau binaire, et non pas au niveau des gènes. Un chromosome peut donc être coupé au milieu d’un gène.

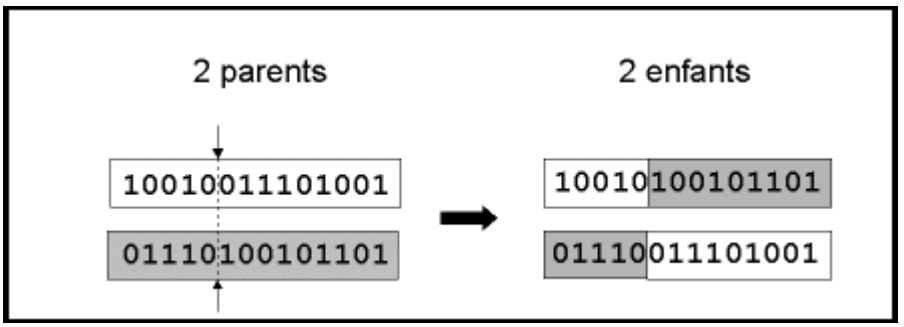


Figure 3 : représentation schématique du croisement unipoint.

Les chromosomes sont bien sûr généralement beaucoup plus longs.

VI.2.7.1.b. Le croisement bipoint [17]

On choisit au hasard deux points de croisement (Figure 4). Par la suite, nous avons utilisé cet opérateur car il est généralement considéré comme plus efficace que le précédent.

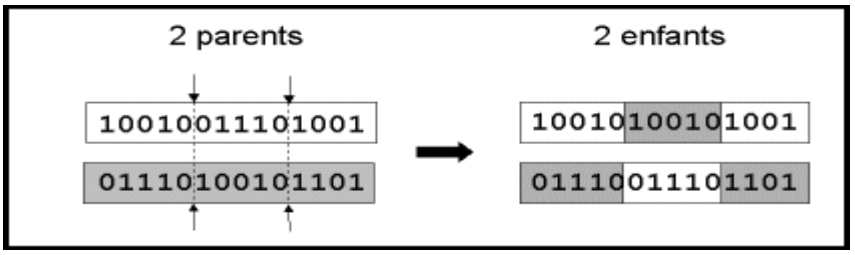


Figure 4 : représentation schématique du croisement bipoints.

VI.2.7.1.c. Le croisement N-point[18]

Ce croisement exige la génération de N point de croisements différents (N<la taille d'un individu -1). Si la partie entre les points P_{i-1} et P_i a été échangée alors la partie entre P_i et P_{i+1} ne le sera pas et ce quelque soit i (0<i<N).

Soit N=4 et les points de croisement générés sont : 3, 4, 6, 8(Figure 5).

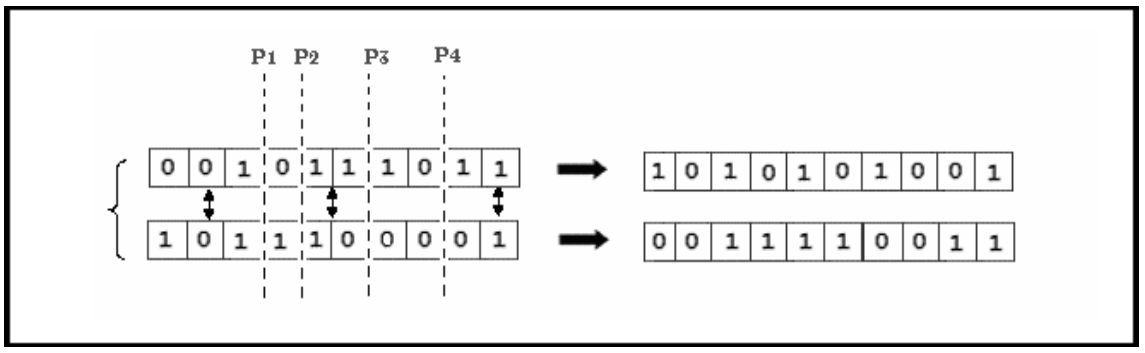


Figure 5: Exemple du croissement N-points (ici 4 points).

VI.2.7.1.d. Le croisement uniforme[18]

Ils existe deux variantes de croisement :

1ère variante :

Le 1er enfant sera produit de la manière suivante :

On prend le 1er gène du 1er parent concaténé au second gène du 2ème parent concaténé au troisième gène du 1er parent ainsi de suite jusqu'à ce que la longueur de l'enfant sera égal à celle des parents. Le 2ème enfant sera construit de la même manière en partant du 2ème parent (Figure 6).

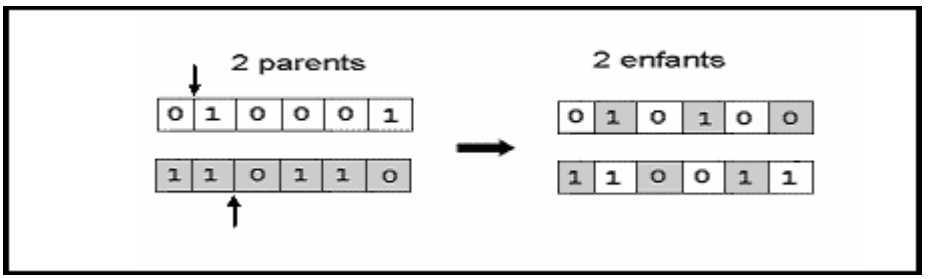


Figure 6 : représentation schématique du croisement uniforme 1ème variante..

2ème variante :

Utilise une chaîne de bits générée aléatoirement et de même longueur que les individus appelés **masque**. Les gènes des individus initiaux sont échangés en fonction de cette chaîne aléatoire lorsque le bit correspondant vaut 0, voire (Figure 7).

Le 1er enfant est conçu de la manière suivante :

Un '1' dans le masque signifie que le gène du 1er parent est copié dans le 1er enfant et un '0' signifie que le gène du 2ème parent est copié dans le 1er enfant.

Le 2ème enfant est conçu de la manière suivante :

Un '1' dans le masque signifie que le gène du 2ème parent est copié dans le 2ème enfant et un '0' signifie que le gène du 1er parent est copié dans le 2ème enfant.

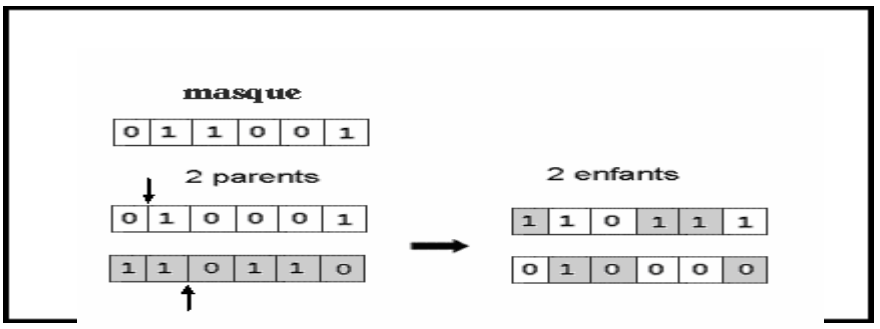


Figure 7 : représentation schématique du croisement uniforme 2ème variante..

VI.2.8. La mutation[19]

La mutation génère des «erreurs»de recopie, afin de créer un nouvel individu qui n'existait pas auparavant. Le but est d'éviter à l'AG de converger vers des extrema locaux de la fonction et de permettre de créer des éléments originaux. Si elle génère un individu plus faible, l'individu est éliminé. La probabilité de mutation représente la fréquence à laquelle les gènes d'un chromosome sont mutés (Figure 8).

- S'il n'y a pas de mutation, le fils est inséré dans la nouvelle population sans changement.
- Si la mutation est appliquée, une partie du chromosome est changée.

La mutation est prévue pour éviter au AG de s'enliser dans des optima locaux. Mais si elle est trop fréquente, le AG est orienté vers une recherche aléatoire de la bonne solution.

Figure 8 : représentation schématique d'une mutation.

VI.2.9. Itération

A partir d'une population initiale de m individus, l'AG sélectionne une population intermédiaire de m individus en faisant une sélection sur la population initiale (un même individu peut être sélectionné plusieurs fois ou peut ne pas être sélectionné du tout, en fonction de la valeur de sa fonction d'adaptation). Les m individus de la population se croisent deux à deux (les couples se forment aléatoirement) pour construire m nouveaux individus. Ces individus passent par un opérateur de mutation (qui agit aléatoirement avec une possibilité faible 2-3% de bits) pour former une nouvelle population. On réitère ensuite le procédé à partir de cette population jusqu'à obtenir une solution que l'on juge satisfaisante.

VI.3. Fonctionnement d'un algorithme génétique.

Un algorithme génétique a la structure suivante :

initialiser le temps;
créer une population initiale;
évaluer l'adaptation de chaque individu;
tant que (il n'y a pas de solution satisfaisante) et (le temps est inférieur au temps limite) faire
    incrémenter le temps;
    sélectionner les parents;
    déterminer les gènes des nouveau nés par recombinaison des gènes parentaux;
        faire subir des mutations aléatoires à la population;
        évaluer l'adaptation de chaque individu;
        sélectionner les survivants;
fait;

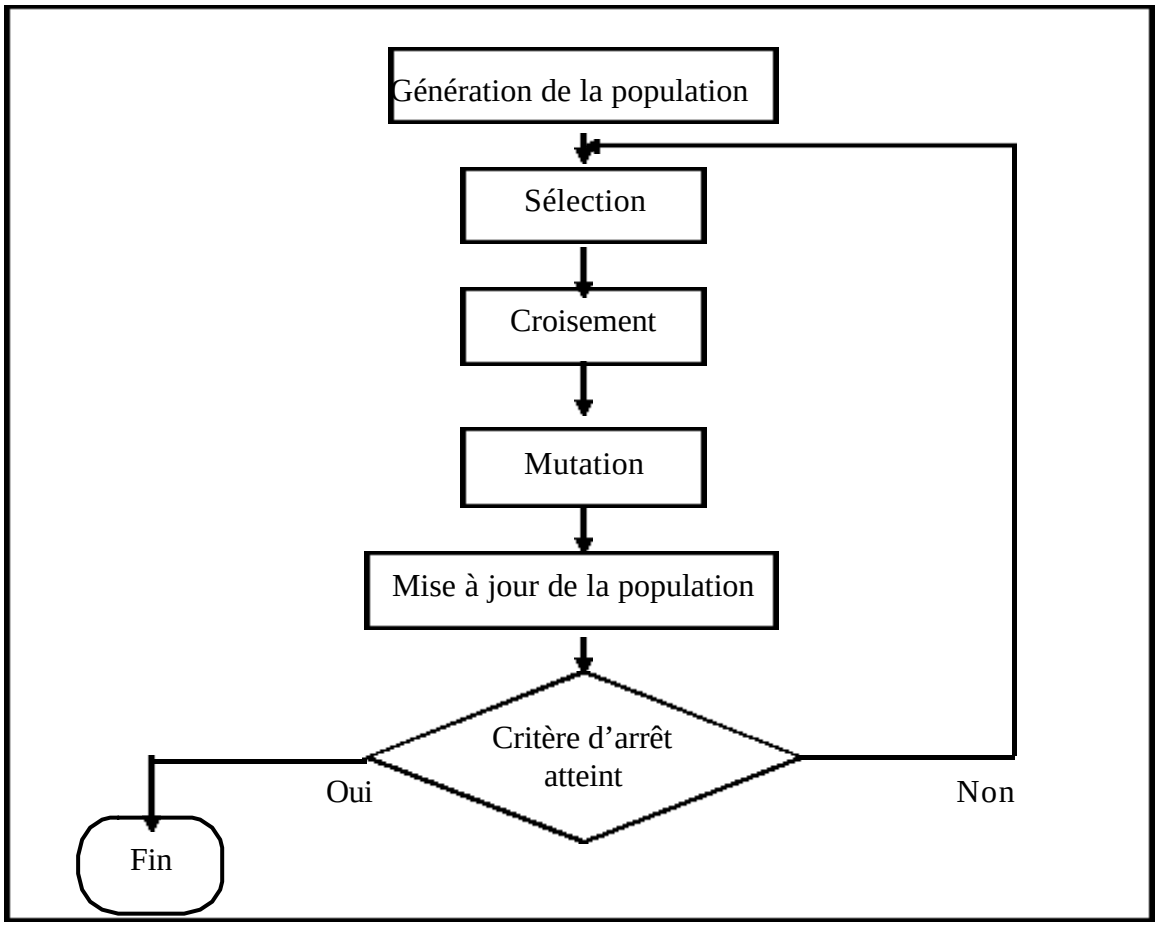


Figure9 : Organigramme d'un Algorithme Génétique.

VI.3. 1. Exemple

Prenons pour exemple un problème de maximisation. On veut chercher le maximum de la fonction $x \mapsto x^2$ sur l'ensemble [|1,31|] (on prend des entiers pour plus de simplicité).

Codage du problème

Chaque individu représentera une valeur pour x. Son génotype sera la valeur de x exprimée en binaire (il suffit de 5 bits, car 31 se note 11111).

Population initiale

On choisit ici une population de quatre individus. On détermine la population initiale en tirant au hasard la valeur de chaque allèle.

Supposons que l'on obtienne les individus suivants :

N° Génotype Phénotype

1 0 1 1 0 1 **13**

2 1 1 0 0 0 **24**

3 0 1 0 0 0 **8**

4 1 0 0 1 1 **19**

Evaluation de l'adaptation

Dans ce cas, la fonction d'adaptation est simple : il s'agit de la fonction $x \rightarrow x^2$. Ainsi pour calculer l'adaptation du premier individu, il suffit de calculer le carré de 13. On obtient ainsi:

N° Génotype Phénotype Adaptation

1 0 1 1 0 1 **13** **169**

2 1 1 0 0 0 **24** **576**

3 0 1 0 0 0 **8** **64**

4 1 0 0 1 1 **19** **361**

Sélection des parents

Il suffit de tirer au hasard 4 individus parmi la population en tenant compte de leurs adaptations respectives. On obtient ainsi une nouvelle population comprenant par exemple:

- Une copie de l'individu 1.
- Deux copies de l'individu 2.
- Aucune copie de l'individu 3.
- Une copie de l'individu 4.

C'est-à-dire:

N° Génotype Phénotype Adaptation

1 0 1 1 0 1 **13** **169**

2 1 1 0 0 0 **24** **576**

3 1 1 0 0 0 **24** **576**

4 1 0 0 1 1 **19** **361**

Recombinaison

On choisi ici, pour le croisement de déterminer au hasard les 2 parents, et de "couper" les chromosomes au hasard : la première partie du chromosome ira au premier descendant, alors que la seconde ira à l'autre. Supposons que l'individu 1 s'accouple avec l'individu 2 et l'individu 3 avec l'individu 4. (On note | la coupure).

N° Avant Après

1 0 1 1 0|1 0 1 1 0 0

2 1 1 0 0|0 1 1 0 0 1

3 1 1|0 0 0 1 1 0 1 1

4 1 0|0 1 1 1 0 0 0 0

Seconde génération

Dans cet exemple, on choisit de limiter la durée de vie de chaque individu à une génération. La nouvelle génération sera donc composée exclusivement d'enfants :

N° Génotype Phénotype Adaptation

1 0 1 1 0 0 **12** **144**

2 1 1 0 0 1 **25** **625**

3 1 1 0 1 1 **27** **729**

4 1 0 0 0 0 **16** **256**

Si on calcule la moyenne des valeurs d'adaptation, on obtient 293 pour la génération initiale, contre 439 pour la seconde. On se rapproche bien de la solution. Si on réitère le processus, on obtiendra des valeurs de plus en plus grandes, jusqu'à l'obtention de la solution.

D’où le fonctionnement d’un algorithme génétique est simple **[20]** :

- Codage du problème sous forme d’une chaîne binaire ;
- Génération aléatoire d’un population. Celle-ci contient un pool génétique qui représente un ensemble de solutions possibles ;
- Calcul d’une valeur d’adaptation pour chaque individu. Elle sera fonction directe de la proximité des différents individus avec l’objectif ;
- Sélection des individus devant se reproduire en fonction de leurs parts respectives dans l’adaptation globale ;
- Croissement des génomes des parents ;
- Sur la base de ce nouveau pool génétique, on repart à partir du point 3.

On peut également exprimer le fonctionnement d’un algorithme génétique en se référant aux notions de génotypes (GTYPE) et phénotypes (PTYPE) **[21]** :

- On sélectionne des paires de GTYPE en fonction de l’adaptation de leurs PTYPE respectifs ;
- On applique les opérateurs génétiques (reproduction, croisement et mutation) pour créer de nouveaux GTYPE ;
- On développe les GTYPE pour obtenir les PTYPE de la nouvelle génération et on repart du point 1.

Pour utiliser un algorithme génétique sur un problème particulier on doit disposer des cinq éléments suivants **[22]** :

- Un principe de codage des éléments de l'espace admissible du problème, en éléments sur lesquels peuvent s'appliquer les trois opérateurs présentés ci-dessus. Ce codage intervient après une phase indispensable de modélisation mathématique du problème.

Le choix du codage des données dépend du problème traité et conditionne l'efficacité (vitesse de convergence, précision,..) de l'algorithme génétique.

- Un mécanisme de génération de la population initiale. Cette population initiale, qui sert de base aux générations futures, doit être *la plus hétérogène possible*.
- Une fonction d'utilité *f*, permettant de calculer l'adaptation de chaque élément au problème. Ce critère retourne une valeur de R^+ appelée *fitness*.

- Des opérateurs permettant de diversifier et d'améliorer la population d'une génération sur l'autre ainsi que d'explorer le plus largement possible l'espace admissible.
- Des paramètres dimensionnels : taille de la population, critère d'arrêt, probabilités de croisement (P_c) et de mutation (P_m).

VI.3. 2. Résultats théoriques

Les algorithmes génétiques ont montré leur efficacité pratique bien avant que les résultats de convergence théorique ne soient établis. Nous disposons aujourd'hui de trois approches théoriques différentes permettant de mieux comprendre le fonctionnement des algorithmes génétiques, ces trois approches donnant des résultats asymptotiques.

- La théorie des Schémas développée par Holland **[15]**, constitue une première approche du problème. S'appliquant sur des chaînes de bits, elle étudie le comportement asymptotique de l'algorithme et l'effet des différents opérateurs sur la structure des schémas.
- La deuxième approche découle des résultats de convergence stochastique sur des méthodes de recuit simulé développées par Laarhoven et Aarts **[23]**. Sous certaines hypothèses, on montre la convergence asymptotique grâce à l'opérateur de mutation. Ce résultat est d'ailleurs conforme à l'intuition, puisque seul cet opérateur permet réellement l'exploration aléatoire de l'espace.
- L'approche théorique la plus récente est proposée par Raphaël Cerf **[24]** et utilise une modélisation par chaîne de Markov de l'algorithme génétique. Les résultats asymptotiques sont obtenus grâce à la théorie de Freidlin et Wentzell **[25]**.

VI.4. Conclusion

Les algorithmes génétiques sont des systèmes originaux, s'inspirant du fonctionnement présumé du vivant. On retient quatre points principaux :

1. Utilisation d'un codage des paramètres, et non des paramètres eux-mêmes ;
2. Travail sur une population de points, au lieu d'un point unique,
3. Utilisation des seules valeurs de la fonction à optimiser, et non de leur dérivée ou d'une autre connaissance auxiliaire ;
4. Utilisation de fonctions de transition probabilistes, non déterministes.

Une des particularités séduisantes des algorithmes génétiques, réside dans l'absence d'hypothèses particulières sur la régularité de la fonction objective. Aucune

hypothèse sur la continuité de cette fonction n'est requise, ses dérivées successives ne sont pas nécessaires, ce qui rend très vaste le domaine d'application de ces algorithmes.

Le peu d'hypothèses requises sur la fonction objective permet de traiter des problèmes très complexes. La fonction objective peut ainsi être le résultat d'une simulation. On peut même imaginer, pour régler certains paramètres de l'algorithme génétique lui-même tels que la taille de la population, les différents pourcentages de croisement et de mutation, d'utiliser un algorithme génétique. La rapidité de convergence du premier devenant ainsi fonction d'évaluation du second.

Le grand avantage des algorithmes génétiques est le fait que pour parvenir au résultat, on n'a pas besoin de connaître les caractéristiques de la solution du problème, mais seulement de déterminer parmi deux solutions quelle est la meilleure. Par contre ce genre d'algorithme est très coûteux en temps de calcul, difficile à programmer (les paramètres comme la taille de la population et la fonction d'évaluation sont difficiles à établir), et il n'a qu'une très faible chance, voire aucune, de trouver la solution idéale, il ne fait que s’en approcher.

Chapitre V
La cryptanalyse basée sur les algorithmes génétiques pour le crypto-système considéré

V.1. Introduction

Les crypto-systèmes basés sur le problème de satisfiabilité constituent une nouvelle tendance dans le domaine de la cryptographie. Cet intérêt vient d'une partie de l'étroite relation entre la cryptologie et la complexité, et d'une autre part pouvoir bénéficier de tous les travaux déjà effectués autour du problème SAT et des problèmes NP-complets en général.

L'utilisation du problème SAT dans un crypto-système implique que la principale propriété de ce système est que sa cryptanalyse nécessite la résolution d'une instance de SAT, pour pouvoir tirer profit de la complexité de SAT pour assurer la sécurité du système. Le problème de retrouver le texte clair à partir du texte chiffré sans la connaissance de la clef de déchiffrement est équivalant à la résolution de SAT.

L'étude des heuristiques d'optimisation numérique et de leurs applications à la cryptanalyse a une importante considération dans le monde où la puissance de traitement par ordinateur, et la distribution de système de calcul deviennent de plus en plus répandues.

Dans ce chapitre on va opter pour la cryptanalyse d'un crypto-système basé sur 3-SAT par l'heuristique des algorithmes génétiques.

V.2. Les principales propriétés d'un crypto-système basé sur SAT

Un tel crypto-système doit tirer sa robustesse de la complexité de SAT. Les principaux traits d'équivalence entre l'algorithme cryptographique et SAT peuvent être vus comme suit :

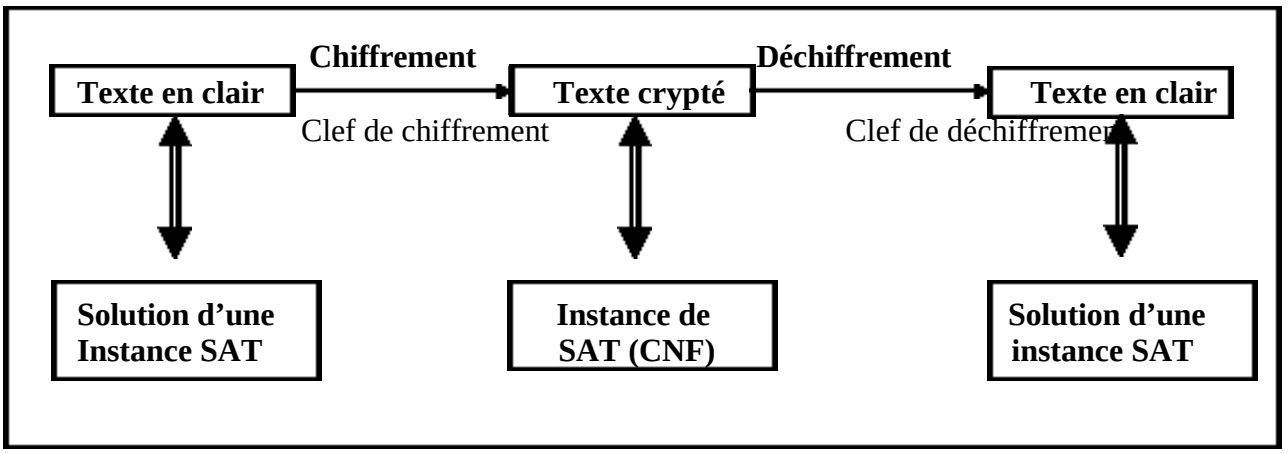


Figure10 : les principales propriétés d'un crypto-système basé sur SAT

- Le texte en clair doit être équivalent à une solution d'une instance de SAT (une assignation aux variables booléennes), ceci ne pose pas problème puisque les deux peuvent être considérés comme une suite de bits.

- La fonction de chiffrement a pour but d'obtenir une instance de SAT ayant pour solution le texte clair auquel elle a été appliquée, l'instance obtenue doit être d'une difficulté acceptable. Cette fonction de chiffrement doit être telle qu'on ne peut obtenir la même instance à partir de deux textes en clair différents avec la même clef de chiffrement.

- Le texte chiffré doit correspondre à l'instance de SAT obtenue précédemment. Elle peut contenir une description totale de l'instance, ou bien comporter juste une description partielle si un format bien défini est appliqué aux instances.

- La fonction de déchiffrement a pour première tâche de déterminer l'instance à partir du texte chiffré, ensuite elle doit retrouver le texte en clair correspondant à la solution de cette instance. Sans la connaissance de la clef de déchiffrement, ceci revient à résoudre le problème de satisfiabilité de l'instance donnée.

- La clef peut consister en une transformation d'une instance donnée vers une autre instance plus facile à résoudre, ou bien les paramètres d'une fonction heuristique qui permettra de résoudre le problème de satisfiabilité en un temps polynomial ; elle peut aussi contenir en plus une indication sur la solution dans le cas où plusieurs solutions existent.

V.3. Le crypto-système considéré

Le crypto-système considéré est basé sur une fonction à sens unique, qui transforme une donnée 2-SAT en une donnée 3-SAT

V.3.1. La fonction à sens unique qui transforme 2-SAT en 3-SAT

Le problème 2-SAT peut être défini de la façon suivante :

Soit une proposition logique E sous forme normale conjonctive (i.e. une conjonction de disjonctions) de degré 2 (i.e. chaque disjonction contient exactement 2 littéraux).

Question : E est-elle satisfiable, i.e. existe-t-il une affectation de valeurs de vérité aux littéraux qui rend E vraie ?.

Exemple : E = (A ou B) et (A ou B) et (B ou C).

Et de même, le problème 3-SAT et une proposition logique E sous forme normale conjonctive de degré 3 où chaque disjonction contient trois littéraux.

Exemple : E = (A ou B ou C) et (A ou B ou C).

Sachons que le problème 2-SAT est polynomial, et le problème 3-SAT est NP-Complet, la fonction à sens unique consiste à trouver une transformation de 2-SAT à 3-SAT.

La méthode de cette transformation est la suivante :

- Soit E une proposition logique 2-SAT sous forme normale conjonctive, par exemple :

$C_1 = x_1 \vee x_2$

$C_2 = x_1 \vee x_4$

$C_3 = x_1 \vee x_5$

$C_4 = x_2 \vee x_3$

$C_5 = x_4 \vee x_6$

$C_6 = x_2 \vee x_5$

$C_7 = x_2 \vee x_4$

Les variables de ce problème sont : $X = (x_1, x_2, \ldots, x_6)$

- Soit une affectation X = (0, 1, 1, 1, 0, 1) qui rend ce système satisfait, c-à-d : pour $x_1 = 0$, $x_2 = 1$, $x_3 = 1$, $x_4 = 1$, $x_5 = 0$ et $x_6 = 1$, la proposition logique E = 1.

- Dans cette étape, on va ajouter à chaque clause une variable qui doit avoir une valeur fausse, pour ne pas changer la nature de chaque clause et donc du système 2-SAT.

- Par la suite, on obtient un système 3-SAT avec un vecteur V, où chaque élément de ce vecteur représente le rang de la variable ajoutée dans chaque clause du nouveau système 3-SAT.

- Chaque élément du vecteur V doit avoir une valeur égale à 1, 2, ou 3.

- Donc, en conséquence, à partir d'un problème 2-SAT, on a obtenu un problème 3-SAT équivalent (c-à-d : que les deux systèmes ont les mêmes solutions), et un vecteur V qui désigne les rangs des variables ajoutées au système 2-SAT.

- A la fin, on extrait deux sortes de clés :

  - Le système 3-SAT, comme clé publique.
  - Le vecteur V, comme clé privée.

V.3.2. L'algorithme de la fonction à sens unique 2-SAT/3-SAT

Entrées :

- Soit un problème 2-SAT de n variables et m clauses :

$X = \{ x_1, x_2, \ldots, x_n \}$ et $C = \{ C_1, C_2, \ldots, C_m \}$.

- Chaque clause est de la forme : $C = x_i + x_j$.

- Soit S le vecteur qui représente la solution de notre système 2-SAT, donc S est de la forme : $S = (s_1, s_2, \ldots, s_n)$.

Sorties :

- Le vecteur V de dimension m qui désigne les rangs des variables ajoutées au système 2-SAT, $V = (v_1, v_2, \ldots, v_m)$.

- Un problème 3-SAT de n variables et m clauses issu de problème 2-SAT.

ALGORITHME

P ——1,

Pour k allant de 1 à m **faire**

Trouve —— faux,

Tant que Trouve **faire**

Si P < i **alors**

- On ajoute à la clause C_k la variable ayant l'indice P à la première position,
- Si $s_p = 0$, on va ajouter (x_p) à la clause C_k,
- Si $s_p = 1$, on va ajouter (x_p) à la clause C_k,
- V[k] = 1 (c-à-d, (le rang de x_p) = 1),
- Trouve —— vrai,

Sinon

Si (p < j) et (p > i) alors

- On ajoute à la clause C_k la variable ayant l'indice P à la deuxième position,
- Si $s_p = 0$, on va ajouter (x_p) à la clause C_k,
- Si $s_p = 1$, on va ajouter (x_p) à la clause C_k,
- V[k] = 2 (c-à-d, (le rang de x_p) = 2),
- Trouve —— vrai,

Sinon

Si P > j alors

- On ajoute à la clause C_k la variable ayant l'indice P à la troisième position,
- Si $s_p = 0$, on va ajouter (x_p) à la clause C_k,
- Si $s_p = 1$, on va ajouter (x_p) à la clause C_k,
- V[k] = 3 (c-à-d, (le rang de x_p) = 3),
- Trouve —— vrai,

FSi

FSi

FSi

FTQ

Si P < n alors

P —— P+1,

Sinon

P —— 1,

FSi

FPour

En fin, on a génère un problème 2-SAT camouflé sous un problème 3-SAT qui est la clé publique, et la clé privée qui est le vecteur V.

V.3.3. La fonction inverse de la fonction à sens unique 2-SAT/3-SAT

Entrées :

- Soit le vecteur V de dimension m qui désigne les rangs des variables supplémentaires au système 2-SAT : $V = (v_1, v_2, \ldots, v_m)$.

- Soit un problème **3-SAT** de n variables et m clauses

Sorties :

- Un problème 2-SAT de n variables et m clauses.

ALGORITHME

Pour chaque clause C_k **faire**

- Enlever la variable x_i qui a le rang v[k] dans la clause C_k,

FPour

V.3.4. L'algorithme du crypto-système considéré

On se base sur un algorithme du problème 2-SAT qui est un problème polynomial facile à résoudre, camouflé sous un problème 3-SAT pour en faire la clé publique, et un vecteur V comme clé privée. Plus en détail, voici ce que cela donne :

- Soit un système 2-SAT de n variables et m clauses, et soit la solution $S = (s_1, s_2, \ldots, s_n)$ qui rend valide le système 2-SAT.

En appliquant l'algorithme de la fonction à sens unique 2-SAT/3-SAT, il en résulte :

- Un système **3-SAT** de n variables et m clauses comme clé publique.
- Un vecteur V de dimension m, où chaque élément doit être compris entre 1 et 3.

Envoi du message :

On veut envoyer le message $M = (m_1, m_2, \ldots, m_n)$(en binaire).

On calcule $MC = (c_1, c_2, \ldots, c_n)$(message chiffré) de la façon suivante:

- Si $m_i = s_i$ alors $c_i = 1$, sinon $c_i = 0$.

Puis on envoie le message chiffré MC.

Réception du message :

Quelqu'un interceptant le message chiffré MC et connaissant le système 3-SAT utilisé ne peut pas remonter au message initial M, car il ne peut pas connaître la solution S qui est la solution de 3-SAT.

Pour ce faire :

- On applique l'algorithme inverse de la fonction à sens unique sur le système 3-SAT et le vecteur V, on obtiendra par la suite le système 2-SAT, et par la suite la solution $S = (s_1, s_2, \ldots, s_n)$.

- On calcule le message initial M de la façon suivante :

  - Si $c_i = s_i$ alors $m_i = 1$, sinon $m_i = 0$.

Et enfin, on obtiendra le message initial $M = (m_1, m_2, \ldots, m_n)$.

Application numérique :

- **Fabrication des clés publiques et privées :**

- On choisit un système **2-SAT** de 6 variables et 7 clauses.

$$\begin{cases} C_1 = x_1 \vee x_2 \\ C_2 = x_1 \vee x_4 \\ C_3 = x_1 \vee x_5 \\ C_4 = x_2 \vee x_3 \\ C_5 = x_4 \vee x_6 \\ C_6 = x_2 \vee x_5 \\ C_7 = x_2 \vee x_4 \end{cases}$$

- On choisit comme solution de ce problème la solution : S = (0, 1, 1, 1, 0, 1).

- après application de l'algorithme de la fonction unique 2-SAT/3-SAT, on obtiendra :

  - Le système **3-SAT** qui va être par la suite clé publique.

$$
\begin{cases}
C_1 = x_1 \vee x_2 \vee x_3 \\
C_2 = x_1 \vee x_4 \vee x_5 \\
C_3 = x_1 \vee x_5 \vee x_6 \\
C_4 = x_1 \vee x_2 \vee x_3 \\
C_5 = x_2 \vee x_4 \vee x_6 \\
C_6 = x_2 \vee x_3 \vee x_5 \\
C_7 = x_2 \vee x_4 \vee x_5
\end{cases}
$$

  - Le vecteur V de dimension 7 comme clé privée.

  V = (3, 3, 3, 1, 1, 2, 3).

• **Envoi du message :**

On veut envoyer le message M en binaire :

M = 1 0 0 1 0 1 0 1 0 1 0 0

$$
\begin{cases}
C_1 = x_1 \vee x_2 \\
C_2 = x_1 \vee x_4 \\
C_3 = x_1 \vee x_5 \\
C_4 = x_2 \vee x_3 \\
C_5 = x_4 \vee x_6 \\
C_6 = x_2 \vee x_5 \\
C_7 = x_2 \vee x_4
\end{cases}
$$

- On calcule le message chiffré MC en appliquant la règle d'envoi, on obtiendra :

MC = (0 0 0 1 1 1 1 1 0 1 1 0).

Et on envoi le message chiffré MC.

- **Réception du message :**

On calcule 2-SAT par l'algorithme inverse de la fonction à sens unique 2-SAT/3-SAT, et on obtiendra le système suivant :

$$\begin{cases} C_1 = x_1 \vee x_2 \\ C_2 = x_1 \vee x_4 \\ C_3 = x_1 \vee x_5 \\ C_4 = x_2 \vee x_3 \\ C_5 = x_4 \vee x_6 \\ C_6 = x_2 \vee x_5 \\ C_7 = x_2 \vee x_4 \end{cases}$$

- On calcule la solution S du système 2-SAT :

S = (0, 1, 1, 1, 0, 1).

- On applique la règle de réception sur le message MC pour obtenir le message déchiffré MD = (1 0 0 1 0 1 0 1 0 1 0 0).

Donc, on obtiendra le message initial M :

M = (1 0 0 1 0 1 0 1 0 1 0 0).

V.4. La cryptanalyse du crypto-système considéré

Comme on a cité auparavant, on va attaquer ce crypto-système par un algorithme génétique.

Le fonctionnement d'un algorithme génétique est le suivant d'après le chapitre précédent :

- Codage du problème.
- Génération aléatoire d'une population.
- Fonction d'adaptation.
- Sélection des individus devant se reproduire.
- Croisement et mutation.
- En se basant sur le nouveau pool génétique, on repart à partir du point 3.

V.4.1. Codage de la solution

Dans notre cas l'espace de recherche Ω est un ensemble de 2^n instanciations ($x_1, x_2, \ldots x_n$), n étant le nombre de variables booléennes utilisées dans la donnée 3-SAT.

L'individu qui représente une solution donnée (qui peut être bonne ou mauvaise) est une instanciation de variables, donc un individu représenté sous forme d'une chaîne binaire.

V.4.2. Génération de la population initiale

La population initiale représente un ensemble d'individus de taille N,chaque individu (chromosome) est une solution potentielle du problème. La population initiale est générée aléatoirement. Cependant certaines méthodes garantissent le démarrage avec des solutions assez bonnes, obtenues à l'aide d'heuristiques.

On utilise la procédure heuristique G-SAT **[26]** qui permet d'obtenir une solution de bonne qualité.

Algorithme de génération d'une population initiale

Début

Pour i :=1 à Taille_pop

Faire

Pour j := 1 à Taille_indiv

Faire

Générer un nombre aléatoire entier dans l'intervalle [0.1]

Fait

Fait

FIN.

V.4.3. La fonction d'adaptation (Fitness)

Soit une donnée 3-SAT à k clauses et n variables booléennes :

$C_1 = I_{11} + I_{12} + I_{13}$
$C_2 = I_{21} + I_{22} + I_{23}$
.
.
.
$C_k = I_{k1} + I_{k2} + I_{k3}$

La longueur des clauses est 3.

On définit, pour chaque clause C_i $(1 \leq i \geq k)$ une variable entière y_i pouvant prendre les deux valeurs entières 0 ou 1.

Soit x une instanciation (individu), $x = (x_1, x_2, \ldots, x_n)$.

$$y_i(x) = \begin{cases} \mathbf{1} & \text{Si x satisfait la clause } C_i \\ \mathbf{0} & \text{Sinon.} \end{cases}$$

La valeur de la fonction d'adaptation est donnée par :

$$\mathbf{F(x) = \sum_{i=1}^{k} y_i(x).}$$

k : Le nombre de clauses dans la donnée.

Notre but est de maximiser cette fonction dans Ω ($\Omega = \{0,1\}^n$), n étant le nombre de variables de la donnée.

Algorithme d'adaptation

Début

Pour i :=1 à Taille_pop

Faire

Compter le nombre de clauses que l'individu i satisfait ;

/* Ce nombre représente le coût de l'individu i*/

Fait

FIN.

V.4.4. Les opérateurs génétiques

V.4.4. 1. La sélection

La phase de la sélection favorise les individus les mieux adaptés pour participer à la phase de reproduction.

Soit f (ind i) la valeur de la fonction d'adaptation de l'individu i, la probabilité de sélection d'un individu i est :

$$\mathbf{P_i = f(ind\ i) / \sum_{I=1}^{Taille\_pop} f(ind\ i)}$$

Plus la valeur de la fonction d'adaptation d'un individu est grande, plus sa probabilité de sélection est grande.

Soit la probabilité de sélection cumulée Q_i :

$$Q_i = \sum_{j=1}^{Taille\_pop} P_j .$$

Q_1, $Q_{2,\ldots\ldots,}$ Q_n correspondent aux individus classés par ordre décroissant de leurs adaptations, ainsi que Q_1 est associe à l'individu ayant le meilleur coût.

Le calcul des Q_i nous permet d'établir un tableau de fréquences de la population triée par ordre croissant et nous aurons Tab_freq [i] = $Q_i * 100$.

On définit la stratégie de sélection comme suit :

On génère aléatoirement sur l'intervalle [0,100] un nombre entier r. L'individu i est sélectionné lorsque r ≤ Tab_freq [i]. Un même individu peut être sélectionné plusieurs fois si sa valeur de la fonction d'adaptation est grande, un individu ayant une petite valeur de la fonction d'adaptation (individu mauvais) risque de ne pas être sélectionné.

Algorithme de sélection

Début

- Trier la population selon un ordre décroissant de leurs valeurs de la fonction d'adaptation ;
- Remplir le tableau Tab_freq des probabilités cumulées ;
- Générer un nombre aléatoire entier dans l'intervalle [0,100] ;
- Parcourir le tableau Tab_freq tant que le nombre généré aléatoirement est supérieur à la valeur de Tab_freq ;
- Retenir l'indice du tableau ; celui-ci représente l'indice de l'individu sélectionné.

FIN.

V.4.4.2. Le croisement

Cette phase s'applique sur deux individus parents choisis au hasard dans la population d'individus déjà sélectionnés, pour produire de nouveaux individus. Ces deux descendants sont appelés des enfants.

En pratique, seulement une partie de la population participe à cette opération c'est-à-dire chaque individu se voit attribué une même probabilité P_c de participer à un croisement et c'est un tirage aléatoire qui détermine sa participation à cette opération.

Le nombre attendu d'individus qui subissent le croisement est : $P_c * Taille\_pop$.

Pour notre problème 3-SAT, l'espace de solutions possibles est $\Omega=\{0,1\}^n$, on voit clairement qu'il n'y a aucune contrainte sur l'ensemble des instanciations Ω, ainsi chaque point de Ω est une solution possible. Cette caractéristique spécifique à notre problème nous laisse le choix du croisement à utiliser (unipoint , bipoint, N-points, uniforme).

Algorithme de croisement

Début

- Sélection de deux individus pour le croisement ;
- Choisir le type de croisement ;
- Selon le type choisi, appliquer le croisement avec une probabilité P_c.

FIN.

V.4.4.3. La mutation

Cette phase est considérée comme un opérateur secondaire par rapport au croisement.

Dans notre problème cette phase permet d'introduire de nouvelles caractéristiques dans l'individu qui n'apparaît pas forcément chez les parents, c'est-à-dire à une tendance d'améliorer la solution du problème ; comme dans notre problème on a un codage binaire, cette phase consiste à changer la valeur d'un bit choisi aléatoirement avec une probabilité de mutation P_m , cela revient tout simplement à changer un 1 en 0 et vise-vers ça.

Pour pallier au problème de la convergence prématurée dû à la similarité des individus dans une population, nous avons proposé une variante de l'algorithme génétique où on applique la mutation adaptative. Cette dernière consiste d'une part à diminuer le taux de mutation durant les premières générations de l'algorithme, car généralement celles ci sont constituées d'individus diversifiés ; d'autre part élever le taux de mutation aux dernières générations.

Algorithme de mutation

Début

- Choisir un nombre aléatoire entier r dans l'intervalle [0,100];

Tant que r < taux de mutation

Faire

- Choisir un gène aléatoire sur l'individu et l'inverser ;
- Choisir un nombre aléatoire entier r dans l'intervalle [0,100] ;

Fait

FIN.

Le remplacement

On a vu que l'application des opérateurs génétiques (croisement et mutation) permettent d'engendrer une nouvelle population appelée population d'enfants, les meilleurs individus de cette dernière vont remplacer les plus mauvais de la population des parents.

V.5. Algorithme génétique proposé

Début

- Générer la population initiale aléatoirement;

Tant que (le nombre de génération total n'est pas atteint) & (le nombre de générations stables n'est pas atteint) & (Solution optimale non trouvée)

Faire

Pour i :=1 à Taille_pop

Faire

- Sélectionner deux individus ;
- Choisir un nombre aléatoire entier R_c dans l'intervalle [0,100];
- Si R_c < Taux de croissement alors appliquer le croisement ;
- Choisir un nombre aléatoire entier r dans l'intervalle [0,100];

Tant que r < Taux de mutation

Faire

- Choisir un gène aléatoire sur l'individu obtenu après croisement et l'inverser ;
- Choisir un nombre aléatoire entier r dans l'intervalle [0,100];

Fait

- Evaluer l'individu produit ;

Fait

- Remplacer les individus produits en favorisant les meilleurs ;

Fait

FIN.

Pour la cryptanalyse du crypto-système considéré, on a deux possibilités :

- On dispose que de la clé publique qu'est le système 3-SAT et le texte chiffré, la cryptanalyse revient à résoudre le système 3-SAT par l'algorithme proposé, puis l'utilisation des solutions permettent le passage du texte chiffré au texte clair.

- On connaît la méthode, c'est-à-dire trouver à partir du système 3-SAT la clé privée qu'est le vecteur V et le texte chiffré, la cryptanalyse revient à trouver le vecteur V par l'algorithme proposé, qui permet le passage du système 3-SAT au système 2-SAT qu'est polynomial, facile à résoudre puis l'utilisation des solutions du système 2-SAT permettent le passage du texte chiffré au texte clair.

V.6. Conclusion

La solution d'un problème donné à l'aide des algorithmes génétiques, nécessite d'une part la codification des paramètres du problème et la détermination de la fonction objective à optimiser.

C'est grâce aux expérimentations qu'on peut définir l'efficacité de l'algorithme de résolution ainsi que les paramètres principaux tels que la taille de la population, le taux de sélection, le taux de croisement et le taux de mutation.

Parmi les principaux problèmes des algorithmes génétiques est la convergence très rapide de la population vers un individu particulier en raison de sa domination grâce à son adaptation (fitness) ; ce qui peut engendrer le blocage de l'algorithme dans un optimum local. Afin de remédier à ce problème, plusieurs techniques ont été développées ; citons le Sharing qui fut introduit par Holland **[15]**, et développé par Goldberg et Richardson **[27]**.

CONCLUSION GENERALE ET PERSPECTIVES

Jusqu'à l'heure actuelle, il n'existe pas d'algorithmes polynomiaux qui permettent d'obtenir des résultats exacts à un problème NP-complet.

Les chercheurs se sont intéressés à une classe d'algorithmes très intéressante permettant d'obtenir des solutions acceptables aux problèmes NP-complets en un temps polynomial, les algorithmes constituant cette classe sont appelés heuristiques.

Parmi eux, on trouve les algorithmes génétiques. Ces derniers constituent une alternative intéressante aux techniques classiques pour la résolution des problèmes d'optimisation. Ceci est dû à quatre caractéristiques essentielles des algorithmes génétiques qui se résument en :

- La manipulation d'un codage des paramètres et non pas par les paramètres eux-mêmes.
- L'exploration au moyen d'une population, au lieu d'un point unique.
- L'exploration aveugle à partir uniquement des valeurs de la fonction d'adaptation.
- L'utilisation des règles de transition probabilistes et non déterministes.

Dans le cadre de ce travail, la cryptanalyse proposée repose sur la résolution du problème 3-SAT par l'approche des algorithmes génétiques.

Nous avons commencé, par effectuer un survol sur la cryptologie, ensuite des principales notions de complexité et ses classes, présenter une définition formelle du problème de satisfiabilité, et résumer des algorithmes de résolution les plus importants. Comme nous avons aussi présenté une étude générale sur les crypto-systèmes ainsi que leurs attaques.

La partie la plus importante est celle qui est présentée au niveau du quatrième et cinquième chapitre. Elle est considérée comme l'objectif de ce travail ; on a choisi l'approche des algorithmes génétiques pour la résolution du problème 3-SAT, c'est-à-dire la cryptanalyse du crypto-système existant.

Le codage de la transformation associée à un crypto-système en formule booléenne à des fins de cryptanalyse.

Nous avons opté pour l'approche des algorithmes génétiques, pour la résolution de problème 3-SAT ; vu les résultats satisfaisants obtenus par cette approche pour la cryptanalyse, citons par exemple résoudre des systèmes simples de substitution **[28]**, des systèmes de transposition **[28]**, des systèmes basés sur le sac à dos(problème NP-Complet) **[29]**,et les systèmes à machines à rotors **[30,31,32]**.

Une des premières perspectives de ce travail est l'implémentation du système de cryptanalyse constitué de :

- Résolution du problème 3-SAT,
- Passage du texte chiffré au texte clair.

D'autres objectifs s'inscrivent dans un proche avenir, concernant la résolution du problème SAT par d'autres méthodes telles que, la recherche dispersée (Scatter Search), la colonie de fourmis (ACO " Ant Colony Optimization "), la recherche tabou et le recuit simulé.

BIBLIOGRAPHIES

[1] B.C. Dupuis : “Short History of Crypto”.

[2] F. Marie:“Histoire de la Cryptologie”.

[3] D. Blan: “ La cryptographie”.

[4] B.F. Cohen:“A Short History of Cryptographiy”.

[5] D.K.Branstad, J.Gait et S.Ktzke (1976) : “Report on the Workshop on Cryptography in support of Computer Security”.

[6] B. Schneier (1994) : “Cryptographie Appliquée”.

[7] V. Rijmen et J. Daemen (2000) : “ Advanced Encryption Standard”.

[8] S. Grafinkel (1995) : “ PGP : Pretty Good Privacy”.

[9] A. Stephen Cook (1971) : “ The Complexity of Theorem Proving Procedures”. Third Annual ACM Symposium on Theory of Computing.

[10] M. Davis et H. Putnam (1960) : “ A Computing Procedure for Quantification Theory”.

[11] M. Davis, G. Logemann et Donald Loveland (1962) : “ A Machine Program for Theorem Proving”.

[12] H. Drias (1993) : “ Approche Probabiliste du Dénombrement des Solutions du Problème de Satisfiabilité”. Thèse de Doctorat USTHB.

[13] A. Vaguer (2003) : “ Résolution Paralléle du Problème SAT Application à la Cryptographie”. Rapport de Stage DEA.

[14] B. Becket (1990) : “ Introduction aux Méthodes de la cryptologie”. Logique Mathématique Informatique.

[15] J. Holland (1975) : “ Adaptation in Natural and Artificial Systems”. University of Michigan.

[16] D. Goldberg (1989) : “ Genetic Algorithms”. Addison Wesly Editor.

[17] Y. Bekhtaoui et S. Kacha (1997) : “ Méthode Génétique et Recuit Simulé au PVC ”. Mémoire de PFE USTHB.

[18] G. Syswerda (1989) : " Uniform Crossover in Genetic Algorithms". Proc. of the Third International Conference on Genetic Algorithms Bloomington USA.

[19] A. Abderrazak, A. Djellouli Hadj (1998) : " Un algorithme Génétique Parallèle et Hybride pour le Problème MAX-SAT". PFE USTHB.

[20] J.P. Rennard (2000) : " Résolution Genetic Algorithm Viewer". Démonstration d'un Algorithme Génétique.

[21] J.C. Hendin (1994) : " La Vie Artificielle, Hernés".

[22] C. Bon Temps (1995) : " Principes Mathématiques et Utilisations des Algorithmes Génétiques".

[23] Laarhoven et Aarts: " Méthodes de Recuit Simulé". Article.

[24] R. Cerf (1994) : " Une Théorie Asymptotique des Algorithmes Génétiques". Thèse de Doctorat Université de Montpellier II.

[25] M.I. Freidlin et Wentzell (1983) : "Random Perturbation of Dynamical System ". Springer Verlag New York.

[26] B. Selman, H. Levesque et D. Mitchell (1992) : "A New Method for Solving Hard Satisfiability Problem ".

[27] Goldberg et Richardson (1987) : "Genetic Algorithms with sharing for multimodal Function Optimization".

[28] R. Spillman et autre (1993) : " Use of Genetic Algorithm in the Cryptanalysis of Simple Substitution Cipher".

[29] R. Spillman (1993) : " Cryptanalisis of Knapsack Cipher Using Genetic Algorithm".

[30] J.W. Mann, A. Kapsalis et G.D. Smith (1995) : " The GAmeter Toolkit". In V.J. Rayward- Smith, Editor Applications of Modern Heuristic Methods Alfred Waller.

[31] A. Kapsalis et G.D. Smith (1992) : " The GAmeter Toolkit Manual".

[32] A. Kapsalis, V.J. Rayward- Smith et GD Smith (1993) : " Fast Sequential and Parallel Implementation of Genetic Algorithm of GAmeter Toolkit". Editors Procedings of the int. Conf on Artificial Neural Nets and Genetic Algorithm Springer Verlang.

[33] J.P. Barthélemy, G. Cohen et A. Lobstein (1992) : " Complexité Algorithmique et Problèmes de Communications".

[34] M. Khabzaoui, S. Sebti (2000) : “ Contribution à la resolution du problème MAX-SAT par la Méta-Heuristique Scatter Search”. PFE USTHB.

[35] R. Belkacem (2002) : “ Résolution des problèmes MAX-SAT et Partitionnement avec les Algorithmes Génétiques et la Recherche Dispersée dans un environnement Parallele Simulé sur Réseau”. PFE INI.

[36] M.A. Garici (2003) : “ Cryptologie et Problème de Satisfiabilité”. Mini-Projet USTHB.

[37] S. Ibri (2001) : “ Parallélisation du Système de Colonie de Fourmis et Application au Problème MAX-SAT ”. Magister USTHB.

[38] G. Audemard (2001) : “ Résolution du Problème SAT et Génération de Modèles Finis en logique du Premier Ordre ”. Thèse de Doctorat en Informatique.

[39] S. Mihna (1996) : “ Tournées de Véhicules et Satisfiabilité logique”. Thèse de Doctorat en Philosophie.

[40] J.P. Dandrieux (2000) : “ Contribution à l’analyse de la Complexité de Problème de Résolution de Contraintes”. Thèse de Doctorat.