Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie Houari Boumedienne
Faculté des Mathématiques

Mémoire présenté pour l'obtention du diplôme de Magister en mathématiques

Spécialité : Recherche Opérationnelle : Génie Mathématique

Par

Louiza REZKALLAH

Thème

DE LA CRYPTOGRAPHIE CLASSIQUE A LA CRYPTOGRAPHIE MODERNE THEORIE ET APPLICATION

Soutenu publiquement, le 03/10/2007 devant le jury composé de :

H. AIT HADDADENE	Professeur	USTHB	Président
S. BOUROUBI	Maître de Conférences	USTHB	Directeur de mémoire
A. BERRACHEDI	Professeur	USTHB	Examinateur
I. BOUCHEMAKH	Maître de Conférences	USTHB	Examinatrice
M. BOUDHAR	Maître de Conférences	USTHB	Examinateur

Table des matières

Introduction générale

Pour de bonnes ou de mauvaises raisons, l'homme a toujours ressenti le besoin de dissimuler des informations, bien avant l'apparition des premiers ordinateurs et des machines à calculer. La cryptographie lui permet d'assurer ce besoin. La cryptographie appelée aussi science du secret, est l'ensemble des méthodes et techniques permettant de dissimuler une information. Ainsi on utilise la cryptographie lorsque l'on désir transmettre une information de telle sorte que toute personne non autorisée soit incapable de déchiffrer le message transmis, si ce dernier aurait été intercepté. Notre correspondant et lui seul doit être capable de reconstituer l'information à partir du message transmis. Cependant, ce dernier est souvent confronté à la cryptanalyse.

La cryptanalyse s'oppose, en quelque sorte à la cryptographie. En effet, si "déchiffrer" consiste à retrouver l'information au moyen d'un paramètre appelé clé, "cryptanalyser" c'est tenter de se passer de cette dernière. Il convient de remarquer qu'un algorithme est considéré comme cassé lorsqu'une attaque permet de retrouver la clé en effectuant moins d'opérations que via une attaque par force brute. Une attaque est souvent caractérisée par les données qu'elle nécessite, notons qu'il existe plusieurs familles d'attaques cryptanalytiques.

Longtemps, la cryptographie est restée un art réservé aux hommes d'état, aux ambassadeurs, aux militaires et aux espions. C'est au cours de ces dernières années qu'elle est devenue une science indispensable à notre société moderne d'information, anisi il y a eu une explosion de recherches académiques publiques en cryptographie. La cryptographie fait aujourd'hui partie de notre vie quotidienne : cartes à puces et monétique, Internet et courrier électronique, ne faisons-nous pas tous de la cryptographie sans le savoir ?

Ce mémoire a deux objectifs principaux, le premier est d'expliquer le chiffrement tout en mettant en valeur ce qu'il peut apporter dans notre vie de tous les jours. Le second est de permettre à toute personne non spécialiste en mathématique ou en informatique, donc à un large publique, de découvrir rapidement ses différents aspects essentiels, à savoir les différentes méthodes utilisées et vulnérabilité ou non de chacune. Il s'adresse aussi à toute personne qui désir approfondir des algorithmes de cryptographie particuliers. C'est pour ces raisons que nous avons réalisé le logiciel RLBS-CRYPTTOOL qui permet l'utilisation des méthodes et techniques de cryptographie les plus célèbres, et ce grâce à son interactivité et aux fenêtres d'aide qu'il contient.

Les applications principales de la cryptographie que nous exposerons par la suite sont bien sûr le chiffrement des messages pour en garantir la confidentialité, mais aussi l'authentification, la signature de documents numériques et le contrôle de l'intégrité des messages.

Ce mémoire est structuré comme suit :

Un premier chapitre, dans lequel nous abordons les définitions nécessaires à la compréhension des techniques que nous exposons par la suite.

Dans le second chapitre, on présente les chiffres cryptographiques utilisés avant l'ère informatique qui sont fondés sur deux grands procédés, la substitution et la transposition, en donnant à chaque fois des exemples qui facilitent leur compréhension, ainsi que la machine à chiffrer Enigma utilisée pendant la seconde guerre mondiale. Pour chaque algorithme de chiffrement nous avons donné une méthode permettant de le casser.

Le troisième chapitre rassemble les notions d'arithmétiques auxquelles on fait appel.

Le quatrième chapitre est consacré aux techniques de chiffrement à clé secrète, ou le chiffrement symétrique, dans lequel une même clé secrète est partagée entre les interlocuteurs. L'algorithme de chiffrement symétrique le plus connu est l'algorithme DES adopté en 1976. Cet algorithme s'est révélé résistant aux attaques par cryptanalyse différentielle et linéaire apparues par la suite. Cependant la taille de la clé secrète limitée à 56 bits le rendait vulnérable aux attaques par recherche exhaustive de la clé. La première solution été l'adoption du triple DES noté TDES qui permet d'augmenter significativement la

sécurité du DES, toutefois il a l'inconvénient majeur de demander plus de ressources pour le chiffrement et le déchiffrement. Dès 1997, le NIST (National Institute of Standards and Technology) lance un appel international pour trouver un standard de chiffrement successeur du DES. Le vainqueur était le AES.

A la fin de ce chapitre nous proposons un cryptosystème basé sur le problème de factorisation.

Dans le cinquième chapitre nous donnons quelques définitions et règles de calcul de la théorie de la complexité fréquemment utilisée en cryptographie, qui permettent de calculer la complexité d'un algorithme de chiffrement et/ou d'un algorithme de cryptanalyse, donc de juger la sécurité ou la vulnérabilité d'un algorithme ainsi que sa rapidité.

Le sixième chapitre, est consacré aux techniques de chiffrement à clé publique, ou le chiffrement asymétrique. Ce concept inventé en 1976 par Whitfield Diffie et Martin Hellman, marquant ainsi la naissance de la cryptologie moderne fondée sur la notion de fonction à sens unique. Chaque utilisateur possède deux clés: une clé publique et une clé privée. Pour chiffrer un message, on utilise la clé publique du destinataire, qui est seul à pouvoir le déchiffrer en utilisant sa clé privée correspondante, nous expliquons pour cette classe les différents algorithmes, Sac à dos, RSA et Rabin, ces deux derniers sont plus résistants car résolvent le problème de distribution des clés posé par la cryptographie à clé secrète. L'algorithme RSA permet aussi la signature de documents et est utilisé pour le chiffrement des clés secrètes dites aussi clés de session dans le chiffrement mixte qui est l'objet du septième chapitre. Dans ce dernier nous présentons une réalisation d'un cryptosystème mixte basé sur le problème de satisfaisabilité.

Dans le dernier chapitre, nous présentons le logiciel RLBS-CRYPTTOOL. Ce logiciel a regroupé les méthodes de chiffrement les plus célèbres comme il contient aussi un outil de calcul sur les grands nombres entiers, il est ainsi un service à l'enseignement et à la recherche scientifique.

Chapitre 1

Notions de base

Sommaire

1.1 Introduction

Supposons que quelqu'un, appelons le expéditeur, veut envoyer un message à quelqu'un d'autre, appelons le, destinataire. Cet expéditeur veut être sûr qu'aucun intermédiaire ne puisse agir sur le message de quelque façon que ce soit: spécifiquement, l'intermédiaire ne peut pas intercepter et lire le message, ni intercepter et modifier le message, ni fabriquer un message factice pouvant être considéré comme valide par le destinataire légitime. Ces problèmes ne sont pas nouveaux, et un moyen de les résoudre est d'utiliser la cryptographie.

1.2 Terminologie

Cryptage, crypter

Termes dérivés de l'anglais *to encrypt* et souvent employés à la place de chiffrement et chiffrer, c'est l'application d'un algorithme cryptographique à un ensemble de données appelées *texte en clair* afin d'obtenir un *texte chiffré*. Par la suite on emploie indifféremment les termes texte en clair, message et message clair.

Déchiffrement

Action inverse du chiffrement, lorsque celui-ci est réversible : à l'aide d'un algorithme cryptographique et d'une *clé*, on reconstruit le texte en clair à partir du texte chiffré.

clé

Paramètre d'un algorithme de chiffrement ou de déchiffrement, sur lequel repose le secret.

Décryptage

Action qui consiste à "casser" le chiffrement d'un texte de façon à retrouver le texte en clair sans connaître la *clé* qui permet son déchiffrement normal.

Cryptographie

Désigne l'ensemble des principes, moyens et méthodes (algorithmes ou systèmes de chiffrement) de transformation des données destinées à chiffrer leur contenu, établir leur authenticité, empêcher que leur modification passe inaperçue, prévenir leur répudiation ou leur utilisation non autorisée.

Cryptosystème

C'est un couple de deux algorithmes permettant d'effectuer respectivement le chiffrement et le déchiffrement associé.
Quelques procéssus sont illustrés par la *Figure 1.1* :

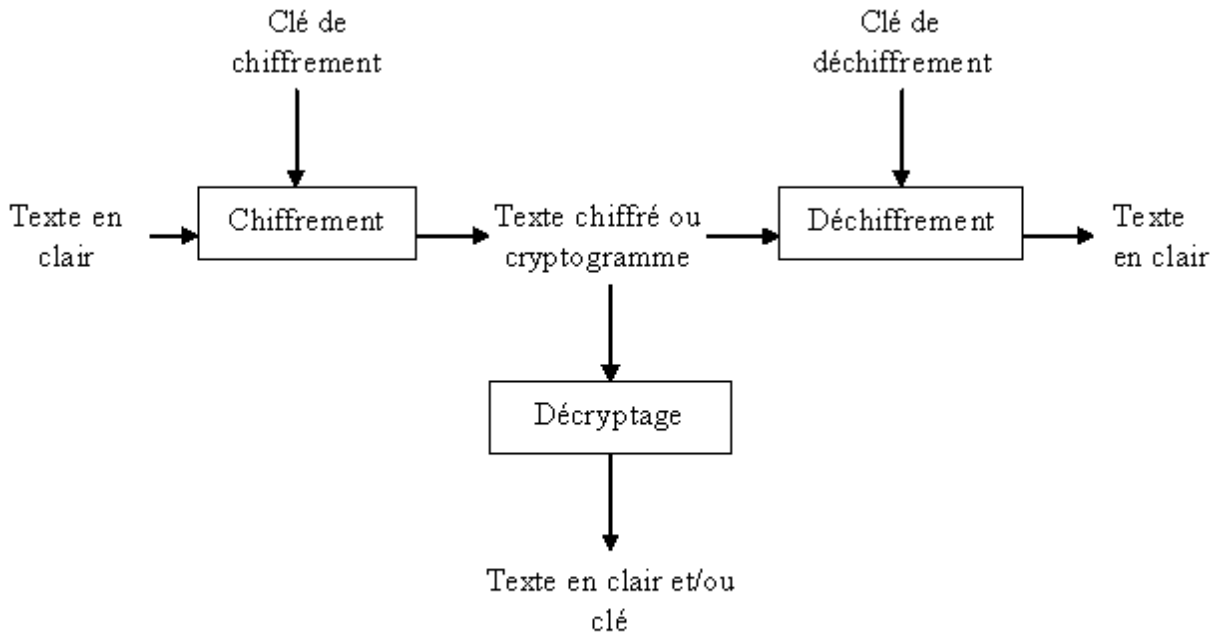


Figure 1.1 : Chiffrement, déchiffrement et décryptage

De nombreuses méthodes de chiffrement différentes ont été imaginées pour se protéger de la curiosité et de la malveillance de ses ennemis depuis de nombreux siècles. On peut classer ces méthodes en deux classes, comme le montre le schéma qui suit :

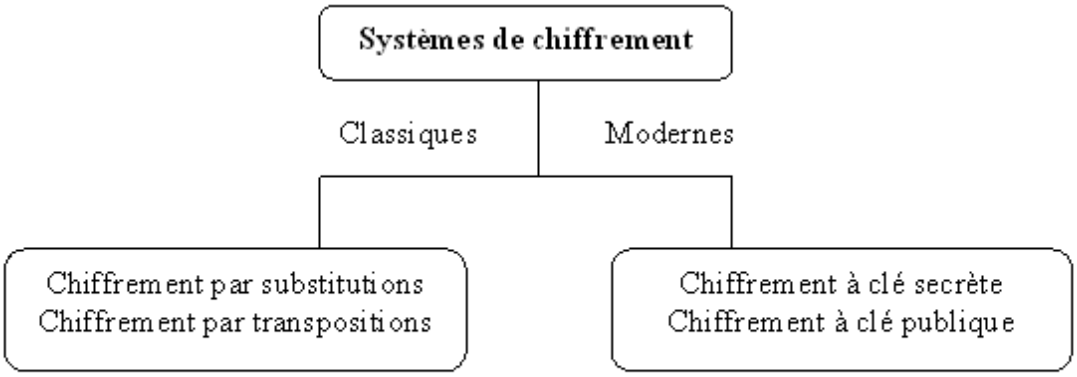


Figure 1.2 : Systèmes de chiffrement

Dans la littérature on trouve une autre classification de ces méthodes, qui considère les systèmes de chiffrement à clé secrète, comme des systèmes de chiffrement classiques. Définissons maintenant chaque classe.

1.3 La cryptographie classique

Elle décrit la période avant les ordinateurs. Elle traite des systèmes reposant sur les lettres et les caractères d'une langue naturelle. Les principaux outils utilisés remplacent des caractères par des autres et les transposent dans des ordres différents. Cela suppose que les procédures de chiffrement et de déchiffrement soient gardées secrètes, si non le système est complètement inefficace (n'importe qui peut déchiffrer le message chiffré).

1.4 La cryptographie moderne

Les méthodes utilisées de nos jours sont plus complexes, cependant la philosophie reste la même. La différence fondamentale est que les méthodes modernes (les algorithmes, puisque l'on utilise maintenant des ordinateurs) manipulent des bits contrairement aux anciennes méthodes qui opéraient sur des caractères alphabétiques. Ce n'est donc qu'un changement de taille (ou de représentation), puisque l'on utilise que deux éléments au lieu

des 26 lettres de l'alphabet. Les règles de chiffrement et de déchiffrement sont connues de tous, la sécurité de ces méthodes reposent maintenant uniquement sur la clé.

Le fait de rendre publiques les méthodes de chiffrement et de déchiffrement offre une certaine garantie sur la sécurité d'un système, dans la mesure où tout nouvel algorithme cryptographique est immédiatement confronté à la sagacité de la communauté scientifque.

Cette classe contient deux grands types d'algorithmes de chiffrement, les algorithmes à clé secrète et les algorithmes à clé publique. Chacun de ces algorithmes possède ses propres avantages et inconvénients.

1.4.1 La cryptographie à clé secrète

Les algorithmes de chiffrement à clé secrète (ou symétriques) sont ceux pour lesquels l'émetteur et le destinataire partagent une même clé secrète autrement dit, les clés de chiffrement et de déchiffrement sont identiques. La sécurité des données ne peut être atteinte qu'à condition qu'il existe un canal physiquement sûr, qui permet un échange préalable de clé. Par exemple : une valise diplomatique, rencontre sans témoins, échange de disquette de main en main.

1.4.2 La cryptographie à clé publique

La cryptographie à clé publique (ou asymétrique) évite le partage d'un secret entre les deux interlocuteurs. Dans un système de chiffrement à clé publique, chaque utilisateur dispose d'un couple de clés, une clé publique qu'il met en général à la disposition de tous dans un annuaire, et une clé secrète connue de lui seul.

Un grand avantage du système à clé publique est que la clé secrète correspondante à la clé publique d'une personne n'est pas partagée, elle ne quitte jamais son propriétaire. Il est donc facile de s'assurer qu'elle n'est pas compromise, mais il n'a pas apporté de solution parfaite parcequ'il est lent.

Dans la plupart des applications actuelles, la meilleure solution consiste à utiliser un système hybride ou mixte, qui combine les deux systèmes à la fois.

1.5 Objectifs de la cryptographie

Le but fondamental de la cryptographie est de respecter adéquatement les objectifs majeurs de la sécurité suivants :

1. confidentialité : Il s'agit de rendre la lecture du message inintelligible à des tiers non autorisés.

2. authentification : Il s'agit principalement de s'assurer que le correspondant connecté est bien le correspondant souhaité et de s'assurer du signataire de l'acte.

3. integrité : Il s'agit de s'assurer que le message n'a pas été modifié durant la transmission.

4. non répudiation : l'expéditeur ne peut pas nier, ultérieurement, avoir envoyé le message. Cet aspect est sous-entendu dans l'authentification.

1.6 Cryptanalyse

La cryptanalyse est la science qui étudie la sécurité des procédés cryptographiques pour tenter de trouver des faiblesses et pouvoir en particulier effectuer un décryptage avec succès.

Si le but de la cryptographie est d'élaborer des méthodes de protection, le but de la cryptanalyse est au contraire de casser ces protections. Une tentative de cryptanalyse d'un système est appelée une attaque, et elle peut conduire à différents résultats.

Parmi les types d'attaques cryptanalytiques nous considérons celles ci-dessus.

1.6.1 L'attaque à texte chiffré seulement

Pour cette attaque, le cryptanalyste, à partir d'un texte chiffré, recherche le texte clair et/ou la clé. Il procède par analyse des fréquences des lettres utilisées dans le texte chiffré. Cette technique ne fonctionne que pour la plupart des chiffrements classiques, les seuls permettant l'utilisation de l'analyse des fréquences.

1.6.2 L'attaque à texte en clair connu

Dans ce cas le cryptanalyste a non seulement accès aux texte chiffré mais aussi à un fragment du texte en clair correspondant, son but est alors de retrouver la ou les clés utilisées pour chiffrer ce messages ou un algorithme qui permet de déchiffrer n'importe quel nouveau message chiffré avec la même clé. Ce genre d'attaque est bien plus courant que ce que l'on pourrait penser. En effet, certaines structures sont caractéristiques des documents et peuvent plus ou moins facilement être analysées. Les en-têtes types, les formules de politesse dans les lettres permettent d'avoir facilement un ou des couples (texte clair - texte chiffré).

1.6.3 L'attaque à texte en clair choisi

Pour ce genre d'attaque, le cryptanalyste dispose du cryptosystème prêt à fonctionner. La clé a été entrée, mais elle est inaccessible. La différence principale avec une attaque à (texte en clair connu) est que l'ennemi peut choisir le texte qu'il veut chiffrer. Il peut donc créer des couples (texte clair - texte chiffré) basé sur des chaînes particulières, par exemple il peut essayer de chiffrer "aaaaa" et toutes sortes de messages qui serait plus faciles à décrypter pour essayer de découvrir la clé, ou du moins pour essayer de tirer le maximum d'information du système. Les algorithmes à clé publique permettent ce genre d'attaques : n'importe qui peut chiffrer des messages et disposer d'autant de couples (texte clair - texte chiffré) qu'il désire. Il est pourtant impossible de déchiffrer les messages puisque la clé privée reste secrète.

1.6.4 L'attaque exhaustive

Cette attaque est la plus sûre, elle marche presque à tous les coups. Le principe est simple, on dispose du cryptosystème et d'un texte chiffré et on essaye toutes les clés possibles jusqu'à trouver la bonne. Cette attaque n'est pourtant pas la plus utilisée car c'est la plus longue. C'est pourquoi elle sert de base de comparaison pour estimer la sécurité d'un cryptosystème : si un codage ne peut être cassé autrement que par une

attaque exhaustive (ou du moins pas plus rapidement), alors on l'estime sûr pour une clé suffisamment longue. Le seul cryptosystème qui résiste à cette attaque est le masque jetable (dans le cas idéal où la clé est de la même longueur que le texte et a été générée de manière totalement aléatoire).

1.6.5 Cryptanalyse différentielle

En 1990, deux chercheurs israéliens du Weitzmann Institute, Biham et Shamir, ont présenté une nouvelle attaque, la cryptanalyse différentielle. La meilleure attaque différentielle connue demande actuellement 2^{47} textes clairs choisis. La phase d'analyse calcule la clé à l'aide de ces données. Une propriété intéressante de cette attaque est qu'il est possible de l'utiliser même si le nombre de données disponibles est petit, en fait la probabilité de succès augmente linéairement avec ce nombre.

1.6.6 Cryptanalyse linéaire

Cette attaque utilise des approximations linéaires pour décrire les opérations conduisant au chiffré. Comme précédemment, plus le nombre d'essais augmente, plus la probabilité d'obtention de la clé devient forte.

Chapitre 2

Cryptographie classique

Sommaire

2.1 Introduction

Avant l'avènement des ordinateures, la cryptographie traitait des cryptosystèmes basés sur les lettres (ou caractères). Les différents algorithmes cryptographiques remplaçaient des caractères par d'autres ou transposaient les caractères. Les meilleurs systèmes faisaient les deux opérations plusieurs fois. Les caractères du texte chiffré sont généralement groupées par cinq.

Tous les chiffres présentés dans ce chapitre sont programmés, et le logiciel RLBS-CRYPTOOL permet leur utilisation pour le chiffrement et le déchiffrement.

2.2 Chiffres à substitution

Un chiffre à substitution remplace les lettres du message clair par des symboles (caractères, chiffres, signes, etc) définis à l'avance. En général, pour des raisons pratiques chaque lettre du message clair est remplacée par une lettre différente.

2.2.1 Substitutions monoalphabétiques

Se dit d'un chiffre où une lettre du message clair est toujours remplacée par le même symbole.

Dans cette catégorie, on peut citer le chiffre de César, de Polybe, les chiffres Hebreux, le chiffre de Wolseley et le chiffre affine.

Chiffre de César Rome au I^{er} siècle av. J.C

Le chiffre de César consiste simplement à décaler circulairement les lettres de l'alphabet de quelques crans vers la droite ou vers la gauche. Si d est le décalage et x est le rang d'un caractère donné alors le caractère chiffré est : $y = (x + d) \bmod 26$.

Exemple 2.2.1 *Décalons les lettres de trois rangs vers la droite comme le faisait César :*

Message clair : VENEZ DEMAIN

Message chiffré : YHQHC GHPDL Q

Le déchiffrement se fait caractère par caractère comme suit : $x = (y - d) \bmod 26$.

Algorithme Cesar;

Chiffrement

Entrées : Un message clair M et un décalage d

$y := (x + d) \bmod 26$

Sortie : Un message chiffré C

Déchiffrement

Entrées : Un message chiffré C et le décalage d

$x := (y - d) \bmod 26$

Sortie : Le message clair M

Chiffre de Polybe Grèce au I^{er} siècle av. J.C

Polybe historein Grec est à l'origine du premier procédé de chiffrement par substitution. C'est un système de transmission basé sur un tableau de 25 cases. contenant les lettres de l'alphabet, i confondu avec j. Chaque lettre est remplacée par ses coordonnées dans le tableau, en écrivant d'abord la ligne puis la colonne.
Ainsi : e=**15**, u=**45**

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

Polybe proposait de transmettre ces nombres au moyen de torches. Il faut une torche à droite et cinq à gauche pour transmettre la lettre "e" par exemple.

Ce procédé permettait donc de transmettre des messages sur de longues distances. On peut aussi transmettre les coordonnées des lettres en tapant des coups sur un mur, sur la tuyauterie, etc.

Remarquons que nous pouvons remplir le tableau de façon différente. Par exemple en commençant par remplir avec un mot clé (supprimer les lettres identiques de ce dernier), puis on complètera en respectant l'ordre alphabétique.

Exemple 2.2.2 *Mot clé : difficile*

Message clair : fuyez

Message chiffré : 13 45 54 21 55

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

On peut si l'on veut compliquer d'avantage le système de chiffrement en commençant la première lettre du mot de passe à partir d'une case définie comme une clé ajoutée au mot de passe.

Le déchiffrement utilise exactement le même algorithme, mais dans le sens inverse.

Algorithme Polybe;

Chiffrement

Entrées : Un message clair M et le tableau P de Polybe (construit à partir d'un mot clé)

$C_k := ij$

Sortie : Un message chiffré C

Déchiffrement

Entrées : Un message chiffré C et le tableau P de Polybe

$M_k := P[i, j]$

Sortie : Le message clair M

Le chiffre Atbesh

Le chiffre Atbesh consiste à inverser l'ordre des lettres de l'alphabet. Le mot "Atbesh" dérive du système qu'il désigne, puisqu'il est composé à partir des lettres **a**leph, **t**au, **be**th et **sh**in, les deux premières et les deux dernières de l'alphabet hébreux.

T

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
z	y	x	w	v	u	t	s	r	q	p	o	n	m	l	k	j	i	h	g	f	e	d	c	b	a

Exemple 2.2.3 *Message clair : bonjour tout le monde*

Message chiffré : ylmql figlf govnl mwv

Pour déchiffrer on utilise la table ci-dessus.

Algorithme Atbesh;

Chiffrement

Entrées : Un message clair M et un tableau T(2×26)

$C_k := T[2, j]$

Sortie : Un message chiffré C

Déchiffrement

Entrées : Un message chiffré C et un tableau T(2×26)

$M_k := T[1, j]$

Sortie : Le message clair M

Le chiffre Albam

Ce chiffre décale les lettres de l'alphabet de 13 positions. Il est réapparu en 1984 sous le nom de ROT13.

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
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

Exemple 2.2.4 *Message clair : bonjour tout le monde*

Message chiffré : obawb hegbi gyrzb aqr

On remarque que ce chiffre est un chiffre de César où le décalage vaut 13, donc pour déchiffrer on peut soit utiliser la table ci-dessus soit utiliser la fonction $x = (y-13) \bmod 26$.

Algorithme Albam;

Chiffrement

Entrée : Un message clair M

$y := (x + 13) \bmod n$

Sortie : Un message chiffré C

Déchiffrement

Entrée : Un message chiffré C

$x := (y - 13) \bmod n$

Sortie : Le message en clair

Le chiffre Atbah

Ce chiffre inverse l'alphabet par tranches de 9/9/8 comme le montre le tableau T suivant :

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
i	h	g	f	e	d	c	b	a	r	q	p	o	n	m	l	k	j	z	y	x	w	v	u	t	s

Les lettres "e" et "n" sont chiffrées par elles mêmes, elles sont dites **invariants**.

Les chiffres Atbesh, Albam et Atbah sont **reversibles**. Un chiffre est dit **reversible** si quand on l'applique deux fois à un message, on obtient le message initial.

Le déchiffrement se fait en utilisant la table T.

Algorithme Atbah;

Chiffrement

Entrées : Un message M et un tableau T(2×26)

$C_k := T[2, j]$

Sortie : Un message chiffré C

Déchiffrement

Entrées : Un message chiffré C et le tableau T(2×26)

$M_k := T[1, j]$

Sortie : Le message clair M

Le chiffre de Lord Garnet Joseph Wolseley $18^{ème}$ **siècle**

C'est un chiffre réversible, dont le principe est de supprimer une lettre de l'alphabet, en général le "J" (en anglais) et le "W" (en français) car ces dernières ont des fréquences faibles en anglais et en français respectivement. Dans la première ligne de la table, on écrit la clé, puis le reste de l'alphabet, sans la lettre supprimée. Ensuite, on écrit dans la deuxième ligne de la table la même séquence mais dans l'ordre inverse pour obtenir la table de substitution.

Exemple 2.2.5 *Clé : maison*

Message clair : tuez le

Message chiffré : nolme l

m	**a**	**i**	**s**	**o**	**n**	b	c	d	e	f	g	h	j	k	l	p	q	r	t	u	v	x	y	z
z	y	x	v	u	t	r	q	p	l	k	j	h	g	f	e	d	c	b	**n**	**o**	**s**	**i**	**a**	**m**

Algorithme Wolseley;

Chiffrement

Entrées : Un message M, et un tableau T(2×26) (construit à partir d'un mot clé)

$C_k := T[2, j]$

Sortie : Un message chiffré C

Déchiffrement

Entrées : Un message chiffré, et un tableau T(2×26) (construit à partir du mot clé)

$M_k := T[1, j]$

Sortie : Le message clair M

Le chiffre affine

L'idée est d'utiliser comme fonction de chiffrement une fonction affine du type $y = (ax + b) \bmod 26$, où a et b sont des constantes, et où x et y appartiennnent à $\mathbb{Z}/26\mathbb{Z}$ et correspondent à l'ordre des lettres de l'alphabet moins un ($A = 0$, $B = 1$, ...).
On remarque que si $a = 1$, alors on retrouve le chiffre de César où b représente le décalage.

Le paramètre "a" doit être choisi de manière à ce qu'il soit premier avec 26, car dans ce cas il admettera un inverse et ce d'après la relation de Bézout.

Exemple 2.2.6 *$a = 3$, $b = 11$*

Message clair	N	S	A
Ordres des lettres (x)	13	18	0
Ordres des lettres (y)	24	13	11
Message chiffré	Y	N	L

Le déchiffrement se fait en utilisant la fonction $x = a^{-1}(y - b) \bmod 26$.

Algorithme Affine;

Chiffrement

Entrées : Un message clair M, deux entiers a et b

$y := (ax + b) \bmod 26$

Sortie : Un message chiffré C

Déchiffrement

Entrées : Un message chiffré C, les entiers a et b

$x := a^{-1}(y - b) \bmod 26$

Sortie : Le message clair M

2.2.2 Décryptage d'une substitution simple

La structure du message quand il est chiffré avec une substitution monoalphabétique ou simple est conservée, c'est-à-dire, les lettres doublées sont doublées dans le chiffré, les lettres qui apparaissent le plus restent les plus courantes dans le chiffré.

Ce mode de chiffrement a résisté aux cryptanalystes jusqu'à ce que le savant arabe Abu Yusuf Ya'qub ibn Is-haq ibn as-Sabbah Oòmran ibn Ismaïl al-Kindi mette au point, au $IX^{ème}$ siècle, une technique appelée Analyse des fréquences dans son traité intitulé "Manuscrit sur le déchiffrement des messages cryptographiques".

Pour décrypter un message en utilisant l'analyse des fréquences on elimine tout d'abord les accents et les ponctuations du message, on met toutes les lettres en majuscules ou en minisclules ensuite on compte les apparitions de chaque lettre et on utilise les caractéristiques de la langue du clair pour déchiffrer le message sans la clé.

La *Figure 2.1* suivante montre les fréquences d'apparition des lettres dans quelques langues. Par exemple en français les lettres les plus fréquemment employées sont dans l'ordre EASIT.

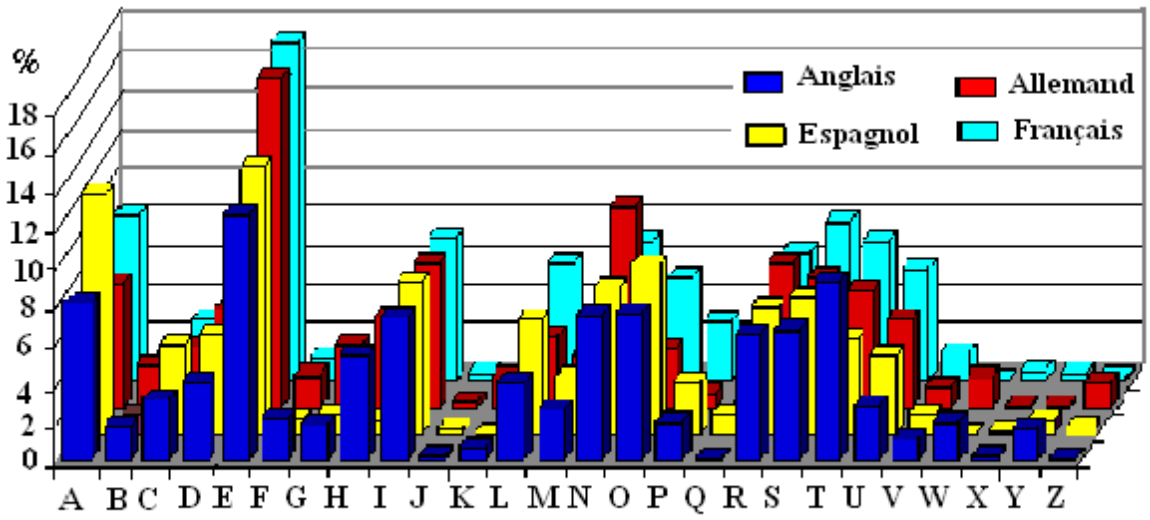


Figure 2.1 : Fréquences des lettres dans différentes langues

Cette technique ne fonctionne bien que si le message chiffré est suffisamment long pour avoir des valeurs significatives (au minimum 100 lettres, ou mieux, 500 lettres).

2.2.3 Substitutions polyalphabétiques

Se dit d'un chiffre où plusieurs alphabets de chiffrement sont utilisés en même temps. Par exemple le chiffre de Porta et le chiffre de Vigenère.

Le chiffre de Giovanni Battista Della Porta Italie 1563

Le physicien italien Giovanni Battista Della Porta fut l'inventeur du premier système pour lequel on change d'alphabet à chaque lettre. Ce système polyalphabétique était extrêmement robuste pour l'époque et a été utilisé avec succès pendant trois siècles, ce système est basé sur le tableau suivant :

AB	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
CD	a	b	c	d	e	f	g	h	i	j	k	l	m
	z	n	o	p	q	r	s	t	u	v	w	x	y
EF	a	b	c	d	e	f	g	h	i	j	k	l	m
	y	z	n	o	p	q	r	s	t	u	v	w	x
GH	a	b	c	d	e	f	g	h	i	j	k	l	m
	x	y	z	n	o	p	q	r	s	t	u	v	w
IJ	a	b	c	d	e	f	g	h	i	j	k	l	m
	w	x	y	z	n	o	p	q	r	s	t	u	v
KL	a	b	c	d	e	f	g	h	i	j	k	l	m
	v	w	x	y	z	n	o	p	q	r	s	t	u
MN	a	b	c	d	e	f	g	h	i	j	k	l	m
	u	v	w	x	y	z	n	o	p	q	r	s	t
OP	a	b	c	d	e	f	g	h	i	j	k	l	m
	t	u	v	w	x	y	z	n	o	p	q	r	s
QR	a	b	c	d	e	f	g	h	i	j	k	l	m
	s	t	u	v	w	x	y	z	n	o	p	q	r
ST	a	b	c	d	e	f	g	h	i	j	k	l	m
	r	s	t	u	v	w	x	y	z	n	o	p	q
UV	a	b	c	d	e	f	g	h	i	j	k	l	m
	q	r	s	t	u	v	w	x	y	z	n	o	p
WX	a	b	c	d	e	f	g	h	i	j	k	l	m
	p	q	r	s	t	u	v	w	x	y	z	n	o
YZ	a	b	c	d	e	f	g	h	i	j	k	l	m
	o	p	q	r	s	t	u	v	w	x	y	z	n

Porta emploie 13 alphabets différents et réversibles qu'il désigne par AB, CD, EF, etc.

Voici comment on procède : on convient à l'avance d'une clé, par exemple INOX. Ce mot-de-passe indiquera, pour chaque lettre, l'alphabet à utiliser. Pour chiffrer un mot,

on va donc voir pour chaque lettre, l'alphabet à utiliser. Une fois qu'on a l'alphabet, on remplace chaque lettre du mot à chiffrer par celle qui se trouve juste au-dessus ou en-dessous d'elle.

Exemple 2.2.7 *Soit à chiffrer le mot " bienvenue " avec la clé " INOX ". On procède comme suit. Pour le "b", on utilise l'alphabet "I", et en-dessous du "b", il y a un "x" : ce "x" sera la première lettre du message chiffré. Ensuite pour le "i", avec l'alphabet "N", cela nous donne "p". Et ainsi de suite ...*

Clair	b	i	e	n	v	e	n	u	e
clé	I	N	O	X	I	N	O	X	I
Chiffré	x	p	x	l	m	y	h	f	n

Pour ce chiffre on peut utiliser 13 alphabets répartis aléatoirement.

Le déchiffrement utilise exactement le même algorithme.

Algorithme Porta;

Chiffrement

Entrées : Un message clair M, un tableau$(13 \times 2, 14)$

(construit à partir d'une clé)

$C_k := T_{k,j}$

(*où $T_{k,j}$ est la lettre au-dessus ou en dessous de M_k en utilisant le $j^{ème}$ alphabet*)

Sortie : Un message chiffré C

Déchiffrement

Entrées : Un message clair M et le tableau du chiffrement

$M_k := T_{k,j}$

(*où $T_{k,j}$ est la lettre au-dessus ou en dessous de C_k en utilisant le $j^{ème}$ alphabet*)

Sortie : Le message clair M

Chiffre de Vigenère France au $XVI^{ème}$ siècle

Blaise de Vigenère, né en 1523, fut l'initiateur d'une méthode de chiffrement qui domina 3 siècles. C'est en 1586 qu'il publie son nouveau chiffre.

L'idée de Vigenère est d'utiliser un chiffre de César, mais où le décalage utilisé change de lettre en lettre.

Pour chiffrer un message, on écrit la clé sous le message à chiffrer, en la répétant aussi souvent que nécessaire pour que sous chaque lettre du message à coder, on trouve une lettre de la clé qui définit le décalage associé (a: décalage de 0 cran, b: 1 cran, c: 2 crans, ..., z: 25 crans).

Clair	c	r	y	p	t	o	g	r	a	p	h	i	e
clé	m	a	t	h	w	e	b	m	a	t	h	w	e
Décalage	12	0	19	7	22	4	1	12	0	19	7	22	4
Chiffré	o	r	r	w	p	s	h	d	a	i	o	e	l

Le déchiffrement utilise exactement le même algorithme, mais dans le sens inverse.

Algorithme Vigenère;

Chiffrement

Entrées : Un message clair M et un mot de passe

$C_k := (M_k + d_k) \bmod 26$

(*où M_k est le rang de la $k^{ème}$ lettre clair et d_k le décalage correspondant*)

Sortie : Un message chiffré C

Déchiffrement

Entrées : Un message chiffré et un mot de passe

$M_k := (C_k - d_k) \bmod 26$

(*où C_k est le rang de la $k^{ème}$ lettre chiffrée et d_k le décalage correspondant*)

Sortie : Le message clair M

Chiffre de Vernam

Il existe un algorithme de chiffrement parfaitement sûr (il est d'ailleurs unique, comme l'a prouvé Shannon en 1949, voir [10]). Et ne croyez pas que cette méthode de chiffrement soit horriblement complexe. Elle a été mise au point par Gilbert Vernam en 1917, et est simplement un chiffre de Vigenère, mais où la clé est de la taille du message à envoyer,

et où les lettres de cette clé sont choisies aléatoirement. Si la clé ne sert qu'une seule fois (chiffre à usage unique ou masque jetable "one-time pad" en Anglais), ce système est absolument sûr : il n'y a aucune corrélation entre le message de départ et sa version chiffrée. En effet, quelque soit le texte que l'on veuille obtenir à partir d'un certain message chiffré, il existe toujours une clé correspondante, et toutes les clés ont la même probabilité de sortir puisqu'elles sont générées aléatoirement.

Clair	m	a	s	q	u	e	j	e	t	a	b	l	e
clé	t	b	f	r	g	f	a	r	f	m	i	k	l
Chiffré	f	b	x	h	a	j	j	v	y	m	j	v	p

Bien sûr, les inconvénients de ce mode de chiffrement est la génération et le transport des clés. Cette clé doit être parfaitement aléatoire et aussi longue que le message, tout cela pour une seule utilisation! Tout le monde ne dispose pas de la valise diplomatique pour pouvoir échanger la clé.

Algorithme Vernam;

Chiffrement

Entrées : Un message clair M et une clé

$C_k := (M_k + d) \bmod 26$

(*où M_k est le rang de la $k^{ème}$ lettre clair et d le décalage aléatoire correspondant*)

Sortie : Un message chiffré C

Déchiffrement

Entrées : Un message chiffré et une clé

$M_k := (C_k - d) \bmod 26$

(*où C_k est le rang de la $k^{ème}$ lettre chiffrée et d le décalage aléatoire correspondant*)

Sortie : Le message clair M

2.2.4 Décryptage du chiffre de Vigenère

Le test de Kasiski

Le système polyalphabétique de Vigenère résista pendant environ 3 siècles, jusqu'à ce que l'ancien major de l'armée prusse, Friedrich Kasiski, publie sa méthode en 1863.

L'idée de Kasiski est de chercher les séquences de 3 lettres répétées dans le texte chiffré, et de se dire que ces répétitions ne sont pas fortuites. Si une séquence de 3 lettres est répétée dans le message chiffré avec une distance d, on peut dire qu'il s'agit de la même séquence de 3 lettres du texte initial, chiffrée avec la même séquence de lettres de la clé. Par conséquent, si m est la longueur de la clé, pour que les 2 séquences soient chiffrées avec les mêmes lettres de la clé, il faut que m divise d.

On prend donc m comme le plus grand commun diviseur des distances de séquences répétées en ne considerant que les valeurs significatives (m est la longueur de la clé!).

Exemple 2.2.8 $m = 3$

Cela signifie que les caractères de rangs $1, 4, 7, 10, ..., 3k + 1$*, sont simplement décalés à la manière du chiffre de César. On peut donc appliquer maintenant l'analyse des fréquences à ces caractères et trouver la première lettre de la clé. Pour la deuxième lettre de la clé, on analysera les fréquences des caractères de rang* $3k + 2$ *et pour la dernière lettre les fréquences des caractères de rang* $3k$*.*

2.2.5 Chiffres tomogrammiques

Dans les systèmes tomogrammiques, chaque lettre est tout d'abord représentée par un groupe de symboles. Ces symboles sont ensuite chiffrés séparément ou par groupes de taille fixe.

Chiffre de Delastelle France 1895

Le chiffre de Delastelle du nom de son inventeur, le Français Félix-Marie Delastelle (1840-1902), qui en avait décrit pour la première fois le principe dans la Revue du Génie civil en 1895, sous le nom de "Cryptographie nouvelle". Il commence par regrouper les lettres du message à chiffrer cinq par cinq (au besoin, on rajoute des nulles pour que la longueur du message soit un multiple de 5) puis il utilise un carré de Polybe. Il repère les coordonnées de plusieurs lettres claires, mélange ces coordonnées, puis lit les lettres chiffrées correspondant aux nouvelles coordonnées obtenues.

Exemple 2.2.9 *Si on souhaite chiffrer le message : je te montre, on procède comme suit. On construit le carré de Polybe avec la clé JOUR :*

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

On regroupe les lettres cinq par cinq :

Clair	j	e	t	e	m	o	n	t	r	e
Ligne	1	2	5	2	4	1	4	5	1	2
Colonne	1	4	1	4	1	2	2	1	4	4

On regroupe alors les chiffres deux par deux, de la gauche vers la droite, puis du haut vers le bas :

Coordonnées chiffrées	12	52	41	45	12	14	14	12	21	44
message chiffré	o	v	m	s	o	r	r	o	b	q

Le dechiffrement opère dans le sens inverse.

Algorithme Delastelle;

Chiffrement

Entrées : Un message clair M et le tableau P de Polybe (construit à partir d'une clé)

Substitution (*de type carré Polybique*)

Regroupement des chiffres

Substitution inverse de Polybe

Sortie : Un message chiffré C

Déchiffrement

Entrées : Un message chiffré et le tableau P de Polybe

Substitution inverse de Polybe

Regroupement des chiffres

Substitution (*de type carré Polybique*)

Sortie : Le message clair M

Le chiffre Digrafide

C'est un chiffre tomogrammique qui utilise un tableau contenant deux alphabets désordonnés representés dans deux rectangles contenant toutes les lettres et un carré dont on trouve les chiffres de 1 à 9. Ces trois tableaux sont disposés comme on le montre ci-dessous. Les lettres du message clair sont chiffrées par bigrammes (groupe de deux lettres). Dans une première étape, ces bigrammes sont transformés en nombres de trois chiffres. Après un mélange simple, ces nombres sont retransformés en lettres et donnent les bigrammes chiffrés.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
	P	E	L	A	S	B	C	D	F
T	G	H	I	J	K	M	N	O	Q
	R	T	U	V	W	X	Y	Z	#

Grille 1

Grille 2

1	**2**	**3**
4	**5**	**6**
7	**8**	**9**

Grille 3

M	C	R	**1**
E	F	T	**2**
L	G	U	**3**
I	H	V	**4**
S	J	W	**5**
A	K	X	**6**
N	O	Y	**7**
D	P	Z	**8**
B	Q	#	**9**

Exemple 2.2.10 *Chiffrons le message: "LASSASSIN EST KHALED"*

1. *On commence par grouper les lettres du message clair six par six (au besoin, on rajoute des nulles pour que la longueur du message soit un multiple de 6).*

2. *On chiffre les lettres du message clair par bigramme. Pour chiffrer le bigramme LA, on repère le chiffre au-dessus du L dans la Grille 1 (3), le chiffre dans la Grille 2 qui est sur la même ligne que le L de la Grille 1 et sur la même colonne que le A de la Grille 3 (1), puis le chiffre à droite de A de la Grille 3 (6). On écrit le nombre obtenu (316) verticalement.*

3. *On procède de même avec les deux bigrammes restants. On obtient trois nombres à trois chiffres que l'on lit horizontalement, chacun de ces nombres correspond à un bigramme chiffré.*

LA	SS	AS	SI	NE	ST	KH	AL	ED
3	5	4	5	7	5	5	4	2
1	1	1	1	4	3	5	1	1
6	5	5	4	2	2	4	3	8
354	111	655	575	143	422	542	511	438
IH	PM	MJ	WS	GL	AF	KE	SM	AZ

Le message chiffré est : IHPMM JWSGL AFKES MAZ

Pour le déchiffrement on procède exactement comme pour le chiffrement.

Algorithme Digrafide;

Chiffrement

Entrées : Un message clair M et le tableau T

Transformation des lettres en chiffres

Transformation des chiffres en d'autres lettres

Sortie : Un message chiffré C

Déchiffrement

Entrées : Un message chiffré C et le tableau T

Transformation des lettres en chiffres

Transformation des chiffres en d'autres lettres

Sortie : Le message clair M

Le chiffre ADFGVX 1918

Ce chiffre est constitué d'une substitution de type carré de Polybe, suivie d'une transposition. Pour réaliser la substitution, les 26 lettres de l'alphabet et les 10 chiffres sont rangés dans un tableau 6×6, aux extrémités desquelles on ajoute les lettres ADFGVX.

	A	**D**	**F**	**G**	**V**	**X**
A	Q	Y	A	L	S	E
D	Z	C	R	X	H	0
F	F	O	4	M	8	7
G	3	I	T	G	U	K
V	P	D	6	2	N	V
X	1	5	J	9	W	B

Chaque caractère est remplacé par le couple de lettres qui correspond à sa ligne et à sa colonne. Ainsi, R est remplacée par DF, et le message RENCONTRE USTHB 10H20 devient : DFAXV VDDFD VVGFD FAXGV AVGFD VXXXX ADXDV VGDX
On choisit ensuite, pour faire la transposition une clé, par exemple DEMAIN. On écrit le texte intermédiaire sous ce mot, puis on réordonne les colonnes par ordre alphabétique croissant :

D	**E**	**M**	**A**	**I**	**N**
D	F	A	X	V	V
D	D	F	D	V	V
G	F	D	F	A	X
G	V	A	V	G	F
D	V	X	X	X	X
A	D	X	D	V	V
G	D	X			

$\rightarrow$

A	**D**	**E**	**I**	**M**	**N**
X	D	F	V	A	V
D	D	D	V	F	V
F	G	F	A	D	X
V	G	V	G	A	F
X	D	V	X	X	X
D	A	D	V	X	V
	G	D		X	

Il ne reste plus qu'à lire le tableau de gauche à droite, et de haut en bas pour obtenir le message chiffré : XDFVA VDDDV FVFGF ADXVG VGAFX DVXXX DADVX VGDX

Algprithme ADFGVX;

Chiffrement

Entrées : Un message clair M, un carré de Polybe et une clé

Substitution (*de type carré de Polybe*)

Transposition (*en utilisant la clé*)

Sortie : Un message chiffré C .

Déchiffrement

Entrées : Un message chiffré C un carré de Polybe la clé du chiffrement

Transposition inverse

Substitution inverse

Sortie : Le message clair M

2.2.6 Chiffres polygrammiques

Se dit d'un chiffre où un groupe de n lettres est chiffré par un autre groupe de n lettres. Exemples : le chiffre de Playfair, chiffrement à deux carrés, chiffrement à trois carrés, chiffrement à quatre carrés et le chiffre de Hill.

Le chiffre de Playfair 1854

On dispose les 25 lettres de l'alphabet (W exclu on utilise V à sa place, la variante anglaise consiste à garder le W et à fusionner I et J) dans une grille 5×5, Pour former les grilles de chiffrement, on utilise une clé pour créer un alphabet désordonné avec lequel on remplit la grille ligne par ligne. Ensuite, on comble la grille avec les lettres de l'alphabet restantes.

On chiffre le texte clair par bigrammes en appliquant les règles suivantes.
Si les 2 lettres sont :

1. sur les coins d'un rectangle, alors les lettres chiffrées sont sur les deux autres coins.

La première des deux lettres chiffrées est sur la même ligne que la première lettre claire.

2. sur la même ligne, on prend les deux lettres qui les suivent immédiatement à leur

droite.

3. sur la même colonne, on prend les deux lettres qui les suivent immédiatement en dessous.

4. identiques, on insère une nulle (usuellement le x) entre les deux pour éliminer ce doublon, « message » devient « me sx sa ge ».

Exemple 2.2.11 *OK devient VA, FJ sera remplacé par US et RM par ID*

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

Règle 1

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

Règle 2

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

Règle 3

Pour déchiffrer on applique les mêmes règles dans l'autre sens.

Algorithme Playfair;

Chiffrement

Entrées : Un message clair M et une clé avec laquelle on construit un alphabet désordonné

Substitution en utilisant les règles

Sortie : Un message chiffré C

Déchiffrement

Entrées : Un message chiffré C et le mot clé

Substitution en utilisant les règles

Sortie : Le message clair M

Le chiffrement à trois carrés

On utilise trois grilles carrées de dimensions 5×5 (dans la version française, on élémine le W qui sera le cas échéant remplacé par le V, en anglais on préfère supprimer le J) que l'on remplit avec un alphabet désordonné en utilisant des clés.

On chiffre les lettres du message clair par bigrammes pour obtenir des trigrammes. À titre d'exemple, considérons les grilles ci-dessous et chiffrons le bigramme **BA**. On repère le **B** dans la grille 1; la première lettre du trigramme sera une lettre quelconque choisie dans la même colonne que le **B** dans la grille 1. On repère le **A** dans la grille 2; la dernière lettre du trigramme sera une lettre quelconque choisie dans la même ligne que le **A** dans la grille 2. La lettre du milieu du trigramme sera la lettre de la grille 3 qui se trouve sur la même ligne que le **B** et la même colonne que le **A**. Ainsi, le bigramme clair **BA** deviendra par exemple le trigramme chiffré **ROR**.

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

Pour déchiffrer, on repère dans la grille 1 la colonne de la lettre **R**. On repère dans la grille 2 la ligne de la lettre **R**. On repère dans la grille 3 la lettre **O**. L'intersection de la ligne du **O** avec la colonne du **R** donne la première lettre du bigramme **B**. L'intersection de la colonne du **O** avec la ligne du **R** donne la deuxième lettre du bigramme **A**.

Algorithme Trois carrés;

Chiffrement

Entrées : Un message clair M et trois clés avec lesquelles on construit des alphabets désordonnés

Remplacement des bigrammes par des trigrammes

Sortie : Un message chiffré C

Déchiffrement

Entrées : Un message chiffré C et les clés du chiffrement

Remplacement des trigrammes par des bigrammes

Sortie : Le message clair M

Le chiffre de Hill 1929

Le chiffre publié en 1929 par Lester S. Hill est un chiffre polygrammique (comme le chiffre Playfair), c'est-à-dire qu'on ne chiffre pas les lettres les unes après les autres, mais par paquets. Nous étudierons ici la version bigrammique de ce chiffre, qui consiste à regrouper les lettres deux par deux, mais on peut imaginer des paquets plus grands, par exemple des paquets de trois lettres.

Les lettres sont d'abord remplacées par leur rang dans l'alphabet. Les lettres P_k et P_{k+1} du texte clair seront chiffrées C_k et C_{k+1} avec la formule ci-dessous :

$$\begin{pmatrix} C_k \\ C_{k+1} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} P_k \\ P_{k+1} \end{pmatrix} \bmod 26$$

On ne peut pas prendre n'importe quoi comme matrice de chiffrement, il faut qu'elle ait une matrice inverse dans $\mathbb{Z}/\mathbb{Z}26$. La taille de la matrice peut varier, elle n'est pas fixée à 2.

Les deux premières lettres du message clair (P_1 et P_2) seront chiffrées (C_1 et C_2) selon les deux équations suivantes :

$$\begin{aligned} C_1 &= aP_1 + bP_2 \,(\mathrm{mod}\, 26) \\ C_2 &= cP_1 + dP_2 \,(\mathrm{mod}\, 26) \end{aligned}$$

Exemple 2.2.12 *Message clair : election*

Clé : $\begin{pmatrix} 3 & 5 \\ 1 & 2 \end{pmatrix}$

Après avoir remplacé les lettres par leur rang dans l'alphabet moins 1 on obtient :

$$\begin{aligned} C_1 &= 3.4 + 5.11 \,(\mathrm{mod}\, 26) = 67(\mathrm{mod}\, 26) = 15 \\ C_2 &= 1.4 + 2.11 \,(\mathrm{mod}\, 26) = 26(\mathrm{mod}\, 26) = 0 \end{aligned}$$

On fera de même avec les lettres restantes. On obtiendra finalement :

Lettres	e	l	e	c	t	i	o	n
Rangs (P_k)	4	11	4	2	19	8	14	13
Rangs chiffrés (C_k)	15	0	22	8	19	9	3	14
Lettres chiffrées	p	a	w	i	t	d	j	o

Remarque 2.2.1 *Si la taille du message est impair on ajoute une nulle.*

Pour déchiffrer, le principe est le même que pour le chiffrement : on prend les lettres deux par deux, puis on les multiplie par la matrice inverse de la matrice de chiffrement comme suit :

$$\begin{pmatrix} P_k \\ P_{k+1} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} \begin{pmatrix} C_k \\ C_{k+1} \end{pmatrix} \bmod 26 = \frac{1}{ad-bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} \bmod 26$$

Notons que la division par $\frac{1}{ad-bc}$ est en fait la multiplication par $(ad-bc)^{-1}$ modulo 26.

Algorithme Hill;

Chiffrement

Entrées : Un message clair et une matrice inversible à éléments dans $\mathbb{Z}/\mathbb{Z}26$

$C_k = aP_k + bP_{k+1} \pmod{26}$

$C_{k+1} = cP_k + dP_{k+1} \pmod{26}$

Sortie : Un message chiffré C

Déchiffrement

Entrées : Un message chiffré C et la matrice du chiffrement

$P_k = a'C_k + b'C_{k+1} \pmod{26}$

$P_{k+1} = c'C_k + d'C_{k+1} \pmod{26}$

Sortie : Le message clair M

2.3 Chiffres à transposition

La transposition consiste à changer l'ordre des lettres du message clair.

2.3.1 La scytale Grèce au $VII^{ème}$ siècle avant JC

L'expéditeur enroulait une bande de cuir sur un bâton de diamètre défini, appelé la scytale (mot qui signifie « bâton » en grec). Il écrivait ensuite son message transversalement dans le sens du fameux bâton ou scytale. En déroulant la bande, le message devenait inintelligible, sauf pour la personne qui possédait un bâton de même diamètre.

Voici une représentation de la scytale.

Figure 2.2 :Représentation de la scytale

2.3.2 Rail Fence

Une forme de transposition parmi les plus élémentaires est nommée Rail Fence. Elle consiste à écrire les caractères en zig-zag horizontalement d'une hauteur prédéterminée appellée niveau (qui représente la clé). Ensuite, il suffit de lire les caractères normalement, ligne par ligne.

Exemple 2.3.1 *Texte clair : VIENS ME REJOINDRE..*

Le Rail Fence à deux niveaux dispose les lettres en zig-zag: il s'agit d'écrire le message sur deux lignes, une lettre sur une ligne et la suivante sur l'autre.

V E S E E O N R

I N M R J I D E

Nous obtiendrons alors, en écrivant les deux lignes à la suite l'une de l'autre :

VESEE ONRIN MRJID E.

Le déchiffrement se fait en écrivant le message sur deux lignes ensuite le lire en zig-zag.

Algorithme Rail Fence;

Chiffrement

Entrées : Un message clair M et le nombre de lignes (ou de niveaux) l

Remplissage du tableau à l lignes par colonnes

Lecture du tableau ligne par ligne

Sortie : Un message chiffré C

Déchiffrement

Entrées : Un message chiffré C et le nombre de lignes l

Remplissage du tableau à l lignes (ligne par ligne)

Lecture du tableau colonne par colonne

Sortie : Le message clair M

Remarque 2.3.1 *On peut aussi chiffrer avec un Rail Fence à trois niveaux ou plus. Les lettres du message VIENS ME REJOINDRE se disposent comme suit quand le nombre de niveaux vaut trois :*

V S E N

I N M R J I D E

E E O R

Le message chiffré serait : VSENI NMRJI DEEEO R.

2.3.3 La grille tournante

On peut utiliser une grille $n \times n$ avec n pair dans laquelle sont percés $n^2/4$ trous (cette grille trouée est appelée cache et représente la clé) de telle sorte que lorsque l'on fait faire 1/4 de tour à la grille, les trous ne se superposent pas. Ce procédé été inventé par le colonel autrichien Edouard Fleissner von Wostrowitz au XVI^{ème} siècle.

Les lettres du message clair sont écrites successivement dans les trous de la grille en lui faisant faire trois fois 1/4 de tour.
On obtient une grille remplie de lettres dans un ordre incompréhensible.

Exemple 2.3.2 *Dans cet exemple, on utilise une grille* 6×6 *à* 9 *trous pour chiffrer le message "JE CHIFFRE CE TEXTE AVEC UN CACHE TOURNANTZ". On pose le cache, puis on remplit les cases découvertes (trous) avec les 9 premières lettres du message (la lettre* **r** *de la grille sert de repère pour voir comment le cache tourne). On tourne ensuite le cache d'un quart de tour, puis on écrit dans les cases découvertes les 9 lettres suivantes du message, et ainsi de suite. On obtient une grille remplie de lettres dans un ordre incompréhensible.*

r	J		E		C
				H	
		I			
	F			F	
					R
			E		

↷

					r
		C			E
			T		
E					X
		T		E	
	A				V

↷

		E			
C					
	U			N	
			C		
	A				
C		H		E	**r**

↷

T				O	
	U		R		
N					A
		N			
T			Z		
r					

$\longrightarrow$

T	J	E	E	O	C
C	U	C	R	H	E
N	U	I	T	N	A
E	F	N	C	F	X
T	A	T	Z	E	R
C	A	H	E	E	V

On écrit ensuite les caractères ligne par ligne pour obtenir le message chiffré :
TJEEO CCUCR HENUI TNAEF NCFXT ATZER CAHEE V

Remarque 2.3.2 *Si le nombre de cases de la grille est supérieur à celui des lettres du message, on peut compléter avec des lettres prises au hasard, sinon on peut soit réutiliser la même grille plusieurs fois soit changer la grille complètement.*

Algorithme Grille tournante;

Chiffrement

Entrées : Un message clair M et un cache

Remplissage des trous en faisant des rotations

Lecture de la grille remplie ligne par ligne

Sortie : Un message chiffré C

Déchiffrement

Entrées : Un message chiffré C et le cache du chiffrement

Remplissage de la grille ligne par ligne

Lecture des lettres à partir des trous du cache en faisant des rotations

Sortie : Le message clair M

2.3.4 La transposition rectangulaire

Une transposition rectangulaire consiste à écrire le message chiffré horizentalement dans une grille de largeur fixe et égale à la longueur d'une clé donnée, puis à arranger les colonnes de cette grille selon les rangs des lettres de la clé.

Exemple 2.3.3 *Message clair : attendons vos instructions*

Clé : grain

En remplissant la grille, on constate qu'il reste une case vide, que l'on peut remplir avec une nulle ou pas.

g	**r**	**a**	**i**	**n**
a	t	t	e	n
d	o	n	s	v
o	s	i	n	s
t	r	u	c	t
i	o	n	s	

→

a	**g**	**i**	**n**	**r**
t	a	e	n	t
n	d	s	v	o
i	o	n	s	s
u	t	c	t	r
n	i	s		o

On relève le texte chiffré horizentalement, on obtient : taent ndsvo ionss utctr niso

Algorithme Transposition rectangulaire;

Chiffrement

Entrées : Un message clair M et une clé de taille t

Remplissage du tableau à t colonnes ligne par ligne

Arrangement du tableau

Sortie : Un message chiffré C

Déchiffrement

Entrées : Un message chiffré C et la clé du chiffrement

Arrangement inverse du tableau

Lecture du tableau ligne par ligne

Sortie : Le message clair M

2.3.5 Technique du mot probable (Cribbing)

Une technique très puissante de décryptage consiste à supposer qu'une séquence de lettres du cryptogramme correspond à un mot que l'on devine (crib en anglais). Ce type d'attaque marche aussi bien pour les substitutions simples que pour le chiffre de Vigenère, le chiffre de Hill et aussi contre la grille tournante.

2.4 La guerre 39-45 et la machine Enigma

2.4.1 Historique

C'est autour de 1919 que deux inventeurs allemands, Arthur Scherbius et Richard Ritter, créèrent une machine à chiffrer électromécanique, la machine Enigma. Elle automatise le chiffrement par substitution et a été utilisée, avec plusieurs améliorations pendant la seconde guerre mondiale par l'armée allemande convaincue de l'inviolabilité de la machine, alors que quelques scientifiques polonais ont réussi à obtenir des informations sur le système employé, puis plus d'une dizaine de milliers de chercheurs alliés ont tenté de casser le code allemand. Finalement l'équipe d'Alain Turing a terminé de résoudre le problème, et certains estiment que ce succès a permis de racourcir la guerre de deux ans, économisant des milliers de vies. Ceci a fait prendre conscience à l'humanité de l'importance et des enjeux de la cryptographie.

2.4.2 Description de la machine Enigma

Enigma se présente sous la forme d'une caisse en bois de $34 \times 28 \times 15$ cm, et pèse une douzaine de kilos.
Elle ressemble à une machine à écrire, elle était constituée :

- d'un clavier pour les 26 lettres de l'alphabet,
- d'un tableau de connexion qui attribue à chaque lettre de l'alphabet une autre,
- de 3 rotors qui établissent des connexions entre différentes lettres de l'alphabet et effectuent des rotations,
- et d'un réflecteur et d'un tableau de 26 ampoules correspondant aux 26 lettres de

l'alphabet.

Figure 2.3 : La machine Enigma

2.4.3 Fonctionnement de la machine Enigma

Le principe de fonctionnement d'Enigma est à la fois simple et astucieux. Quand on presse sur une touche, deux choses se passent. Premièrement, un circuit électrique est fermé et une lettre s'allume sur le panneau lumineux : c'est la lettre chiffrée. Deuxièmement, un mécanisme fait tourner le rotor de droite d'un cran, toutes les 26 frappes, le deuxième rotor tourne d'un cran, toutes les 676 frappes (26×26) c'est le troisième rotor qui tourne d'un cran. Ces rotors tournants modifient les connexions électriques dans la machine, ce qui fait que la touche "A" allumera peut-être le "B" la première fois, mais le "X" la deuxième et le "E" la troisième, etc. De plus le chiffrement est réversible, si A est chiffrée D, alors D est chiffrée A.

Sous chaque lettre étaient disposés deux contacts électriques, l'un à gauche, l'autre à droite. Nous appelleront Ag et Ad les contacts gauche et droite situées sous la lettre A.

Suivons sur le schéma ci-dessous, où la machine est en position AAA (chaque rotor est à la position A), ce qui se passe lorsque la touche A est enfoncée. Le courant entre en Ag, et ressort du rotor I en Dd, pour sortir du rotor II en Bd et fini par allumer la lampe correspondant à la lettre C. En respectant la position des rotors, le système peut être utilisé pour chiffrer et déchiffrer un message.

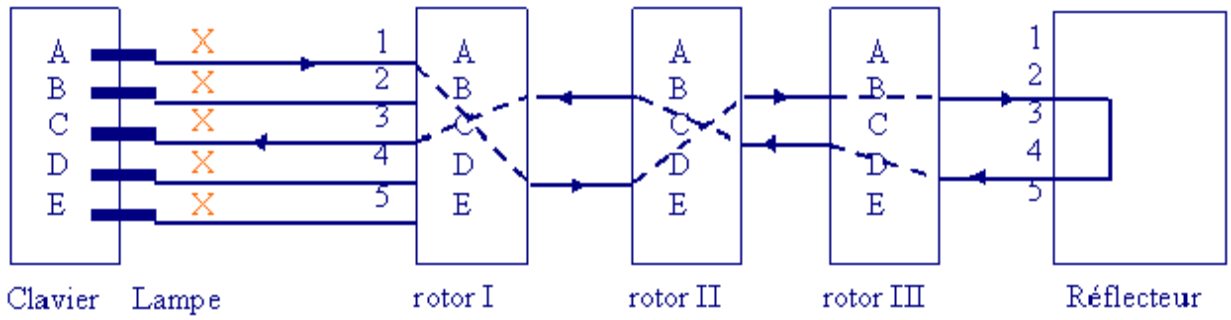


Figure 2.4 : Enigma en position AAA

Une fois la touche A enfoncée, la lampe C s'est allumée et le rotor I a avancé d'un cran. La position de la machine devient en BAA. Si la touche A est à nouveau enfoncée, c'est la lampe E qui va s'allumer.

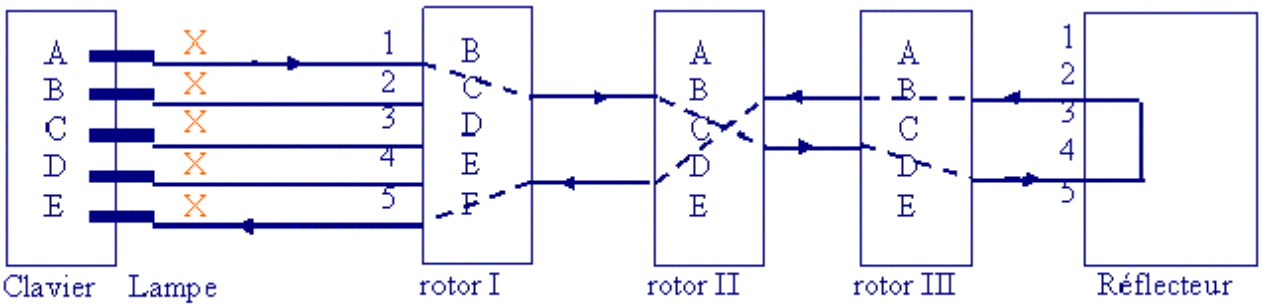


Figure 2.5 : Enigma en position BAA

2.4.4 Chiffrement et déchiffrement avec Enigma

Pour chiffrer un message, on choisit une position des rotors : par exemple en position WGJ qui constitue la clé du chiffrement. Ensuite on tape le message lettre par lettre en notant à chaque frappe la lettre qui s'allume, et on envoie le message chiffré obtenu.

Pour déchiffrer le destinataire positionne les rotors sur WGJ, tape le texte reçu en notant à chaque fois la lettre qui s'allume.

Notons q'un simulateur Enigma appelé "enigmars" téléchargé du site

http://users.telenet.be/d.rijmenants/en/enigmasim.htm a été intégré dans le logiciel RLBS-CRYPTTOOL.

Chapitre 3

Éléments de mathématiques

Sommaire

3.1 Introduction

La plupart des systèmes cryptographiques à clé publique sont basés sur des techniques mathématiques, qu'elles soient arithmétiques, algébriques ou analytiques. Ce chapitre rassemble les notions auxquelles on fera appel par la suite.

3.2 Eléments d'arithmétique

3.2.1 Division euclidienne

Soit a et b deux entiers naturels avec $a > b$. Il existe un couple unique d'entiers naturels q et r (avec $0 \leq r < b$) verifiant $a = bq + r$, le nombre q est appelé le quotient de la division euclidienne de a par b et r le reste.

3.2.2 Nombre premier

Un nombre p est premier s'il admet exactement deux diviseurs, 1 et lui même (cette définition exclu 1 de la liste des nombres premiers), s'il en admet plus de deux il sera dit composé.

Théorème 3.2.1 *(Fondamental de l'arithmétique) [21] Tout entier naturel $n \geq 2$ peut s'écrire de façon unique en un produit de nombres premiers, appelée décomposition en facteurs premiers de n.*

Théorème 3.2.2 *(Euclide) [22] L'ensemble des nombres premiers est infini.*

3.2.3 Plus grand commun diviseur

Le plus grand commun diviseur de deux entiers a et b, noté $pgcd(a, b)$ est le plus grand entier d divisant à la fois a et b. Ainsi a et b sont premiers entre eux si $pgcd(a, b) = 1$.

Proposition 3.2.1 *Si $a \equiv b[n]$, $a \equiv b[m]$ et n et m sont premiers entre eux alors $a \equiv b[nm]$.*

3.2.4 Algorithme d'Euclide

Quel est le plus grand commun diviseur de $a = 255$ et de $b = 141$?.

On calcule donc le pgcd en appliquant l'algorithme d'Euclide. Cet algorithme est décrit dans "Eléments d'Euclide" écrits 300 ans avant J.C.

Étant donné deux entiers naturels a et b, on commence par tester si b est nul. Si oui, alors le pgcd est égal à a. Sinon, on calcule r, le reste de la division de a par b. On remplace a par b, et b par r, et on recommence le procédé. Le dernier reste non nul est le pgcd. Calculons par exemple, le pgcd de 255 et 141 par cet algorithme avec les étapes suivantes :

$$255 = 1 \times 141 + 114$$
$$141 = 1 \times 114 + 27$$
$$114 = 4 \times 27 + 6$$
$$27 = 4 \times 6 + 3$$
$$6 = 2 \times 3 + 0$$

Ainsi, le $pgcd(255, 141) = 3$.

Algorithme Euclide (a,b);

$r_0 := a$;

$r_1 := b$;

$m := 1$;

Tant que $r_m \neq 0$ **faire**

$q_m := \left\lfloor \frac{r_{m-1}}{r_m} \right\rfloor$; (où $\lfloor\rfloor$ désigne la partie entière inférieure)

$r_{m+1} := r_{m-1} - q_m r_m$;

$m := m + 1$;

Fait;

Retourner r_m;

L'algorithme d'Euclide calcule le pgcd de a et b en $O(logM)$ où $M = max(a, b)$.

3.2.5 Preuve d'exactitude de l'algorithme d'Euclide

La preuve de cet algorithme n'est pas difficile. Supposons que a et b soient des nombres entiers et que r soit le reste de la division euclidienne de a par b.
On a $a = qb + r$ où q est le quotient de la division d'où $r = a - qb$.
Puisque $d = pgcd(a, b)$ alors on en déduit que a et b sont des multiples de d.
Comme $r = a - qb$ alors r est la somme de deux multiples de d. On peut donc dire que pour trouver le plus grand commun diviseur de a et b, il suffit de trouver le plus grand commun diviseur de b et r.

Cela justifie que l'on puisse continuer le procédé avec les nombres b et r. Puisque r est le reste de la division entière de a par b alors r est toujours plus petit que b, nous atteindrons $r = 0$ après un nombre fini d'étapes.

L'algorithme d'Euclide permet d'établir aisément le théorème suivant.

Théorème 3.2.3 *(Identité de Bézout) [22] Soit a et b deux entiers tels que $pgcd(a, b) = d$. Il existe alors deux entiers relatifs u et v tels que $au + bv = d$.*

Algorithme d'Euclide étendu

L'algorithme d'Euclide étendu est une version de l'algorithme d'Euclide, à partir de deux entiers a et b, l'algorithme calcule leur plus grand commun diviseur ainsi que deux entiers x et y tels que $ax + by = pgcd(a, b)$. L'algorithme d'Euclide permet d'obtenir de tels entiers parce qu'à chaque étape de l'algorithme, on n'a que des sommes de multiples de a et b.

L'équation $ax + by = d$ est particulièrement utile quand a et b sont premiers entre eux, x est alors l'inverse de a modulo b.

3.2.6 Théorème des restes chinois

Théorème 3.2.4 *(des restes chinois) [22] Soit le système S des congruences suivant :*

$$(S)\begin{cases} x \equiv a_1[m_1] \\ x \equiv a_2[m_2] \\ \quad\vdots \\ x \equiv a_s[m_s] \end{cases}$$

tels que $\forall\ i, j,\ j \neq i,\ pgcd(m_i, m_j) = 1$. *Alors il existe une solution* x_0 *commune à toutes les congruences ci-dessus et toute autre solution est congruente à* x_0 *modulo* M, *où* $M = m_1 m_2 \cdots m_s$.

Théorème 3.2.5 *(Petit théorème de Fermat) Si* n *est un nombre premier, alors pour tout entier* $0 \leq a < n$ *on a :* $a^n \equiv a[n]$.

La preuve de ce théorème nécessite le résultat suivant :

Lemme 3.2.1 *Pour tout* n *premier,* $(a+b)^n = a^n + b^n \bmod n$.

En effet, la formule du binôme affirme que pour tout entier n positif :

$(a+b)^n = \sum_{i=0}^{n} C_n^i a^i b^{n-i} = a^n + b^n + \sum_{i=1}^{n-1} C_n^i a^i b^{n-i}$, où $C_n^i = \dfrac{n!}{i!(n-i)!}$ est le coefficient binomial.

Ce coefficient est un entier, de plus pour $0 < i < n$, aucun des dénominateurs ne contient le nombre n et d'après l'énoncé du théorème, n est un nombre premier. Par conséquent tout coefficient binomial est un multiple de n i.e., $C_n^i \equiv 0[n]$, $0 < i < n$ donc les termes $C_n^i a^i b^{n-i}$ sont également des multiples de n. Cela signifie que $\sum_{i=1}^{n-1} C_n^i a^i b^{n-i} \equiv 0[n]$, d'où le résultat $(a+b)^n = a^n + b^n \bmod n$.
Revenons maintenant à la démonstration du théorème 3.3.6. Nous procédons par récurrence.

Preuve.

- Pour $a = 1$ nous avons effectivement $a^n \equiv a[n]$.
- Supposons maintenant que la relation est vraie pour tout $a \in \{1, ..., k\}$: $a^n \equiv a[n]$.

- Regardons ce qui ce passe pour $a = k + 1$. D'après le lemme précédent $(k + 1)^n = k^n + 1^n \bmod n$. Or, $k^n \equiv k[n]$ par hypothèse, on déduit que $a^n \equiv a[n]$.

■

Définition 3.2.1 *(Fonction indicatrice d'Euler)*

On appelle fonction indicatrice d'Euler, la fonction notée ϕ, qui à tout entier naturel n associe le nombre d'entiers naturels strictement positifs premiers avec n et strictement inférieurs à n.

Exemple 3.2.1 *Pour $n = 15$, on a $\phi(n) = |\{1, 2, 4, 7, 8, 11, 13, 14\}| = 8$.*

Généralisation : Soit un entier p qui s'écrit $p = p_1^{\alpha_1} p_2^{\alpha 2} ... p_k^{\alpha_k}$, où les p_i sont des nombres premiers.

On remarque que pour p premier, $\phi(p) = p - 1$; d'autre part, $\phi(p^\alpha) = p^\alpha - p^{\alpha-1}$, puisque les nombres premiers avec p^α et inférieurs à p^α sont tous les nombres inférieurs à p^α (soit $p^\alpha - 1$ nombres) moins les multiples de p qui sont $p, 2p, 3p, ..., (p^{\alpha-1} - 1)p$ (et il y en a $p^{\alpha-1} - 1$), donc $\phi(p^\alpha) = (p^\alpha - 1) - (p^{\alpha-1} - 1) = p^\alpha - p^{\alpha-1}$.

Lemme 3.2.2 *La fonction ϕ est multiplicative pour les nombres premiers entre eux : si $pgcd(p, q) = 1$ alors $\phi(pq) = \phi(p)\phi(q)$.*

3.2.7 Théorème d'Euler

Théorème 3.2.6 *Soient a et n deux nombres premiers entre eux avec $a < n$. Alors $a^{\phi(n)} \equiv 1[n]$.*

Preuve. Supposons que $n = p^\alpha$, où p est premier. Raisonnons par récurrence sur α. Pour $\alpha = 1$ la relation est vérifiée et ce d'après le petit théorème Fermat.

Supposons que la relation est vraie pour $\alpha - 1$ c'est-à-dire, $a^{p^{\alpha-1} - p^{\alpha-2}} = 1 + p^{\alpha-1} b$, où b est un entier. Elevons les membres de cette équation à la puissance p, on obtient, $a^{p^\alpha - p^{\alpha-1}}$ est égal à 1 plus une somme dont chaque terme est divisible par p^α. Par conséquent $a^{\phi(p^\alpha)} \equiv 1[p^\alpha]$.

Soit, maintenant $n = p_1^{\alpha_1} p_2^{\alpha 2} ... p_k^{\alpha_k}$, on a d'après l'hypothèse de récurrence $a^{\phi(p_i^{\alpha_i})} \equiv 1[p_i^{\alpha_i}]$, $i = 1, ..., k$. Comme les $p_i^{\alpha_i}$ sont premiers entre eux, alors d'après la proposition 3.3.1 des congruences et par multiplicativité de ϕ on a $a^{\phi(n)} \equiv 1[n]$. ■

Le théorème des restes chinois et le théorème d'Euler sont respectivement utilisés dans les chiffres de Rabin et RSA, qu'on citera au chapitre consacré pour la cryptographie à clé publique, tandis que le petit théorème de Fermat est utilisé dans les testes de primalité probabilistes.

Chapitre 4

Chiffrement à clé secrète

Sommaire

4.1 Introduction

Le chiffrement symétrique repose sur l'existence d'une clé secrète, commune à l'émetteur et au récepteur du message, qui doit rester secrète. Parmi les systèmes symétriques on cite le DES, le triple DES et le AES.

4.2 Le DES : Data Encryption Standard

4.2.1 Introduction

Le 15 mai 1973 le NBS (National Bureau of Standards des Etats Unis, aujourd'hui appelé NIST - National Institute of Standards and Technology) a lancé un appel dans le Federal Register pour la création d'un algorithme de chiffrement répondant aux critères suivants :

- l'algorithme doit reposer sur une clé relativement petite, qui sert à la fois au chiffrement et au déchiffrement,
- l'algorithme doit être facile à implémenter, logiciellement et mâtériellement, et doit être très rapide,
- le chiffrement doit avoir un haut niveau de sécurité, uniquement lié à la clé, et non à la confidentialité de l'algorithme.

Les efforts conjoints d'IBM, qui propose l'algorithme Lucifer fin 1974, et de la NSA (National Security Agency) conduisent à l'élaboration du DES (Data Encryption Standard) et ce en 1976.

4.2.2 Schéma de Feistel

1. Soit M un message à $2n$ bits (succession de 0 et de 1) clairs ou chiffrés. On décompose M en deux blocs consécutifs de n bits chacun, le premier est noté L (Left) le deuxième R (Right), ainsi $M = (L, R)$.

2. La clé K est un ensemble de vecteurs binaires $K = \{K_1, ..., K_d\}$ chacun de longueur k.

3. On dispose d'une fonction f définie de $\{0,1\}^{n+k}$ dans $\{0,1\}^n$, dont le choix est important car elle doit être non linéaire et elle doit bien "brouiller le message". Il n'y a pas en principe d'autre condition à respecter pour le choix de la fonction f.

L'image du bloc (L_i, R_i) par le schéma de Feistel est le bloc (L_{i+1}, R_{i+1}), avec $L_{i+1} = R_i$, et $R_{i+1} = L_i \; xor \; f(R_i, K_i)$ (xor est l'opérateur logique "ou exclusif" qu'on notera par la suite $\oplus$). Cette transformation est répétée un certain nombre de fois (16 dans le cas du DES) et est appelée ronde.

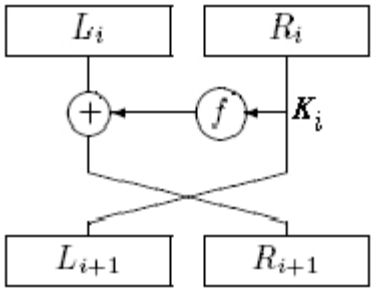


Figure 4.1 : Shéma de Feistel

Cela signifie qu'après chaque ronde, la partie droite est placée à gauche, tandis que la partie gauche est additionnée modulo 2 avec le résultat de la fonction f appliquée à la partie droite, le résultat étant placé à droite.

La table d'addition modulo 2 est : $1 \oplus 1 = 0$, $0 \oplus 0 = 0$, $1 \oplus 0 = 1$ et $0 \oplus 1 = 1$.

Propriétés importantes :

1. $b \oplus b = 0$

2. $a \oplus b \oplus b = a$

3. $a \oplus b = c \Rightarrow c \oplus b = a$

Algorithme Feistel;

Chiffrement

Entrées : Un message clair $M = (L_0, R_0)$ et les clés K_i

Pour i=1 **haut** d **faire**

$L_i := R_{i-1}$

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

Sortie : Un message chiffré $C = (L_d, R_d)$

Déchiffrement

Entrées : Un message chiffré $C = (L_d, R_d)$ et les clés de chiffrement

Pour i=d **bas** 1 **faire**

$R_{i-1} := L_i$

$L_{i-1} := R_i \oplus f(L_i, K_i)$

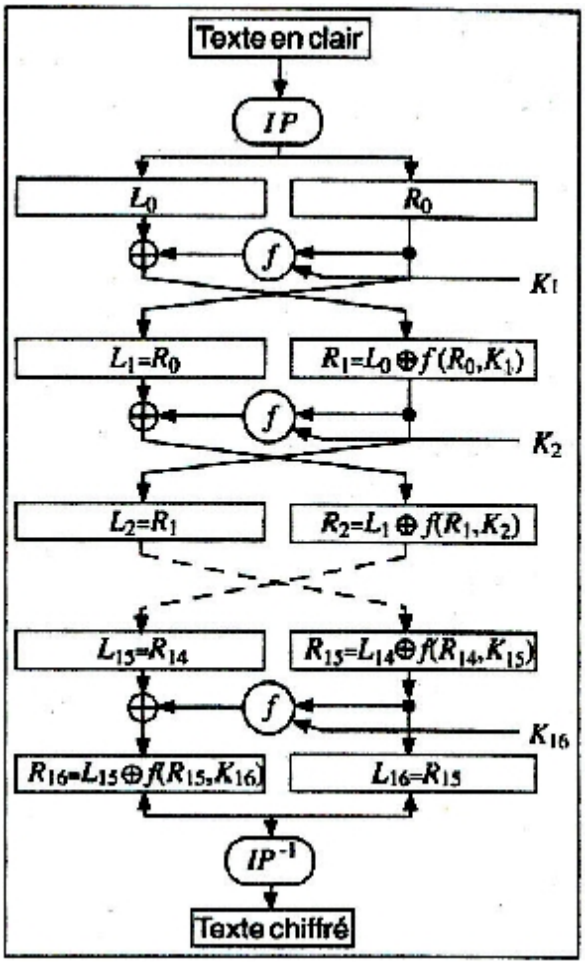
Sortie : Le message clair $M = (L_0, R_0)$

4.3 Description du DES

4.3.1 Schéma général

C'est un système de chiffrement par blocs de 64 bits, qui fonctionne selon le schéma précédent de Feistel tels que : $2n = 64$, $d = 16$ et $k = 48$.

L'algorithme consiste à faire des combinaisons, des substitutions et des permutations entre le texte à chiffrer et la clé, en faisant en sorte que les opérations puissent se faire dans les deux sens (chiffrement et déchiffrement). On parlera de rondes. Le DES a 16 rondes, c'est-à-dire qu'il applique 16 fois la même combinaison de techniques au bloc de texte en clair comme le montre le schéma suivant :



Pour cet algorithme la fonction f est la composée de plusieurs fonctions. L'idée est la suivante, tout d'abord une permutation initiale notée IP, est appliquée, ensuite le message obtenu est scindé en deux blocs de 32 bits chacun un bloc droite R_0 et un bloc gauche L_0. Ainsi le déroulement de la première ronde est commencé, la partie droite R_0 est affectée à la partie gauche $L_1(L_1 := R_0)$, R_0 subit ensuite quelques transformations, premièrement une permutation dite expansive (notée E), dans laquelle les 32 bits sont mélangés et 16 d'entre eux sont dupliqués. La sortie de E à 48 bits est additionné modulo deux avec la clé de ronde K_1, le résultat de cet addition est scindé en 8 blocs de 6 bits, chacun passe par une fonction de sélection (appelée boîte ou table de substitution) notées Si, $i = 1, \ldots, 8$ est ainsi substitué en un bloc de 4 bits, ces bits sont regroupés pour former un bloc de 32 bits. Ce dernier obtenu est enfin soumis à une permutation P et est additionné modulo deux avec L_0 pour former R_1.

Détaillons maintenant le processus.

4.3.2 Fractionnement du texte

Le texte clair est fractionné en blocs de 64 bits.

4.3.3 Permutation initiale

Dans un premier temps, chaque bit d'un bloc est soumis à la permutation initiale, pouvant être représentée par le tableau de permutation initiale notée IP suivant :

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

Ce tableau, ainsi que les autres de ce chapitre, doivent être lu de gauche à droite et de haut en bas. Par exemple la permutation initiale remplace le bit 58 par le premier bit, le $50^{ème}$ par le deuxième et ainsi de suite.

4.3.4 Scindement en blocs de 32 bits

Une fois la permutation initiale réalisée, le bloc de 64 bits est scindé en deux blocs de 32 bits, notés respectivement L_0 et R_0.

L_0 :	58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
	62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
R_0 :	57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
	61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

4.3.5 Ronde

Les blocs L_0 et R_0 sont soumis à un ensemble de transformations itératives appelées ronde, explicitées dans ce schéma, et dont les détails sont donnés plus bas :

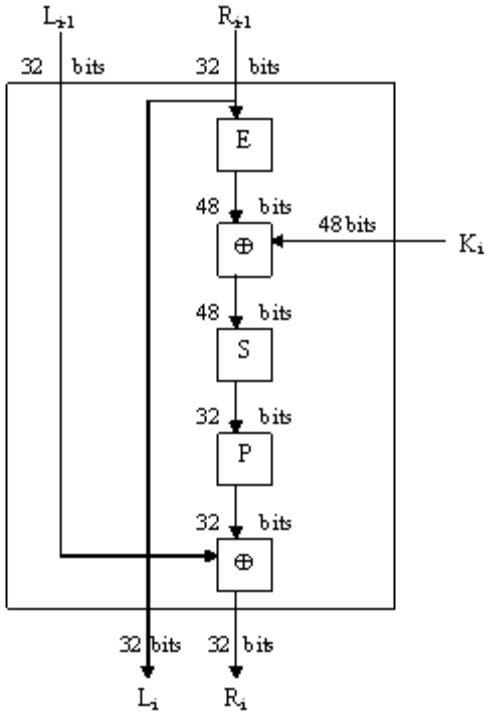


Figure 4.2 : Une ronde du DES

Fonction d'expansion

Les 32 bits du bloc R_0 sont étendus à 48 bits grâce à une table appelé table d'expansion notée E, dans laquelle les 32 bits sont mélangés et 16 d'entre eux sont dupliqués :

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

Ainsi, le dernier bit de R_0 devient le premier, le premier devient le second, ... etc.
De plus, les bits 1,4,5,8,9,12,13,16,17,20,21,24,25,28,29 et 32 de R_0 sont dupliqués.

Cette opération a deux buts : le résultat a la même taille que la clé pour l'opération "ou exclusif", et elle fournit un résultat plus long qui pourra être comprimé pendant l'opération de substitution.

Bien que le bloc de sortie soit plus long que le bloc d'entrée, il n'existe qu'un bloc d'entrée possible pour un bloc de sortie donné.

Ou exclusif avec la clé

La matrice résultante de l'expansion à 48 bits est notée $E[R_0]$. L'algorithme DES procède ensuite à un "ou exclusif" entre la première clé K_1 et $E[R_0]$. Le résultat est une matrice à 48 bits que nous appelerons par commodité R_0.

Fonction de substitution

Le résultat de 48 bits est soumis à une opération de substitution. Cette substitution est réalisée à l'aide de huit tables de substitution, ce sont les tables S.

R_0 est ensuite divisé en 8 blocs de 6 bits, chaque bloc est manipulé séparément par une table S différente selon le schéma suivant :

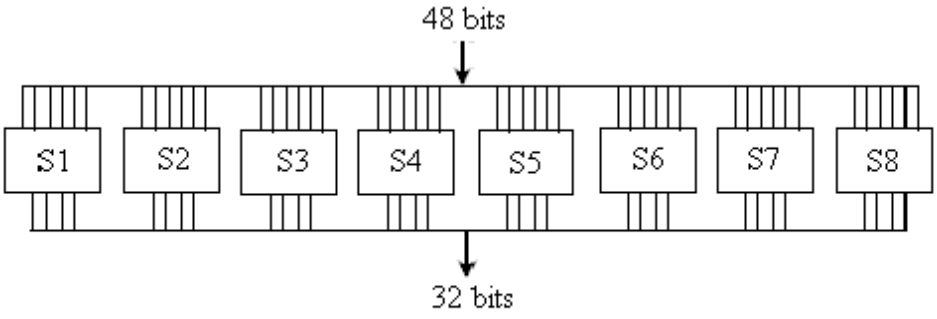


Figure 4.3 : Les tables S

Chaque table S a 4 lignes et 16 colonnes dont la cellule est un nombre de 4 bits (de 0 à 15).

Les premiers et derniers bits de chaque bloc déterminent la ligne de la table S correspondante, les autres bits (respectivement 2, 3, 4 et 5) déterminent la colonne à exploiter pour obtenir la sortie. La sélection de la ligne se faisant sur deux bits, il y a 4 possibilités (0,1,2,3). La sélection de la colonne se faisant sur 4 bits, il y a 16 possibilités (0 à 15).

Supposons que les 6 bits à l'entrée de la table S1 soient 110101.
Le premier et le dernier bit, soient 11 indiquent la ligne 3, les quatre bits centraux 1010 indiquent la colonne 10. A l'intersection de la $3^{ème}$ ligne et de la $10^{ème}$ colonne de S1, on

trouve 3 qui, écrit en binaire donne 0011, qui sont les 4 bits de sortie.

Chaque bloc de 6 bits est ainsi substitué en un bloc de 4 bits. Ces bits sont regroupés pour former un bloc de 32 bits.

Voici les tables de substitution :

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

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

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

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

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

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

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

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

Conception des tables S

Pas mal d'efforts ont été investi dans l'étude de la conception des tables S. Biham et Shamir montrèrent que la conception des tables S, ainsi que leur ordre lui même étaient optimisés pour résister à la cryptanalyse différentielle (dont les concepteurs étaient au courant, et ils été restés silencieux parcequ'ils voulaient éviter sa découverte et son utilisation).

La permutation P

Les 32 bits de sortie des substitutions par les tables S sont permutés à l'aide de la table P suivante :

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

Ou exclusif

Finalement le résultat de la permutation P est combiné par un ou exclusif avec la moitié gauche L_0 pour donner R_1, tandis que le R_0 initial donne L_1.

4.3.6 Itération

L'ensemble des étapes précédentes, qui constituent une ronde est réitéré 16 fois.

4.3.7 Permutation initiale inverse

A la fin des 16 itérations, les deux blocs L_{16} et R_{16} sont concaténés pour reformer un seul bloc de 64 bits, puis subit la permutation IP^{-1}(qui est l'inverse de la permutation initiale IP) :

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

Algorithme DES;

Chiffrement

Entrées : Un message clair M, la taille des blocs est de 64, le nombre de rondes $d = 16$, les clés $(K_1, ..., K_d)$ déduites de K_0

$M := IP(M)$ (permutation initiale de texte en clair)

Pour i:=1 **haut** 16 **faire**

$L_i := R_{i-1}$

$R_i := f(R_{i-1}, K_i) \oplus L_{i-1} = P\left(S\left(E\left(R_{i-1}\right) \oplus K_i\right)\right) \oplus L_{i-1}$

Avec : $E : \{0,1\}^{32} \rightarrow \{0,1\}^{48}$ une fonction linéaire appelée Expansion

$S : \{0,1\}^{48} \rightarrow \{0,1\}^{32}$ une transformation non linéaire qui utilise les tables de substitution S et une permutaion P

$C := IP^{-1}(L_{16}R_{16})$

Sortie : un message chiffré $C = (L_{16}, R_{16})$

Déchiffrement

Entrées : Le message chiffré $C = (L_{16}, R_{16})$ et les clés de chiffrement $(K_1, ..., K_d)$

$C := IP(C)$

Pour i:=16 **bas** 1 **faire**

$R_{i-1} := L_i$

$L_{i-1} := f(L_i, K_i) \oplus R_i$

$M := IP^{-1}(L_0R_0)$

Sortie : Le message clair M

Remarque 4.3.1 *Pour l'algorithme DES que nous avons programmé les clés de rondes K_i sont générées aléatoirement de clé initiale K, tandis qu'il existe dans la littérature une autre méthode pour leurs génération, les détails sont disponibles dans* [4] *et* [6].

Exemple 4.3.1 *Soit à chiffrer le message*

M=1111111110000000000111111111100000000001111111110000000000111,

avec une clé ayant tous ces bits égaux à 1. Pour cet exemple on va exécuter uniquement deux rondes du DES.

- **Permutation initiale (IP) :**

  IP(M)=1111101100101001101011011010010101 1101011100101001101011011010010101.

  $L_0 = 11111011001010011010110110100101$, $R_0 = 01101011001010011010110110100101$.

- **Première ronde :**

  $L_1 := R_0 = 01101011001010011010110110100101$.

  $E(R_0) = 10110101011010010101001111010101101111010000101 0$.

  $E(R_0) \oplus K_1 =$ 010010 101001 011010 101100 001010 100100 001011 110101.

  $S(E(R_0) \oplus K_1) = 1010\ 0011\ 0100\ 0111\ 1010\ 1111\ 1001\ 1001$.

  $P(S(E(R_0) \oplus K_1)) = 11011011111000010111101010001001$.

  $P(S(E(R_0) \oplus K_1)) \oplus L_0 = 11111011001010011010110110100101$.

  $R_1 = 11111011001010011010110110100101$.

- **Deuxième ronde :**

  $L_2 := R_1 = 11111011001010011010110110100101$.

  $E(R_1) = 00010000000101100101000101101010111010010101100 0$.

  $E(R_1) \oplus K_2 =$ 111011111110100110101110100101010001011010100111.

  $S(E(R_1) \oplus K_2) = 00001111100111011100011010100111$.

  $P(S(E(R_1) \oplus K_2)) = 11000101001011100101110101111001$.

  $P(S(E(R_1) \oplus K_2)) \oplus L_1 = 01101011001010011010110110100101$.

  $R_2 = 01101011001010011010110110100101$.

- **Permutation initiale inverse ($\mathbf{IP^{-1}}$) :**

$$\begin{aligned} C &= IP^{-1}(L_2R_2) \\ &= 00100100101001001010011110010011000011101100100100011110100111 10. \end{aligned}$$

  Le déchiffrement utilise l'algorithme ci-dessus, on part du texte chiffré et on utilise les clés $K_{16}...K_1$ dans cet ordre. Déchiffrons le message C.

- **Permutation initiale :**

  IP(C)=00100000110010001101011100101100 10101110000001111111000011011100.

  $L_2 = 00100000110010001101011100101100$, $R_2 = 10101110000001111111000011011100$.

- **Première ronde :**

  $R_1 := L_2 = 00100000110010001101011100101100$.

  $E(L_2) = 000100000001011001010001011010101110100101011000$.

  $E(L_2) \oplus K_2 = 111011\ 111110\ 100110\ 101110\ 100101\ 010001\ 011010\ 100111$.

  $S(E(L_2) \oplus K_2) = 0000\ 1111\ 1001\ 1101\ 1100\ 0110\ 1010\ 0111$.

  $P(S(E(L_2) \oplus K_2)) = 11000101001011100101110101111001$.

  $P(S(E(L_2) \oplus K_2)) \oplus R_2 = 01101011001010011010110110100101$.

  $L_1 = 01101011001010011010110110100101$.

- **Deuxième ronde :**

  $R_0 := L_1 = 01101011001010011010110110100101$.

  $E(L_1) = 10110101011010010101001111010101101111010000 1010$.

  $E(L_1) \oplus K_1 = 010010\ 101001\ 011010\ 101100\ 001010\ 100100\ 001011\ 110101$.

  $S(E(L_1) \oplus K_1) = 1010\ 0011\ 0100\ 0111\ 1010\ 1111\ 1001\ 1001$.

  $P(S(E(L_1) \oplus K_1)) = 11011011111000010111101010001001$.

  $P(S(E(L_1) \oplus K_1)) \oplus R_1 = 11111011001010011010110110100101$.

  $L_0 = 11111011001010011010110110100101$.

- **Permutation initiale inverse ($\mathbf{IP^{-1}}$) :**

$$\begin{aligned} M &= IP^{-1}(L_0R_0) \\ &= 11111111110000000000111111111100000000001111111111000000000001111. \end{aligned}$$

4.4 Cryptanalyse du DES

On éstimait à l'époque de sa création que cet algorithme serait sûr pendant de nombreuses décennies, c'était sans compter la rapidité de l'évolution des technologies de l'informaique et l'extraordinaire progression de la puissance des ordinateurs. Le 17 juin 1997, le DES est cassé en 3 semaines par une fédération de petites machines sur Internet après avoir exploré environ un quart de l'espace des clés. Bien qu'une telle recherche demande des moyens considérables, elle a montré que le DES n'offre plus aujourd'hui une grande sécurité.

Il existe d'autres types d'attaques contre le DES, qui consistent à exploiter certaines structures particulières de l'algorithme ou certains biais statistiques dans la distribution des couples claire-chiffré. Les plus connues sont la cryptanalyse différentielle, inventée par les Israeliens Biham et Shamir en 1991, et la cyptanalyse linéaire proposée par le Japonais Matsui en 1993. Le lecteur intéressé peut voir [13].

4.5 Le triple DES

Après la mort du DES la solution a été dans un premier temps l'adoption du triple DES noté TDES, trois applications du DES à la suite avec 2 clés différentes (avec une première clé K_1, puis une deuxième K_2, puis à nouveau la première).

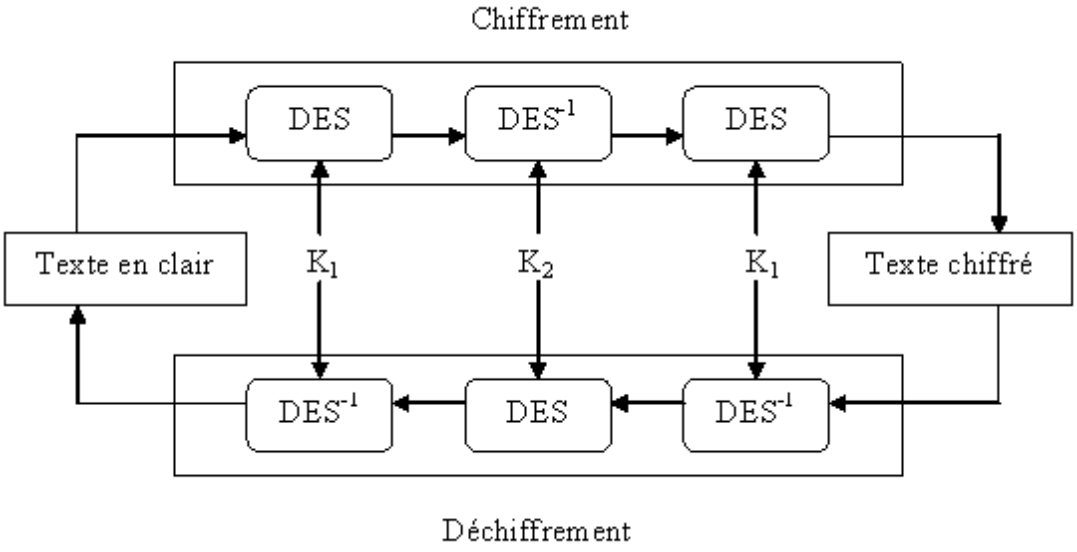
Soit à chiffrer le message M, le message chiffré C est donné par :

$$C = DES(K_1) \circ DES^{-1}(K_2) \circ DES(K_1)(M)$$

ainsi :

$$DES^{-1}(K_1) \circ DES(K_2) \circ DES^{-1}(K_1)(C) = M$$

Ce qui est montré par le schéma suivant :



Le TDES permet d'augmenter significativement la sécurité du DES, toutefois il a l'inconvénient majeur de demander également plus de ressources pour le chiffrement et le déchiffrement.

Dès 1997, le NIST (National Institute of Standards and Technology) lance un appel international pour trouver un standard de chiffrement successeur du DES. Le vainqueur est l'algorithme suivant.

4.6 L'algorithme AES

4.6.1 Introduction

L'AES (Advanced Encryption Standard) est, comme son nom l'indique, un standard de cryptage symétrique destiné à remplacer le viellissant DES dont la taille des clés devenait trop petite.

Après l'appel lancé par le NIST en deux phases de 1998 à 1999, une quinzaine de candidats se sont mis sur les rangs. Cinq ont été retenu au cours de l'été 1999, et finalement c'est l'algorithme Rijndael proposé par les deux concepteurs belges Rijmen et Daemen, qui a été retenu en octobre 2000 pour devenir l'AES après une période de normalisation.

4.6.2 Description de l'AES

Cet algorithme suit les spécifications suivantes :

- l'AES est un standard, donc libre d'utilisation,
- c'est un algorithme de type symétrique,
- c'est un algorithme de chiffrement par blocs,
- Rijndael considère une taille de bloc et une taille de clé pouvant être fixées à un multiple quelconque de 32 bits, avec un minimum de 128 bits et un maximum de 256 bits, la taille du bloc peut différer de celle de la clé. Par contre l' AES fixe la taille du bloc à 128 bits représentée par $N_b = 4$, où N_b reflète le nombre de mots de 32 bits dans un bloc. La clé pouvant faire 128, 192 ou 256 bits seulement, la longueur de la clé est caractérisée de façon similaire par $N_k = 4$, 6 ou 8.

L'AES exécute une séquence de rondes qui seront détaillées dans la suite. On note N_r le nombre de rondes qui doivent être effectuées. Ce nombre dépend des valeurs de N_b et de N_k. Les différentes configurations possibles sont données dans le tableau suivant :

	N_k	N_b	N_r
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Définition 4.6.1 *(Construction de $GF(2^8)$) Le corps de base est $\mathbb{Z}/2\mathbb{Z}$ (noté aussi GF(2), contenant deux éléments 0 et 1). Un polynôme à coefficients dans $\mathbb{Z}/2\mathbb{Z}$, par exemple : $x^6 + x + 1$ peut être noté à l'aide d'une suite de 0 et de 1 :* 1000011. *Et inversement une suite de 0 et de 1 peut être interprétée comme un polynôme sur $\mathbb{Z}/2\mathbb{Z}$. Si on veut travailler avec des octets le polynôme de l'exemple sera noté :* 01000011. *Un élément de $GF(2^8)$ est un polynôme de degré au plus 7.*

4.6.3 Arithmétique dans $GF\left(2^8\right)$

L'objet de base pour le AES est un élément du corps $GF(2^8)$. Ce corps contient 2^8 éléments. On représente un élément de ce corps par un polynôme de degré au plus 7. Un tel polynôme s'écrit :

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0$$

avec $b_i \in \{0, 1\}\, \forall i$, et on peut stocker ce polynôme en mémoire par l'octet $(b_7b_6b_5b_4b_3b_2b_1b_0)$ que l'on peut aussi écrire en hexadécimal.

Par exemple la valeur hexadécimale 63 correspond au polynôme $x^6 + x^5 + x + 1$ et à l'octet (01100011).

Intéressons nous maintenant aux opérations dans $GF\left(2^8\right)$.

L'addition

La somme de deux éléments de $GF\left(2^8\right)$ est la somme des polynômes correspondants modulo 2.

Par exemple :

$(x^6 + x^4 + x^3 + 1) + (x^7 + x^5 + x^3 + 1) = (x^7 + x^6 + x^5 + x^4)$ (Notation polynomiale)

$(01011001) \oplus (10101001) = (11110000)$ en binaire, ou encore en hexadécimal $59 + a9 = f0$.

La multiplication

Pour multiplier deux éléments de $GF\left(2^8\right)$ on les écrits sous forme polynomiale, on les multiplie, puis on cherche le reste du résultat dans la division par un polynôme irréductible $m(x)$ fixé au préalable de degré 8 sur $GF(2)$. Pour AES, ce polynôme est $m(x) = x^8 + x^4 + x^3 + x + 1$. Le résultat sera à nouveau un polynôme de degré inférieur ou égal à 7.

Par exemple, le produit de 59 par a1 donne :

$$\begin{aligned}(x^6 + x^4 + x^3 + 1) * (x^7 + x^5 + 1) &= x^{13} + x^{10} + x^9 + x^8 + x^7 + x^6 + x^5 + x^3 + 1 \\ &= \left(x^5 + x^2\right) m(x) + x^7 + x^6 + x^5 + x^2 + 1\end{aligned}$$

Inverse d'un élément non nul

Tout élément non nul $p(x)$ de $GF\,(2^8)$ admet un inverse. Celui-ci peut se calculer via l'algorithme d'Euclide étendu, qui détermine deux polynômes $u(x)$ et $v(x)$ tels que :

$$u(x)p(x) + v(x)m(x) = 1$$

L'élément $u(x)$ est appelé inverse de $p(x)$.

Soit par exemple, à chercher l'inverse de $x^6 + x^4 + x^2 + x + 1$ dans $GF\,(2^8)$.

Les divisions successives de l'algorithme fournissent :

(1) $x^8 + x^4 + x^3 + x + 1 = (x^6 + x^4 + x^2 + x + 1)(x^2 + 1) + x^4$

(2) $x^6 + x^4 + x^2 + x + 1 = x^4(x^2 + 1) + (x^2 + x + 1)$

(3) $x^4 = (x^2 + x + 1)(x^2 + x) + x$

(4) $x^2 + x + 1 = x(x + 1) + 1$

L'équation (4) n'est autre que la relation de Bézout entre les polynômes entre $x^2 + x + 1$ et x. Elle s'écrit : $(x^2 + x + 1) + x(x + 1) = 1$.

En remplaçant x par l'expression (3) : $(x^2+x+1)+[x^4 + (x^2 + x + 1)(x^2 + x)]\,(x+1) = 1$.

Après simplification :$(x^2 + x + 1)(x^3 + x + 1) + x^4(x + 1) = 1$.

En remplaçant $x^2 + x + 1$ à l'aide de l'équation (2) :

$$\left[(x^6 + x^4 + x^2 + x + 1) + x^4(x^2 + 1)\right](x^3 + x + 1) + x^4(x + 1) = 1$$

Après simplification : $(x^6 + x^4 + x^2 + x + 1)(x^3 + x + 1) + x^4(x^5 + x^2) = 1$.

En remplaçant x^4 par l'expression (1) et après simplification on obtient :

$$(x^8 + x^4 + x^3 + x + 1)(x^5 + x^2) + (x^6 + x^4 + x^2 + x + 1)(x^7 + x^5 + x^4 + x^3 + x^2 + x + 1) = 1$$

Ceci est la relation de Bézout entre les polynômes $(x^8 + x^4 + x^3 + x + 1)$ et $(x^6 + x^4 + x^2 + x + 1)$.

On en déduit que l'inverse de $(x^6 + x^4 + x^2 + x + 1)$ est $(x^7 + x^5 + x^4 + x^3 + x^2 + x + 1)$ dans $GF\,(2^8)$.

4.6.4 Implémentation d'AES pour une clé de 128 bits

Un état de l'AES est un bloc de 128 bits, soit 16 octets, ces 16 octets sont ensuite ordonnés sous forme d'un tableau 4×4 :

a_0	a_4	a_8	a_{12}
a_1	a_5	a_9	a_{13}
a_2	a_6	a_{10}	a_{14}
a_3	a_7	a_{11}	a_{15}

que l'on notera par : $a_0a_1a_2a_3a_4a_5a_6a_7a_8a_9a_{10}a_{11}a_{12}a_{13}a_{14}a_{15}$ où les a_i sont des octets écrits en héxadécimal. Ce tableau étant lui même lu par colonne de quatre **words** (un word est un mot de 32 bits, c'est-à-dire 4 octets) :

w_1	w_2	w_3	w_4

4.6.5 Le chiffrement AES

Le chiffrement s'éffectue de la manière suivante :

- La clé est étendue à 11 ($N_r + 1$) clés différentes.
- La clé initiale est additionnée à l'état courant.
- Ensuite 9 rondes sont éffectuées, chacune est constitué de quatre étapes :
  a. ByteSub
  b. ShiftRow
  c. MixColumn
  d. AddRoundKey
- Enfin, une ronde finale FinalRound est appliquée (elle correspond à une ronde dans laquelle l'étape MixColumn est omise).

  La clé de la i^{ème} ronde sera notée RoundKey[i], et RoundKey[0] référencera la clé initiale. La dérivation de la clé de chiffrement K dans le tableau RoundKey est notée KeyExpansion.

Le chiffrement AES peut être décrit de façon formelle de la manière suivante :

Algorithme AES_Chiffrement;

Entrées : Un message binaire M et une clé K à 128 bits

AddRoundKey(etat, RoundKey[0]) (* Addition initiale *)

(* Les $N_r - 1$ rondes*)

Pour $r := 1$ **haut** $N_r - 1$ **faire**

ByteSub(etat)

ShiftRow(etat)

MixColumn(etat)

AddRoundKey(etat, RoundKey[r])

(* Ronde finale *)

ByteSub(etat)

ShiftRow(etat)

AddRoundKey(etat, RoundKey[N_r])

Sortie : Un message chiffré C

Détaillons maintenant chaque étape.

ByteSub

Chaque case du tableau (correspondant au message) contient un élément a_i de $GF(2^8)$, ce dernier est inversé dans $GF(2^8)$, le 0 est inchangé.

Le résultat b qui est un octet subit la transformation définie par le cacul matriciel :

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Notons que ces deux opérations sont inversibles.

ShiftRow

Chaque ligne du tableau état (obtenu après ByteSub) subit une permutation circulaire vers la gauche, de 0 cran pour la ligne 1, 1 cran pour la ligne 2, 2 crans pour la ligne 3 et 3 crans pour la dernière ligne. D'où le schéma suivant :

S					$S\prime$			
$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$		$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$	$\longrightarrow$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$	$S_{1,0}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$		$S_{2,2}$	$S_{2,3}$	$S_{2,0}$	$S_{2,1}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$		$S_{3,3}$	$S_{3,0}$	$S_{3,1}$	$S_{3,2}$

La transformation inverse consiste à éffectuer les mêmes permutations, mais vers la droite.

MixColumn

Cette opération quand à elle, agit sur les colonnes. Elle multiplie chaque colonne, considérée comme un polynôme de degré 3 sur $GF\left(2^8\right)$ par le polynôme $P(y) = 3y^3 + y^2 + y + 2$ et réduit modulo le polynôme $y^4 + 1$.

Le polynôme $P(y)$ a été choisi afin de permettre le déchiffrement, son inverse modulo $y^4 + 1$ est $P(y)^{-1} = by^3 + dy^2 + 9y + e$.

AddRoundKey

Le tableau obtenu est remis en ligne de 16 octets, ou plutôt sous forme d'une suite de 128 bits. Cette suite est additionnée bit à bit modulo 2 avec la clé de ronde.

Cette addition étant son propre inverse, elle ne posera aucun problème au niveau du déchiffrement.

La génération des clés de ronde

Pour AES-128, cette opération transforme une clé de 4 mots en 11 clés de 4 mots (soit 10 supplémentaires, la première étant celle de départ) dites clés de rondes RoundKey. Ceci est réalisé par un processus de récurrence sur les mots :
Pour commencer on a la clé initiale de 4 mots $w_0 w_1 w_2 w_3$, w_i s'obtient en additionnant bit à bit w_{i-1} et w_{i-4} :

$$w_i = w_{i-1} \oplus w_{i-4}$$

Cependant, avant d'éffectuer cette opération dans le calcul de $w_4, w_5, ..., w_{43}$, w_{i-1} aura le droit à un traitement en faveur si i est un multiple de 4.

Ecrivons $w_{i-1} = [c_0, c_1, c_2, c_3]$ où c_k est un octet.

a) Permuter circulairement w_{i-1} pour obtenir $[c_1, c_2, c_{3,} c_0]$.

b) Appliquer ByteSub à chacun octet c_k, on obtient $\left[c_1^{'}, c_2^{'}, c_{3,}^{'} c_0^{'}\right]$.

c) Additionner bit à bit $c_1^{'}$ avec le polynôme $x^{i/4-1} \bmod [x^8 + x^4 + x^3 + x + 1]$.

4.6.6 Le déchiffrement AES

Pour déchiffrer le message reçu, le générateur de clés de rondes fonctionne exactement de la même façon.

Dans la description des rondes, on a vu que chaque opération est inversible. Il suffira donc de reprendre l'algorithme dans l'ordre inverse avec les fonctions inverses.

Algorithme AES_Dechiffrement;

Entrées : Un message chiffré C et les clés K_i de chiffrement

AddRoundKey(etat, RoundKey[N_r]) (* Addition initiale *)

(* Les $N_r - 1$ rondes*)

Pour $r := N_r - 1$ **bas** 1 **faire**

InvShiftRow(etat)

InvByteSub(etat

AddRoundKey(etat, RoundKey[r])

InvMixColumn(etat)

(* Ronde finale *)

InvShiftRow(out)

InvByteSub(out)

AddRoundKey(out, RoundKey[0])

Sortie : Le message clair M

4.6.7 Caractéristiques et points forts de l'AES

Le fait que l'AES soit uniquement composé d'opérations simples sur les octets le rend extrêmement rapide, il atteint un débit de chiffrement de 70 Mbits/s pour une implémentation en C++ sur un Pentium à 200 MHz de plus ses besoins en ressources et mémoire sont très faibles. En conclusion, l'AES est plus sûr que le DES car il présente une résistance aux attaques connues pour ce dernier.

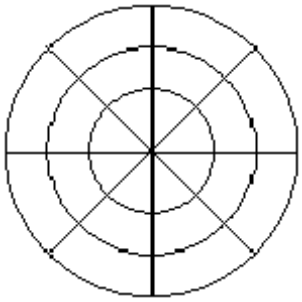
Certains groupes ont affirmé avoir cassé le AES complet mais après vérification par la communauté scientifique, il s'avérait que toutes ces méthodes étaient erronées. Donc en l'absence d'attaques efficaces pour l'AES, il reste considéré comme sûr.

4.7 Proposition d'un cryptosystème symétrique basé sur le problème de factorisation

Nous proposons dans cette section un cryptosystème basé sur la difficulté de résolution du problème de la factorisation d'un grand nombre composé (voir complexité des problèmes au chapitre suivant) nommé **le disque tournant**.

Soit D un disque partitionné en pistes et secteurs selon deux paramètres p et q, où p désigne le nombre de pistes et q le nombre de secteurs. Le disque D ainsi formé sera partitionné en $p \times q$ clusters.

Exemple 4.7.1 *Un disque à 3 pistes et 8 secteurs (24 clusters) :*

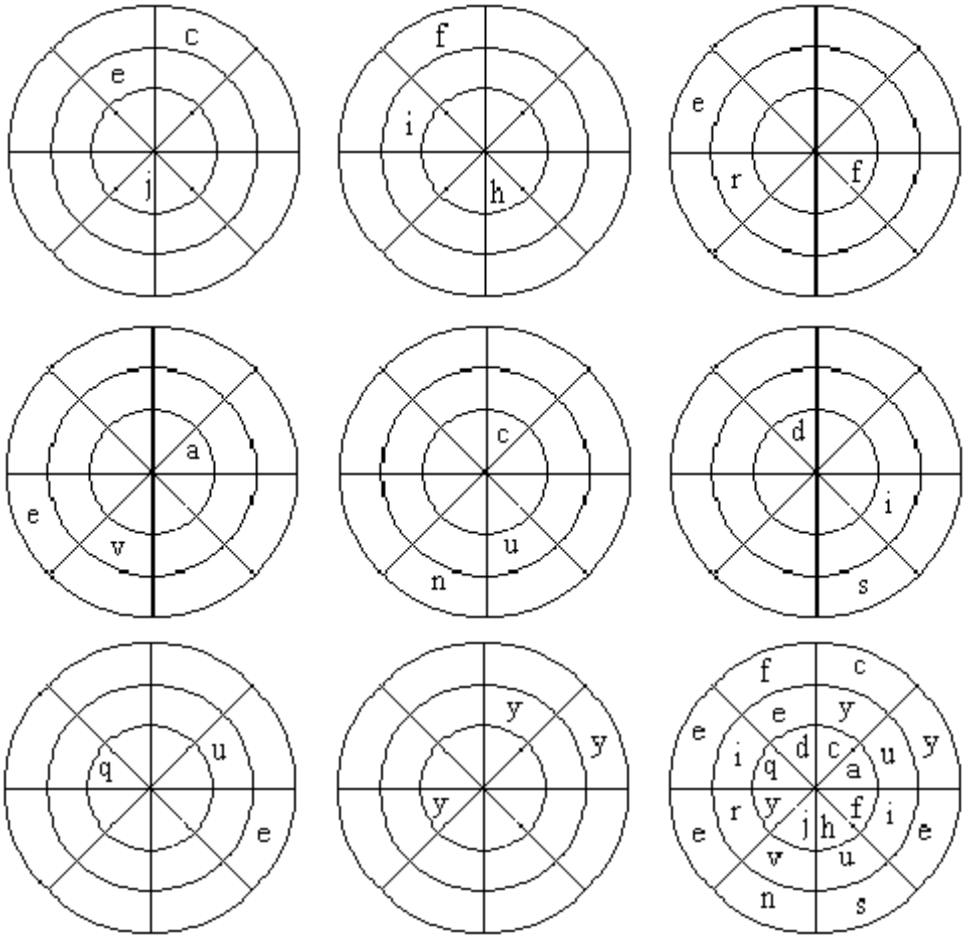


Nous décrivons maintenant la méthode du chiffrement avec le disque D.

4.7.1 Description de la méthode de chiffrement

De chaque piste on choisit au hasard un cluster à trouer, le disque troué obtenu est appelé cache et représente la clé. Les p premières lettres du message sont écrites dans les trous, en commençant par la première piste puis la deuxième et ainsi de suite jusqu'à épuisement des pistes. A chaque fois qu'un emplacement d'une p-chaîne de caractères est terminé, on fait tourner le disque autour de son centre d'un angle $\alpha = \dfrac{2\pi}{q}$, dans le sens trigonométrique, ce qui permet un emplacement d'une nouvelle p-chaîne de caractères du message clair. Une fois tous les clusters du disque sont remplis, on écrit les caractères de chaque secteur de l'intérieur vers l'extérieur, dans le sens des aiguilles d'une montre à partir de l'angle 0.

Exemple 4.7.2 *Chiffrons le message "je chiffre avec un disque yyy" avec un disque tournant ayant 3 pistes avec un angle de rotation* $\alpha = \frac{\pi}{4}$.



Le message chiffré est alors : fiehu sjvny reqie defcy cauy.

Le déchiffrement utilise exactement les mêmes étapes, mais dans le sens inverse.

4.7.2 Algorithme et complexité

Algorithme Disque tournant;

Chiffrement

Entrées : Un message binaire M, un cache et deux grands nombres premiers p et q.

Ecriture des bits du message dans les trous en lui faisant des rotations

Lecture du disque par secteurs

Sortie : Un message chiffré C

Déchiffrement

Entrées : Un message chiffré, p, q et le cache du chiffrement

Remplissage du disque par secteurs

Lecture des bits à partir des trous du caché en faisant des rotations au disque

Sortie : Le message clair M

L'algorithme du chiffrement avec le disque tournant est en $O(pq)$.

4.7.3 Sécurité du disque tournant

En pratique, on préfère choisir p et q deux grands nombres premiers. Ainsi une personne qui intercepte le message et connaît le nombre $n = p \times q$ (n est la taille du message chiffré) ne peut déterminer p et q, car le problème de la factorisation est difficile. Et si elle réussit à le faire, elle sera confrontée à un autre problème, qui est la connaissance des positions des trous, malheureusement aussi difficile. En effet, le nombre de caches possibles est q^p.

Chapitre 5

Complexité et cryptographie

Sommaire

5.1 Introduction

Chaque fois qu'un système cryptographique repose sur une sécurité calculatoire on aimerai mesurer la difficulté à laquelle est confronté le cryptanalyste, c'est la raison pour laquelle on se tourne vers la théorie de la complexité. Nous rappelons ici quelques définitions et résultats classiques à la base de la théorie de la complexité.

5.2 Complexité des algorithmes

5.2.1 Définitions

Définition 5.2.1 *Un problème est une question à laquelle on veut apporter une réponse. Tout problème comporte dans sa description des paramètres formels, en donnant des valeurs à ces paramètres on obtient une instance du problème. Résoudre un problème c'est donner une réponse à toutes ses instances possibles.*

Définition 5.2.2 *Un algorithme de résolution d'un problème P donné est une procédure, décomposable en nombre fini d'opérations élémentaires, transformant une chaîne de caractères représentant les données de n'importe quel exemple du problème P en une chaîne de caractères représentant les résultats de P.*

Le mot algorithme provient du nom du Mathématicien du IX^e siècle, Mohammed ibn-Moussa al-Khawarizmi. Autant dire que les algorithmes, au sens de solution à des problèmes, sont connus et utilisés bien avant les débuts de l'informatique.

Il arrive souvent que des algorithmes conçus pour résoudre le même problème diffèrent fortement entre eux. Ces différences peuvent être bien plus importantes que celles dues au matériel, aux logiciels ou encore aux programmeurs. Il faut donc trouver un moyen pour juger l'efficacité d'un algorithme et pour comparer plusieurs algorithmes entre eux sans tenir compte des machines utilisées.

Définition 5.2.3 *La complexité d'un algorithme mesure le temps d'éxécution et aussi l'espace nécessaire à un algorithme pour résoudre un problème.*

Définition 5.2.4 *Le temps d'exécution d'un algorithme est le nombre d'opérations élémentaires (accès à une cellule mémoire, comparaison de valeurs, opérations arithmétiques, ...) qu'il effectue. On dit que la complexité de l'algorithme est $O(f(n))$ où f est d'habitude une combinaison de polynômes, logarithmes ou exponentielles quand le nombre d'opérations effectuées est borné par $cf(n)$, où c est une constante positive, lorsque n tend vers l'infini (n est la taille du problème).*

5.2.2 Calcul de la complexité

Voici quelques exemples pour illustrer le calcul de la complexité.

Exemple 5.2.1 *Calcul de la somme des éléments d'un vecteur V à n éléments.*

$S := 0$;
Pour $i := 1$ **haut** n **faire**
$S := S + V[i]$;
Fait;

Le temps d'exécution de cet algorithme est égal à $C_1 + n(C_2 + C_3)$ où :
C_1 est le temps d'exécution de l'affectation $S := 0$,
C_2 est le temps d'exécution de l'incrémentation et du test de la variable i dans la boucle,
C_3 est le temps d'exécution de l'instruction $S := S + V[i]$ (affectation et somme).
C_1, C_2 et C_3 sont des constantes, donc le temps d'exécution de l'algorithme s'écrit $an+b = O(n)$ qui est une fonction linéaire en n, où n est le nombre d'éléments du vecteur V ou la taille de l'entrée. On dit que l'algorithme s'exécute en $O(n)$ ou que la complexité de l'algorithme est en $O(n)$.

Exemple 5.2.2 *Recherche d'une valeur val donnée dans un vecteur V à n éléments*

$i := 1$;
Tantque $(i \leq n)$ et $(V[i] \neq val)$ **faire**
$i := i + 1$;
Fait;

$trouve := i \leq n$;

Le temps d'exécution de cet algorithme est égal à $C_1 + n_b\,(C_2 + C_3) + C_4$

C_1 est le temps d'exécution de l'affectation $i := 1$,

n_b est le nombre de fois que la boucle est exécutée,

C_2 est le temps d'exécution du test de la boucle,

C_3 est le temps d'exécution de l'instruction $i := i + 1$,

C_4 est le temps d'exécution de l'instruction $trouve := i \leq n$.

C_1, C_2, C_3 et C_4 sont des constantes, donc le temps d'exécution de l'algorithme s'écrit $an_b + b$. On remarque que ce temps d'exécution dépend des éléments du vecteur (la nature des données). Dans le cas le plus favorable ($n_b = 0$, la valeur val est à la première position), le temps d'exécution est une constante et est noté $O(1)$. Dans le cas le plus défavorable ($n_b = n$, la valeur val n'existe pas), le temps d'exécution est égal à $an + b = O(n)$.
On dit dans ce cas que l'algorithme s'exécute en $O(n)$.

Exemple 5.2.3 *Evaluation d'un polynôme de degré n*

$$P\,(x) = a_0 + a_1x^1 + a_2x^2 + a_3x^3 + ... + a_nx^n$$

L'évaluation directe de la valeur de $P\,(x)$ conduit à une complexité :

$$C_{P(x)} = O\left(n_{additions} + \left((1 + 2 + 3 + ... + (n-1) + n) = \frac{n(n+1)}{2}\right)_{multiplications}\right) = O\left(n^2\right)$$

On doit à Hörner un algorithme plus performant qui utilise une factorisation du polynôme sous la forme :

$$P\,(x) = a_0 + x\,(a_1 + x\,(a_2 + x\,(... + a_n)\,...))$$

On remarque que cette factorisation maintien le même nombre d'additions n mais réduit le nombre de multiplications à n. La complexité qui en découle est donc $O(n)$. Le gain est important, de plus cette factorisation évite tout calcul de puissance comme le montre l'algorithme suivant :

Algorithme évaluation;

Entrées : le degré n du polynôme, V le vecteur des coefficients dans l'ordre croissant et la valeur x

$p := V[n]$

Pour $i := n - 1$ **bas** 0 **faire**

$p := p * x + V[i]$

Sortie : $p = P(x)$

Exemple 5.2.4 *Calcul du déterminant par la méthode directe de Cramer*

Un déterminant d'ordre 2 se calcule de la manière suivante :

$$\begin{vmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{vmatrix} = a_{00}a_{11} - a_{01}a_{10},$$

dont la complexité est de 2 multiplications plus une addition.

Pour un déterminant d'ordre 3, on a :

$$\begin{vmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{vmatrix} = a_{00} \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} - a_{10} \begin{vmatrix} a_{01} & a_{02} \\ a_{21} & a_{22} \end{vmatrix} + a_{20} \begin{vmatrix} a_{01} & a_{02} \\ a_{11} & a_{12} \end{vmatrix}$$

dont la complexité est de 3 multiplications, 2 additions, plus 3 fois la complexité du calcul d'un déterminant d'ordre inférieur. Par récurrence, on peut montrer que la complexité du calcul d'un déterminant d'ordre n est n multiplications, $n - 1$ additions plus n fois la complexité d'un déterminant d'ordre $n - 1$. Par cumul, on arrive alors à $O(n \times n!)$. Faisons l'hypothèse que l'ordinateur utilisé effectue une opération élémentaire en 10^{-9} seconde. On obtient alors les temps de calcul suivants pour les valeurs de n suivantes :

n	5	10	15
temps	6×10^{-7} s	0.04 s	5.45 heures
n	20	50	100
temps	1543 ans	4.8×10^{46} millénaires	2.9594×10^{140} millénaires

Par conséquent, il est nécessaire de faire un calcul de complexité avant de mettre l'algorithme en route.

5.2.3 Classification des algorithmes

Habituellement la plupart des algorithmes admettent un paramètre de base n qui correspond à la taille du problème à résoudre. Cela peut être le degré d'un polynôme, le nombre d'enregistrement dans une base de données ou encore la profondeur d'un arbre. L'analyse de la complexité d'un algorithme reviendra à étudier l'incidence de l'augmentation de n sur la durée d'exécution du programme.

Les algorithmes habituellement rencontrés peuvent être classés dans les catégories suivantes :

- **Complexité constante O(1):** On rencontre cette complexité quand toutes les instructions sont exécutées une seule fois qu'elle que soit la taille n du problème, par exemple le problème de parité d'un nombre.

- **Complexité logarithmique O(logn):** La durée d'exécution croit légèrement avec n. Ce cas de figure se rencontre quand la taille du problème est divisée par une entité constante à chaque itération comme par exemple la recherche dichotomique.

- **Complexité linéaire O(n):** C'est typiquement le cas d'un programme avec une boucle de 1 à n et le corps de la boucle effectue un travail de durée constante et indépendante de n, par exemple analyse des fréquences d'un texte à n caractères.

- **Complexité n-logarithmique O(nlogn) :** Se rencontre dans les algorithme où à chaque itération la taille du problème est divisée par une constante. Un exemple de ce genre de complexité est l'algorithme de tri par fusion.

- **Complexité quadratique $\mathbf{O(n^2)}$:** C'est le cas des algorithmes avec deux boucles imbiquées chacune allant de 1 à n et avec le corps de la boucle interne qui est constant.

- **Complexité cubique $\mathbf{O(n^3)}$:** par exemple trois boucles imbriquées, avec les corps des deux boucles interne qui est constant.

- **Complexité exponentielle $\mathbf{O(2^n)}$:** Les algorithmes de ce genre sont dits "naïfs" car ils sont inefficaces et inutilisable dès que n dépasse 50. On rencontre typiquement ces algorithmes dans les parcours arborescents.

5.2.4 Algorithme efficace

La complexité d'un algorithme est dite polynomiale si elle est en $O(n^k)$, pour un certain entier k, on dit aussi, pour abréger, que l'algorithme lui-même est polynomial. Un algorithme est considéré comme efficace si et seulement si il est polynomial. Il existe plusieurs raisons de s'intéresser à cette classe d'algorithmes.

Remarque 5.2.1 *Le fait qu'un algorithme soit non polynomial ne signifie pas que le problème qu'il résout est difficile, dans ce cadre on cite l'algorithme du simplexe qui est exponentiel, mais le problème de la programmation linéaire qu'il résout est facile.*

5.3 Complexité des problèmes

Dans cette section nous allons parler des problèmes faciles et difficiles et de la manière de les distinguer.
L'expérience montre que certains problèmes sont plus difficiles à résoudre que d'autres. Par exemple, étant donné un nombre, il est très facile de tester si ce nombre est pair ou non : il suffit de regarder si le dernier chiffre appartient à l'ensemble {0,2,4,6,8}. En revanche, il est beaucoup plus difficile de déterminer sa décomposition en produits de facteurs premiers. La théorie de la complexité permet de classer les problèmes de décision (un problème de décision est un problème pour lequel la réponse est oui ou non) en deux classes faciles et difficiles en fonction de la complexité des algorithmes qui existent pour les résoudre.

Cependant il est possible d'associer à chaque problème d'optimisation, un problème de décision en introduisant un seuil k correspondant à la fonction objectif f. Le problème de décision devient : "existe-t-il une solution réalisable (S) tel que $f(S) \leq k$ (ou $\geqslant k$)?".

Définition 5.3.1 *Un problème est dit facile s'il existe un algorithme polynomial pour le résoudre. Sinon le problème est dit difficile.*

Définition 5.3.2 *La complexité d'un problème est la complexité (dans le pire des cas) du meilleur algorithme connu pour le résoudre.*

5.3.1 Classes de complexité

- **La classe P (Polynomial) :** On dit qu'un problème est dans P s'il existe un algorithme polynômial en la taille des données pour le résoudre.

  Exemple 5.3.1 *Le problème de détermination d'un chemin Eulérien dans un graphe appartient à P.*

- **La classe NP (Non deterministic polynomial) :** Cette classe réunit les problèmes de décision pour lesquels la réponse *oui* peut être décidée par un algorithme non-déterministe en un temps polynomial par rapport à la taille de l'instance un algorithme est dit non déterministe s'il comporte des instructions de choix. Pour ces algorithme, si à chaque instruction, le bon choix est effectué, le temps de calcul est polynomial. Si au contraire tous les choix sont énumérés, l'algorithme devient déterministe et son temps d'exécution devient exponentiel.

- **La classe Co-NP :** C'est le nom parfois donné pour l'équivalent de la classe NP, mais avec la réponse *non*.

Pour montrer qu'un problème est dans NP, il suffit de trouver un algorithme qui vérifie si une solution donnée est valide en temps polynomiale.

Exemple 5.3.2 *Le problème de satisfaisabilité (SAT) est dans la classe NP.*

Soit $F(x_1, x_2, ..., x_n)$ une expression booléenne à n variables. Le problème SAT consiste à affecter une valeur vrai ou faux pour chacune des variables x_i de manière à rendre vrai l'expression $F(x_1, x_2, ..., x_n)$.

Un algorithme non déterministe résolvant SAT est comme suit :

Pour $k := 1$ **haut** n **faire**

x_k:=choix(vrai, faux);

Fait;

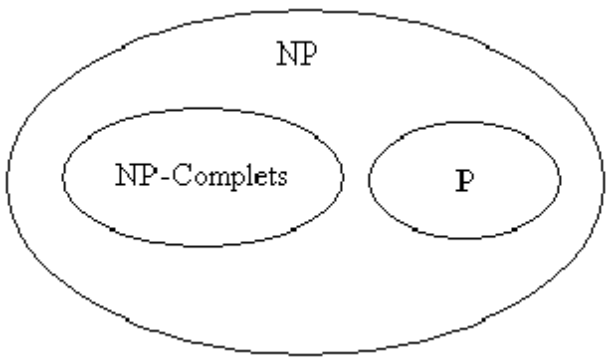
Si $F(x_1, x_2, ..., x_n) = vrai$ F est satisfaisable sinon F n'est pas satisfaisable.

- **Définition 5.3.3** *Un problème A est réductible à un problème B s'il existe un algorithme résolvant A qui utilise un algorithme résolvant B.*

  Définition 5.3.4 *Si l'algorithme résolvant A est polynomial, en considérant un appel à l'algorithme résolvant B comme une opération élémentaire, la réduction est dite polynomiale. On dit que A est polynomialement réductible à B.*

- **La classe NP-Complet :** La classe NP-Complet est une sous-classe des problèmes NP. Un problème est NP-Complet quand tous les problèmes appartenant à NP lui sont polynomialement réductibles. Si on trouve un algorithme polynomial pour un problème NP-Complet, on trouve alors automatiquement une résolution polynomiale de tous les problèmes de la classe NP.

Voici une schématisation de certaines classes :



5.3.2 Conjecture fondamentale de la théorie de la complexité

Tout problème dans P est aussi dans NP, mais on ne sait rien sur l'égalité P = NP. L'attente de la plupart des informaticiens et mathématiciens est que l'égalité soit fausse.

Malheureusement, il n'existe pas de preuve pour cette assertion, ainsi la conjecture P≠NP émise en 1971 est déclarée un des problèmes du millénaire par l'Institut Clay qui propose 1 million de dollars pour sa résolution!.

5.3.3 Existence de problèmes NP-Complets

Théorème 5.3.1 *(de Cook) Le problème de satisfaisabilité est NP-Complet.*

Cook a montré en 1971 que tous les problèmes de la classe NP sont réductibles au problème de la satisfaisabilité d'une expression logique quelconque.

5.3.4 Exemples de problèmes NP-Complets

Nous présentons ici quelques problèmes NP-Complets dont la NP-complétude est démontrée grâce à l'existence du premier problème NP-Complet SAT et à l'aide des réductions polynomiales.

Problème du stable

Trouver dans un graphe fini d'ordre n un ensemble de m sommets deux à deux non adjacents ($m \leq n$).

Problème du sac à dos

Ce problème modélise une situation analogue au remplissage d'un sac à dos, ne pouvant supporter plus d'un poids P, avec n objets d'un ensemble ayant chacun un poids p_i et une valeur w_i. Les objets mis dans le sac à dos doivent maximiser la valeur totale, sans dépasser le poids maximum. Mathématiquement il s'agit de trouver un vecteur binaire $X = (x_1, x_2, ..., x_n)^t$ tel que :

$$\begin{cases} \sum_{i=1}^{n} p_i x_i \leq P & x_i \in \{0,1\} \\ \max Z(x) = \sum_{i=1}^{n} w_i x_i \end{cases}, \quad \text{où} \begin{cases} x_i = 1 \text{ si l'objet } i \text{ est pris} \\ x_i = 0 \text{ si non} \end{cases}$$

Problème du cycle Hamiltonien

Soit un graphe fini d'ordre n. Existe t-il un cycle de longueur n passant une et une seule fois par tous les sommets.

5.3.5 Comment déterminer la classe d'un problème?

Lorsque l'on est confronté à un problème P dont on ignore la classe de complexité, la démarche consiste soit :

- à supposer que le problème est facile, et on cherche alors un algorithme polynomial pour le résoudre
- à supposer que le problème est NP-Complet, et on cherche alors à le démontrer comme suit :

1. Choisir un problème $P2$, NP-Complet connu.
2. Construire une réduction f, qui transforme $P2$ vers P telle que : I une oui-instance de $P2 \Leftrightarrow f(I)$ une oui-instance de P,
3. Prouver que f est une transformation polynomiale.

5.4 Cryptographie et complexité

Les notions présentées ci-dessus nous permettent d'évaluer la difficulté d'un problème de cryptographie donné, à savoir la complexité d'un algorithme de chiffrement ou de cryptanalyse. Notons qu'il est intéressant de fonder des algorithmes de chiffrement sur des problèmes difficiles ou encore NP-Complets car la résolution de ces derniers est réputée difficile, on obtient ainsi une garantie sur la sécurité du schéma de chiffrement lui même.

Parmi les problèmes NP-Complets nous citons le problème du sac à dos qui a été à la base des algorithmes de chiffrement publique, le problème de factorisation qui est à la base du cryptosystème proposé au chapitre précédent et le problème SAT qui constitue le sujet de la seconde proposition. Dans cette proposition de chiffrement hybride exposée au

septième chapitre, nous avons tout d'abord expliqué la façon de générer la clé privée qui va être chiffrée avec l'algorithme RSA, ensuite nous avons décrit la méthode de chiffrement et de déchiffrement.

5.4.1 Le problème de la primalité

Instance : Un entier n

Question : n est-il premier?.

Les nombres premiers jouent un rôle très fondamental en mathématiques, il est donc intéressant de savoir les reconnaître. Ce problème qui semble facile lorsqu'on s'intéresse à des nombres de petite taille, devient plus délicat dans le domaine des grands nombres (ayant plus de 150 chiffres). Nous verrons dans le chapitre suivant l'importance de ces nombres en cryptographie.

Depuis l'invention de la théorie de la complexité le problème de la primalité que nous noterons PR passionnait les chercheurs qui souhaitaient prouver que $PR \in P$.
En août 2002, Manindra Agrawal de l'Indian Institute of Technology à Kanpur et deux de ses étudiants Neeraj Kayal et Nitin Saxena ont trouvé un algorithme polynomial en $O(\log^{12}(n))$ qui teste la primalité de n, résolvant ainsi une conjecture déjà ancienne en théorie de la complexité.

Nous ne décrirons pas ici cet algorithme, nous renvoyons le lecteur à la réference [32]. L'intérêt théorique de cet algorithme est grand, car il est le premier algorithme déterministe polynomial, cependant il est moins efficace que les algorithmes probabilistes (qu'on citera au chapitre suivant), ce qui explique entre autre qu'il reste peu utilisé par les compagnies informatiques pour tester les nombres premiers.

5.4.2 Le problème de la factorisation d'entiers

Le problème de la factorisation d'un entier N consiste à trouver sa décomposition en facteurs premiers. Il s'agit donc d'écrire N sous la forme :

$$N = \prod p_i^{e_i} = p_1^{e_1}.p_2^{e_2}...p_k^{e_k},$$

où les p_i sont des nombres premiers et $e_i \geq 1 \; \forall \; i \in \{1, ..., k\}$.

Le problème de décision associé au problème de factorisation est : étant donné un entier N et un entier $M \leq N$, existe t'il un diviseur de N inférieur à M ?
Ce dernier a été démontré appartenant à la classe $NP \cap Co - NP$.

Chapitre 6

Cryptographie à clé publique

Sommaire

6.1 Introduction

L'avancée majeure en cryptographie a incontestablement été la publication, en 1976, de l'article "*New directions in cryptography*", de Whitfield Diffie et Martin Hellman [57]. Cet article introduit le concept révolutionnaire de cryptographie à clé publique qui résout le problème de distribution des clés posé par la cryptographie à clé secrète. Même si les auteurs ne donnent pas de réalisation pratique d'un système à clé publique, les propriétés d'un tel système sont clairement énoncées. Depuis lors, la littérature sur ce sujet n'a cessé de se développer .

Dans un système à clé publique ou asymétrique, chaque individu possède deux clés distinctes. Une clé privée qui ne le quitte jamaiset une clé publique qu'il met à la disposition de tous dans un annuaire. Cette clé permet de chiffrer des messages que seul le propriétaire de la clé privée pourra déchiffrer.

Un grand avantage du chiffrement à clé publique est que la clé privée correspondante à la clé publique d'une personne n'est pas partagée, elle ne quitte jamais son propriétaire. Il est donc plus facile de s'assurer qu'elle n'est pas compromise. Pour cette raison on fait la distinction entre clé privée et clé secrète.

Le principe sur lequel repose le chiffrement à clé publique est celui de *fonction à sens unique avec trappe.*

6.2 Définitions

• Une fonction $f : M \longrightarrow C$ est dite à sens unique (one way function) si :

- pour tout x de M, il est **facile** de calculer $f(x)$ ($f(x)$ peut être calculée en temps polynomial) et

- il est **difficile** de trouver, pour la plupart des $y \in f(M)$ un $x \in M$ tel que : $f(x) = y$, à moins d'exécuter un nombre prohibitif d'opérations, ou d'avoir une chance sur laquelle il est déraisonnable de compter (la fonction f est difficile à inverser).

• Une fonction à sens unique est dite à trappe (ou à brèche) secrète si elle est à

sens unique et le calcul de l'inverse devient facile dès qu'on détient une information supplémentaire (la trappe), soit une fonction g telle que $g \circ f = Id$.

La construction des couples (f, g) doit être facile et la publication de f ne doit rien révéler sur g.

> Le mélange de peinture bleue et de peinture jaune est facile à réaliser, leur séparation est ensuite très difficile.

Les problèmes NP-Complets semblent de bons candidats pour construire des fonctions à sens unique, nous allons examiner quelques exemples.

6.3 L'algorithme de Merkle-Hellman

L'algorithme de Merkle-Hellman proposé par Ralph Merkle et Martin Hellman en 1978, est historiquement le premier algorithme montrant le fonctionnement d'un système à clé publique. Il repose sur le problème du sac à dos (knapsack problem).

Soient un ensemble S de nombres entiers positifs et un entier P. Existe t-il une partie A de S telle que $\sum_{i \in A} p_i = P$?.

Ce problème est NP-Complet dans le cas général mais il existe des instances extrêmement faciles à résoudre, ce sont celles faisant intervenir les séquences super-croissantes, c'est-à-dire $\forall j, 1 < j \leq n, \sum_{i=1}^{j-1} p_i < p_j$, l'algorithme glouton suivant permet de résoudre cet instance.

Algorithme glouton

Pour $i := 1$ **haut** n **faire**

Si $P \geq p_i$ **alors**

$P := P - p_i$

$x_i := 1$

Sinon

$x_i := 0$

Si $P = 0$ **alors** $(x_1, ..., x_n)^t$ est une solution sinon pas de solution.

6.3.1 L'idée de Merkle-Hellman

L'idée de Merkle-Hellman est de transformer un problème du sac à dos trivial basé sur une séquence super-croissante (qui sert de clé privée) en un problème du sac dos général pour en faire une clé publique. Nous allons présenter tout de suite les détails.

6.3.2 Génération des clés

- Choisir une suite super-croissante $\{a_1, ..., a_n\}$, et un entier M supérieur à $\sum_{i=1}^{n} a_i$.
- Choisir au hasard un entier $W \in \{1, ..., M-1\}$ premier avec M.
- Choisir au hasard une permutation σ de de l'ensemble $\{1, ..., n\}$.
- Calculer $b_i = a_{\sigma(i)} W \bmod M$ pour tout i dans $\{1, ..., n\}$.

La clé publique du système est la suite (a priori quelconque) $b_1, ..., b_n$. La clé privée consiste en M, W, la permutation σ et la suite super-croissante $a_1, ..., a_n$.

6.3.3 Le chiffrement

Soit le message écrit en binaire $m_1 m_2 ... m_n$ (si le message est trop long, il est coupé en blocs de n bits), le message chiffré est :

$$C = \sum_{i=1}^{n} b_i m_i$$

où $b_1, ..., b_n$ est la clé publique calculée précédemment.
Quelqu'un qui intercepte C et connait la clé publique ne peut retrouver les m_i en un temps raisonnable, car la résolution du sac à dos général est un problème difficile.

6.3.4 Le déchiffrement

Soit le message chiffré C, celui qui possède la clé privée procède comme suit :

- Il calcule $D = W^{-1} C \bmod M$ (W^{-1} est obtenu en utilisant l'algorithme d'Euclide étendu).

- Il résoud ensuite le sac à dos super-croissant $D = \sum_{i=1}^{n} e_i a_i$.
- Après avoir calculé les e_i, il déduit les m_i par la relation : $m_i = e_{\sigma(i)}$.

La justification de cette phase de déchiffrement est simple. En effet, on a modulo M :

$$\begin{aligned} D &= W^{-1}C = W^{-1}\sum_{i=1}^{n} b_i m_i \\ &= W^{-1}\sum_{i=1}^{n} a_{\sigma(i)} W m_i \\ &= \sum_{i=1}^{n} a_{\sigma(i)} m_i. \end{aligned}$$

Exemple 6.3.1 *Soit* $n = 10$.

Choix de la clé privée : $\{a_1, ..., a_{10}\} = \{4, 9, 30, 70, 185, 451, 1306, 3534, 6517, 17486\}$, $M = 50349$, $W = 36334$ et la permutation :

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 8 & 6 & 7 & 5 & 2 & 4 & 10 & 3 & 1 & 9 \end{pmatrix}$$

La clé publique est :

$$\{b_1, ..., b_{10}\} = \{14406, 23206, 23446, 25373, 24912, 25930, 32642, 32691, 44638, 47680\}$$

Pour envoyer le message 1001000110 on transmet :

$$C = 14406 + 25373 + 32691 + 44638 = 117108$$

Le récepteur utilise sa clé privée pour déchiffrer, il calcule :

$$D = W^{-1}C \bmod M = 7865 \times 117108 \bmod 50349 = 3753$$

Il résoud le problème du sac à dos et il trouve :

$$(e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}) = (1, 0, 1, 0, 1, 0, 0, 1, 0, 0)$$

Sachant que $x_i = e_{\sigma(i)}$, $i = 1, n$, il déduit le message initial 1001000110.

6.3.5 La chute de Merkle-Hellman

Le cryptosystème de Merkle-Hellman fut très vite démontré non sûr. En 1982, Shamir casse (en théorie seulement) l'algorithme et Adleman le casse en pratique en 1983.
En 1995, personne ne croit en la possible utilité du chiffre de Merkle-Hellman car toutes ces variantes sont cassées. Plus de détails sur la cryptanalyse du sac à dos de Merkle Hellman et notamment de nombreuses références, sont disponibles dans le chapitre 8 de [4].

6.4 L'algorithme RSA

Le RSA (nom condensé de ces inventeurs Ron Rivest, Adi Shamir et Leonard Adleman) est un algorithme à clé publique inventé en 1978 qui sert aussi bien au chiffrement de documents, qu'à l'authentification. Il est basé sur l'impossibilité, tout au moins à ce jour, de factoriser rapidement de grands nombres composés.
Son fonctionnement repose sur les résultats d'arithmétique présentés au troisième chapitre, comme par exemple le théorème d'Euler.

6.4.1 Fonctionnement du RSA

Si Meriem et Sara souhaitent communiquer en utilisant le RSA, elles procèdent de la façon suivante :

Génération des clés

La construction des clés du RSA se fait en trois étapes :

1. Meriem choisi deux grands nombres premiers p, q et calcule $n = pq$ et $\phi(n) = (p-1)(q-1)$. Le nombre n est rendu public, p et q doivent rester secrets et sont donc détruits une fois les clés générées.

2. Elle choisi ensuite aléatoirement un entier $e < \phi(n)$ tel que e soit premier avec $\phi(n)$.

3. Elle calcule le nombre d, l'inverse de e dans $\mathbb{Z}/\phi(n)\mathbb{Z}$ en utilisant l'algorithme d'Euclide étendu, i.e., $ed \equiv 1 \bmod \phi(n)$.

Le couple (n, e) constitue la clé publique de Meriem et le couple (n, d) constitue sa clé privée.

Le chiffrement

Dans RSA, les blocs du message en clair sont représentés par des entiers compris entre 0 et $n-1$, si n a k chiffres les blocs auront $k-1$ chiffres chacun sauf peut être le dernier. Pour envoyer un message M à Meriem, Sara va donc chercher la clé publique de Meriem et elle calcule le message chiffré $C \equiv M^e \bmod n$, et c'est ce dernier nombre qu'elle envoie à Meriem.

Le déchiffrement

Meriem reçoit C, elle calcule grâce à sa clé privée $D = C^d$.

$$\begin{aligned} C^d &= (M^e)^d (\bmod n) = M^{1+k\phi(n)} (\bmod n) \\ &= M.M^{k\phi(n)} (\bmod n) = M.(M^{\phi(n)})^k (\bmod n) \end{aligned}$$

D'après le théorème d'Euler, $M^{\phi(n)} \bmod n = 1$. Ella a donc reconstitué le message initial.

Algorithme RSA;

Chiffrement

Entrées : Un message clair M et une clé publique (n, e)

$C := M^e \bmod n$

Sortie: Un message chiffré C

Déchiffrement

Entrées : Un message chiffré C et la clé privée (n, d)

$M := C^d \bmod n$

Sortie : Le message clair M

Exemple 6.4.1 *1.* ***Génération des clés :***

- *Pour se fabriquer un couple de clés, Sara commence par choisir deux nombres premiers aléatoires soient :* $p = 17$, $q = 11$. *Elle peut alors calculer* n *et* $\phi(n)$, $n = 187$, $\phi(n) = 160$.
- *Sara choisit aléatoirement l'exposant public* e *inférieur à* $\phi(n)$ *et premier avec, soit* $e = 7$.
- *Reste à calculer* d, *l'inverse de* e *modulo* $\phi(n)$. *Pour cela, Sara peut utiliser l'algorithme d'Euclide étendu :*

$$\begin{aligned} ed \bmod \phi(n) &= 1 \\ d &= e^{-1} \bmod \phi(n) d = 7^{-1} \bmod 160 = 23. \end{aligned}$$

Sara peut maintenant publier sa clé publique constituée de n *et* e. *Elle détruit* p *et* q *et conserve précieusement la clé privée* d *qui permis de déchiffrer les messages qui lui seront envoyés.*

2. ***Le chiffrement***

- *Meriem utilise la clé publique de Sara pour chiffrer le message "arrive le 17". Pour commencer elle convertit son texte en chiffres en prenant par exemple a=01, b=02, ..., z=26, espace=00, 0=30, 1=31, 2=32, ..., ce qui lui donne :* 01181809220500120500313 7.
- *Elle regroupe ce nombre en tranches ayant moins de chiffres que la clé* $n = 187$, *soit en tranches de deux chiffres :*$M_1 = 01$, $M_2 = 18$, $M_3 = 18$, $M_4 = 09$, $M_5 = 22$, $M_6 = 05$, $M_7 = 00$, $M_8 = 12$, $M_9 = 05$, $M_{10} = 00$, $M_{11} = 31$, $M_{12} = 37$.
- *Le premier bloc est chiffré avec la clé publique de Sara :* $C_1 = M_1^e \bmod n = 01^7 \bmod 187 = 1$.

  Meriem répète cette opération sur les autres blocs et obtient : 1 171 171 70 44 146 0 177 146 0 47 125 181.

- *Ajoute des 0 à gauche si nécessaire pour n'avoir que des nombres de trois chiffres comme la clé n et obtient le message chiffré :*

$$C = 001\ 171\ 171\ 070\ 044\ 146\ 000\ 177\ 146\ 000\ 047\ 125\ 181$$

Elle peut maintenant envoyer ce message à Sara par un canal qui n'a pas besoin d'être sécurisé.

Remarque 6.4.1 *Pour cet exemple, nous avons choisi une petite clé n et nous remarquons que le "r" et le "e" du message clair sont toujours chiffrés* 171 *et* 146 *respectivement, par conséquent une attaque par analyse des fréquences est possible, d'où la nécessité du bon choix des clés.*

3. ***Le déchiffrement***

- *Pour déchiffrer le message, Sara utilise sa clé privée. Elle reconstitue le message après avoir découpé le message chiffré en tranches de trois chiffres puisque la clé $n = 187$ en a trois. Elle va ainsi trouver pour le premier bloc :*

$$M_1 = C_1^{23} \bmod n = 001^{23} \bmod 187 = 01$$

- *Ainsi de suite. Il ne lui reste plus qu'à prendre son tableau de correspondance alphabétique pour recueillir "arrive le 17". Bien entendu, le chiffrement et le déchiffrement se font sur ordinateur vu la lenteur des opérations.*

6.4.2 La sécurité du RSA

La sécurité du RSA repose sur la difficulté de factoriser de grands nombres, car d'après le théorème suivant la détermination de la clé privée d à partir de la clé publique (n, e) est équivalente au problème de factorisation de n.

Théorème 6.4.1 *Les deux problèmes suivants :*

(i) calculer le nombre d, connaissant le couple (n, e)

(ii) factoriser n

sont équivalents, dans le sens qu'une solution de l'un conduit à une solution de l'autre [49].

Le record établi le 22 Août 1999, est la factorisation d'un entier à 155 chiffres (soit une clé de 512 bits) en faisant travailler près de 285 ordinateurs pendant cinq mois. Ce record a été faiblement amélioré au cours des années :

- 160 chiffres en mars 2003 (530 bits),
- 174 chiffres en décembre 2003 (576 bits),
- 200 chiffres (660 bits) en mai 2005 et c'est le dernier record connu.

Il faut donc, pour garantir un minimum de sécurité, choisir des clés réellement grandes et réellement aléatoires. En pratique, pour n, les experts recommandent une clé de 768 bits pour un usage privé, et de 1024 bits (309 chiffres), voire 2048 bits (617 chiffres) pour un usage sensible.

Si l'on admet que la puissance des ordinateurs double tous les 18 mois, une clé de 2048 bits devrait tenir jusqu'à 2079, à moins qu'une percée théorique ou technique n'y parvienne avant.

6.4.3 Implémentation du RSA

On a vu précédemment que le fonctionnement du RSA repose sur le bon choix des clés, c'est-à-dire la génération des deux grands nombres premiers p et q ensuite e et d qui vérifient les conditions. La taille de ces entiers dépasse les types prédéfinis en Pascal, ainsi nous avons mis en oeuvre une structure de chaîne pour représenter des grands entiers puis on a programmé les opérations de base sur ces nombres, ces dernières ont servi pour les autres fonctions.

Génération des clés

Il est facile de construire de grands nombres premiers. Cela est du au théorème des nombres premiers qui donne la distribution asymptotique des nombres premiers. Le

théorème des nombres premiers assure qu'après en moyenne 125 essais on devrait obtenir un nombre premier.

Pour construire un grand nombre premier on génère au hasard le nombre nécessaire de chiffres à condition que le premier chiffre soit différent de zéro et le dernier n'appartienne pas à $\{0, 1, 2, 4, 6, 8\}$ si non le nombre serait pair. Une fois ce nombre obtenu, on effectue des tests de primalité (test de Fermat et de Miller-Rabin) pour savoir si le nombre tiré est premier.

Les grands nombres entiers aléatoires

Pour générer des nombres aléatoires on a utilisé la fonction Random pour générer aléatoirement des chiffres, ensuite on teste la primalite du nombre obtenu.
La suite des nombres premiers semble imprévisible, cependant certaines formules en donnent, par exemple le polynôme $n^2 - n + 41$ pour $0 \leq n \leq 40$.

On appelle nombre de Mersenne un nombre de la forme $2^n - 1$, noté M_n. Les nombres de Mersenne ne sont pas tous premiers, par exemple $M = 15$ n'est pas premier, en effet pour que M_n soit premier il est nécessaire que n le soit.
Les records des plus grands nombres premiers sont depuis des années réalisés à partir des nombres de Mersenne en voici quelques exemples :

Date de la découverte	Valeur de n	Nombre de chiffres de M_n
17-11-03	20 996 011	6 320 430
15-05-04	24 036 583	7 235 733
18-02-05	25 964 951	7 816 230
15-12-05	30 402 457	9 152 052
4-10-06	32 582 657	9 808 358

Néanmoins ces nombres ne sont jamais utilisés en cryptographie car il sont particuliers, et il serait facile de les reconnaître.

Les tests probabilistes de primalité

Nous avons déjà annoncé qu'il existe un algorithme deterministe polynomial pour le test de primalité d'un entier [32], mais en pratique, les algorithmes probabilistes sont suffisants et efficaces, car leur temps de calcul est nettement inférieur, cependant ils affirment qu'un entier est premier avec une probabilité ≈ 1 ce qui n'est pas une certitude. Ils est possible d'obtenir des probabilités de primalité très grandes en effectuant plusieurs fois ces tests et même en utilisant plusieurs tests simultanéments. Parmi ces algorithmes on cite le test de Fermat et le test de Miller Rabin.

Le test de Fermat

Le test de primalité de Fermat est basé sur la version suivante du petit théorème de Fermat.

Le petit théorème de Fermat fournit un test qui peut permettre de montrer qu'un nombre n est composé. En effet, en calculant $2^{n-1} \bmod n$, si on trouve un résultat distinct de 1, on est certain que n n'est pas premier. Par contre, si on trouve 1, cela ne veut pas dire que n est premier. On peut alors prendre d'autres valeurs a que 2, et calculer a^{n-1}. Si l'on trouve 1 à chaque fois, cela ne prouvera pas que n est premier car il existe des nombres composés n pour lesquels on a $a^{n-1} \equiv 1[n]$ pour tout a premier avec n, ce sont les nombres de Carmichael. On croyait qu'il n'existerait qu'une finité, cependant Alford, Granville et Pomerance ont démontré en 1994 que ces nombres sont infinis[58], et qu'à partir d'une certaine borne, ils ne seraient pas si rares qu'on le pensaient.
Un renforcement de ce test a été donné par Rabin et Miller en 1976.

Le test de primalité de Miller Rabin

Le test de primalité de Miller-Rabin est un test de primalité probabiliste célèbre, sa version originale due à G.L Miller est déterministe, et M.O Rabin l'a modifiée pour obtenir un algorithme probabiliste.

Proposition 6.4.1 *Soit n un entier impair. On pose $n-1 = t \times 2^s$ avec t impair. On*

a alors, $\forall a \in \{2, ..., n-1\}$:

$$\begin{aligned} a^{n-1} - 1 &= \left(a^t\right)^{2s} - 1 \\ &= (a^t - 1)(a^t + 1)(a^{2t} + 1)...(a^{(2^{s-1})t} + 1) \end{aligned}$$

Si n est premier, alors d'après le théorème de Fermat $a^{n-1} - 1 \equiv 0 \bmod n$*, donc :*

- *(P1) Soit* $a^t - 1 \equiv 0 \bmod n \Leftrightarrow a^t \equiv 1 \bmod n$
- *(P2) Soit il existe* $i \in [0, s[$ *tel que* $a^{t2^s} + 1 \equiv 0 \bmod n \Longleftrightarrow a^{t2^s} \equiv -1 \bmod n$

Le test de Miller Rabin est basé sur cette proposition :

- (P1) et (P2) non vérifiées $\Rightarrow n$ est composé.
- (P1) ou (P2) vérifiée $\Rightarrow n$ est probablement premier.

Un entier a qui ne vérifie ni (P1) ni (P2) pour n composé, est dit **témoin de Miller** pour n, on notera $t(n)$ le nombre de témoins de n.

Théorème 6.4.2 *(Rabin) [36] Pour chaque nombre impair composé* $n > 9$*, le nombre de témoins de n vérifie* $\dfrac{t(n)}{n-1} \geq \dfrac{3}{4}$.

Les deux résultats suivants découlent de ce théorème. Si a n'est pas un témoin, on peut conclure avec une probabilité supérieure à $\dfrac{3}{4}$ que n est premier. Ainsi, supposons que l'on ne trouve aucun témoin pour n après avoir effectué k fois le test de Miller Rabin (pour différents choix de a), alors la probabilité que n soit premier est supérieure à $1 - \left(\dfrac{1}{4}\right)^k$.

Exemple 6.4.2 $n = 561 \Rightarrow n - 1 = 560 = 2^4.35 = 2^s.t.$

On choisit $a = 5$.

$5^t \equiv 23 \neq 1 \bmod 561.$

$5^{2.t} \equiv 529 \neq -1 \bmod 561.$

$5^{2^2.t} = 5^{4.35} \equiv 463 \neq -1 \bmod 561.$

$5^{2^3.t} = 5^{8.35} \equiv 67 \neq -1 \bmod 561$

Donc $n = 651$ *n'est pas premier et* 5 *est un témoin de composition de n. Notons que* 561 *est le plus petit nombre de Carmichael.*

Exponentiation modulaire rapide

L'exponentiation modulaire est au coeur de nombreux cryptosystèmes à clé publique parmi lesquels : le chiffrement, la signature RSA et l'échange de clé dans les systèmes de chiffrement hybrides. Ainsi de nombreuses méthodes pour calculer efficacement une exponentiation modulaire ont donc été proposées, parmi lesquelles celle ci-dessous.
Soit $m \in \mathbb{N}$, et $e \geq 1$, pour calculer $m^e \bmod n$, on utilise la formule :

$$m^e \bmod n = \begin{cases} m \bmod n & \text{si } e = 1 \\ (m^{e/2})^2 \bmod n & \text{si } e \text{ est pair} \\ (m^{(e-1)/2})^2 . m \bmod n & \text{si } e \text{ est impair} \end{cases}$$

Soient $m = 324$, $n = 531$ et $e = 9$. $m = 324$, $m^2 = 104976$, ..., $m^9 = 39346408075296537575424$, $m^9 \bmod n = 441 \rightharpoonup 9$ opérations sur des entiers de taille croissante, tandis que l'application de la formule donne :

$$\left.\begin{aligned} m^2 \bmod n &= 369 \\ m^4 \bmod n = (m^2)^2 \bmod n &= 225 \\ m^9 \bmod n = (m^4)^2 \times m \bmod n &= 441 \end{aligned}\right\} \text{ 3 opérations sur des entiers de taille inférieure}$$

Lorsque m, e et n sont donnés le coût du calcul de $m^e \bmod n$ par cette formule est en $O(\log^2 n)$.

6.5 Le chiffre de Rabin

Le cryptosystème de Rabin est un cryptosystème asymétrique basé sur la difficulté du problème de la factorisation (comme le RSA). Il a été inventé en 1979 par Michael Rabin.

6.5.1 Description de la méthode

- **Génération des clés :**

  Sara choisit deux nombres premiers p et q, tous deux congrus à 3 modulo 4, et calcule leur produit $n = pq$. Elle diffuse n tout en gardant secrets p et q.

- **Chiffrement :**

Pour chiffrer un message M qui doit être représenté par un nombre inférieur à $n-1$, Meriem utilise la formule : $C = M^2 \bmod n$

- **Déchiffrement :**

Lorsque Sara reçoit C, elle déchiffre en calculant les racines carrées modulo n de C . Notons que la fonction de chiffrement n'est pas injective. Il existe en effet quatre nombres clairs qui peuvent se chiffrer en le même nombre chiffré. Pour cela elle calcule d'abord les racines r_p et r_q modulo p et q respectivement :

$$r_p = C^{(p+1)/4}[p], r_q = C^{(q+1)/4}[q]$$

Puis, elle utilise le théorème des restes chinois, pour obtenir les quatre racines modulo n $r = (\pm upr_p \pm vqr_q) \bmod n$ où u et v vérifient : $up + vq = 1$.
L'une de ces racines est le message initial M, il existe des méthodes qui permettent de distinguer le bon message parmi les quatre possibles.

6.5.2 Exemple

Sara désire envoyer le message "je te blâme" à Meriem, elle va le chiffrer en utilisant la clé publique $n = 48010717$. Elle va le décomposer en blocs de trois lettres et au début de chacun elle ajoute '*', ensuite elle le converti en son équivalent numérique en utilisant la table suivante :

a	b	c	d	e	f	g	h	i	j	k	l	m	n
00	01	02	03	04	05	06	07	08	09	10	11	12	13
o	p	q	r	s	t	u	v	w	x	y	z	*	
14	15	16	17	18	19	20	21	22	23	24	25	26	

ainsi, *jet *ebl *ame devient 26090419 26040111 26001204.
Pour chiffrer les trois blocs elle calcule : $c_1 = m_1^2 \bmod n = (26090419)^2 \bmod 48010717 = 46850914$, $c_2 = (26040111)^2 \bmod 48010717 = 5842871$ et $c_3 = (26001204)^2 \bmod 48010717 = 12031786$. En fin elle transmis le message : 46850914 5842871 12031786.

Meriem qui possède la clé privée $(p, q) = (6911, 6947)$ est la seule qui peut déchiffrer le message en utilisant la formule : $M = \sqrt{C} \bmod n$.
Pour déchiffrer le premier bloc, Meriem doit résoudre l'équation

$$m_1 = (46850914)^{\frac{1}{2}} \bmod 48010717$$

ce qui revient d'après le théorème des restes chinois à résoudre le système suivant :

$$\begin{cases} r_p = (46850914)^{1/2} \bmod 6911 = (46850914)^{(6911+1)/4} \bmod 6911 = 1394 \\ r_q = (46850914)^{1/2} \bmod 6947 = (46850914)^{(6947+1)/4} \bmod 6947 = 2513 \end{cases}$$

A l'aide de l'algorithme d'Euclide étendu, elle détermine les coefficients u et v tels que $up + vq = 1$.

$$\begin{aligned} v &= (6947)^{-1} \bmod 6911 = 192 \\ u &= (6911)^{-1} \bmod 6947 = 6754 \end{aligned}$$

En posant $r = (\pm upr_p \pm vqr_q) \bmod n$, elle obtient les quatre racines suivantes :

$$r_1 = 43796401,\ r_2 = 21920298,\ r_3 = \mathbf{26090419},\ r_4 = 4214316$$

La valeur qui donne le message initial est celle qui commence par 26, donc 26090419 c'est-à-dire les lettres *jet.

Elle procède de la même façon avec les deux derniers blocs, elle trouvera le message clair.

6.6 La signature

6.6.1 Le principe de la signature

Un procédé de signature numérique ou électronique consiste à adjoindre au texte clair une valeur qui dépend simultanément du message et de son auteur, elle permet de vérifier l'intégrité du message reçu.

La signature électronique assure également une fonction de non-répudiation, c'est-à-dire qu'elle permet d'assurer que l'expéditeur a bien envoyé le message (autrement dit elle empêche l'expéditeur de nier d'avoir expédié le message).

Un schéma de signature doit posséder un certain nombre de propriétés. En particulier, il doit être en pratique impossible de contrefaire une signature : seul le détenteur de la clé secrète peut signer en son nom. La signature ne doit plus être valide si le message clair a été modifié et il doit être impossible de réutiliser une signature. Notons que dans ce cas la confidentialité du message n'est pas importante.

6.6.2 Exemple de signature RSA

Le principe de la signature RSA est d'inverser les rôles des exposants e et d par rapport à leur utilisation dans un schéma de chiffrement, reprenant l'exemple de chiffrement RSA précedent.

Phase de génération des clés : Identique à la génération des clés de RSA.

Phase de signature :

- Pour signer le message "arrive le 17" Sara utilise ses clés $n = 187$ et $d = 23$.
- Pour commencer elle convertit le texte en chiffres soit 011818092205001205003137, puis elle calcule : $s_1 = m_1^d \bmod n = 01^{23} \bmod 187 = 1$, $s_2 = s_3 = 18^{23} \bmod 187 = 35$, $s_4 = 9^{23} \bmod 187 = 36$, $s_5 = 22^{23} \bmod 187 = 44$, $s_6 = 05^{23} \bmod 187 = 180$, $s_7 = 00^{23} \bmod 187 = 0$, $s_8 = 12^{23} \bmod 187 = 177$, $s_9 = s_6 = 180$, $s_{10} = 0$, $s_{11} = 31^{23} \bmod 187 = 91$, $s_{12} = 37^{23} \bmod 187 = 130$.
- Elle envoie à Meriem,

  $s = 001035035036044180000177180000091130$ et $m = 011818092205001205003137$.

Phase de déchiffrement :

- Meriem reçoit s et m calcule $\acute{m}_1 = s_1^e \bmod n = 001^7 \bmod 187 = 01 = m_1$, $\acute{m}_2 = s_2^e \bmod n = 035^7 \bmod 187 = 18 = m_2$ et ainsi de suite elle trouve $\acute{m} = m$.
- Meriem est alors sûre que m a été signé par la clé privée de Sara.

Le système décrit dans l'exemple ci-dessus comporte des inconvénients. Il est lent, et il produit un volume énorme de données. Une amélioration de ce concept consiste à utiliser une fonction de hachage à sens unique dans le processus.

Définition 6.6.1 *(fonction de hachage à sens unique)*

Une fonction de hachage est une fonction qui convertit une chaîne de longueur quelconque en une chaîne de taille inférieure et généralement fixe, la chaîne résultante est appelée empreinte, haché ou condensée de la chaîne initiale.

Une fonction de hachage à sens unique est une fonction de hachage qui est en plus une fonction à sens unique, il est aisé de calculer l'empreinte d'une chaîne donnée, mais il est difficile d'engendrer des chaînes qui ont une empreinte donnée, et donc de déduire la chaîne initiale à partir de l'empreinte. On demande généralement en plus à une telle fonction d'être sans collision, c'est-à-dire qu'il soit impossible de trouver deux messages ayant la même empreinte. On utilise souvent le terme fonction de hachage pour désigner une fonction de hachage à sens unique sans collision.

Exemple 6.6.1 *Voici un exemple simple de fonction de hachage sur l'ensemble des chaînes de longueur l à valeurs dans $\{0, ..., N-1\}$:*

$$h(x) = \sum_{i=0}^{l-1} x_i B^{l-1-i} \bmod N$$

où B est une puissance de 2, on prend d'habitude 128 ou 256 et N est un nombre premier.

Dans cet exemple, pour un message de longueur l aussi grande que l'on veut, la fonction h produit une sortie de longueur réduite. La fonction de hachage assure que, si l'informationait été changée en quoi que ce soit –même d'un seul bit– une sortie totalement différente serait produite.
Les fonctions de hachage les plus utilisées actuellement sont :

- **MD5** (MD signifiant Message Digest) : Développée par Rivest en 1991, MD5 crée une empreinte digitale de 128 bits à partir d'un texte de taille arbitraire. MD5 manipule le texte d'entrée par blocs de 512 bits.

- **SHA** (pour Secure Hash Algorithm) crée des empreintes d'une longueur de 160 bits, SHA-1 est une version améliorée de SHA datant de 1994 et produisant une empreinte de 160 bits à partir d'un message d'une longueur maximale de 2^{64} bits en le traitant par blocs de 512 bits.

Pour plus de détails sur les fonctions de hachage le lecteur peut consulter [4] et [17].

6.6.3 La signature numérique

Le principe de la signature numérique est le suivant :

1. Soumettre le document M à une fonction de hachage H (connue publiquement) pour produire un condensé h_M de taille fixe i.e., $H(M) = h_M$.

2. La signature s du document M est le résultat du chiffrement du condensé h_M par la clé privée RSA d : $s(M) = (h_M)^d \bmod n$.

3. Le document signé $[M, s(M)]$ est transmis au destinataire.

   Ce dernier pour vérifier la signature procède comme suit :

1. Soumit le document reçu $\tilde{M}$ (ce document est potentiellement illégitime) à la même fonction de hachage pour calculer le condensé $\tilde{h}_M = H(\tilde{M})$.

2. Si $s(M)^e = \tilde{h}_M \bmod n$ alors la signature est valide, sinon elle est incorercte.

En effet, $s(M)^e = (h_M)^{ed} \bmod n = (h_M) \bmod n = h_M$ et si le document est authentique alors : $h_M = \tilde{h}_M$.

Ainsi, si une fonction de hachage est utilisée, il n'y a aucun moyen de recopier la signature de quelqu'un sur un document pour l'attacher à un autre, ni d'altérer en quoi que ce soit un document signé. Le plus petit changement dans un document signé provoquerait l'échec de la vérification de la signature.

6.7 Certificats

En évitant le partage d'un secret entre les protagonistes, la cryptographie à clé publique est confrontée à un autre problème extrêmement difficile : comment garantir la validité des clés publiques?.

Si les clés publiques sont stockées dans des annuaires non sécurisés, elles risquent d'être interceptées et remplacées par d'autres clés. Vous pourriez vous limiter à utiliser les clés publiques qui vous ont été remises physiquement, de la main à la main, par leur propriétaire. Mais supposez que vous deviez échanger des informations avec des gens que vous n'avez jamais rencontrés; comment vous assurer que vous en possédez les véritables clés?.

Ce problème peut être résolu en introduisant une tierce partie, appelée autorité de certification, qui valide le lien entre l'identité des utilisateurs et leurs clés publiques.

Définition 6.7.1 *Un certificat est une garantie sur la clé publique (ou en quelque sorte la carte d'identité de la clé publique) offerte par une autorité de certification, pour une période donnée.*

Si une personne A veut certifier sa clé publique, elle envoie sa clé à un organisme de certification, ainsi que différentes informations la concernant (nom, email, etc...). Cet organisme vérifie les informations fournies par A, et à partir de ces dernières l'organisme calcule un résumé en appliquant une fonction de hachage. Puis il signe ce résumé en lui appliquant sa clé secrète, la clé publique ayant été préalablement largement diffusée afin de permettre aux utilisateurs de vérifier la signature avec la clé publique de l'autorité de certification.

Lorsque B désire communiquer avec A, elle télécharge le certificat de celle-ci sur un serveur de certificat (on parle de PKI, Public Key Infrastructure). Elle calcule le résumé du certificat en appliquant une fonction de hachage, puis applique la clé publique de l'organisme auteur du certificat à la signature électronique. Si cette quantité est égale au résumé, elle est sûr qu'elle a bien affaire à A.

Chapitre 7

La cryptographie en pratique

Sommaire

7.1 Introduction

Comme nous l'avons présenté, la cryptographie à clé secrète brille par sa facilité d'implantation et sa rapidité, mais souffre d'une grave lacune qui est le partage de la clé. La cryptographie asymétrique dépasse ce défaut et permet la signature numérique de documents, cependant elle rend le chiffrement extrêmement lent (1000 fois plus lent que les systèmes de chiffrement à clé secrète) et impraticable sauf pour des messages courts.

En pratique, ces deux types de cryptographie s'utilisent le plus souvent conjointement, pour bénéficier de la protection des clés et de la vitesse du chiffrement et du déchiffrement, c'est ce qu'on appelle la cryptographie hybride ou mixte.

7.2 La cryptographie hybride

Les systèmes hybrides procèdent de la manière suivante. Une clé aléatoire est générée pour l'algorithme symétrique (3DES, AES et bien d'autres encore). L'algorithme de chiffrement symétrique est ensuite utilisé pour chiffrer le message. La clé aléatoire quant à elle, se voit chiffrée grâce à la clé publique du destinaire, c'est ici qu'intervient la cryptographie asymétrique, comme la clé est courte, ce chiffrement prend peu de temps. Chiffrer l'ensemble du message avec un algorithme asymétrique serait bien plus lourd, c'est pourquoi on préfère passer par un algorithme symétrique. Il suffit ensuite d'envoyer le message chiffré avec l'algorithme symétrique accompagné de la clé chiffrée correspondante. Le destinataire déchiffre la clé symétrique avec sa clé privée et via un déchiffrement symétrique, retrouve le message.
Un exemple de ce genre d'association est PGP.

7.2.1 PGP (Pretty Good Privacy)

PGP est un cryptosystème inventé par l'informaticien américain Philip Zimmermann, il est utilisé pour chiffrer des messages envoyés par courrier électronique. Philip Zimmermann a travaillé de 1984 à 1991 sur un programme permettant de faire fonctionner le RSA sur des ordinateurs personnels (PGP). Cependant, ceci lui valut divers problèmes

avec la justice, car d'une part, le PGP utilise l'algorithme RSA sans l'accord de ses auteurs. D'autre part, la NSA a tout fait pour tenter d'empêcher la diffusion du PGP. La puissance de ce programme met en effet, à la disposition de chacun un moyen de cacher ses échanges électroniques qui résiste même aux assauts de la plus puissante des agences de renseignements du monde.

7.2.2 Fonctionnement du PGP

Il fonctionne selon le principe suivant :

1. **Compression :** Le texte à envoyer est tout d'abord compressé à l'aide de la fonction de hachage MD5. Cette étape permet de réduire le temps de transmission du message, et améliore également la sécurité. En effet, la compression détruit les modèles du texte (fréquences des lettres, mots répétés).

2. **Chiffrement du message :** Cette opération se fait en deux étapes :

   - PGP crée une clé secrète IDEA de manière aléatoire, et chiffe le texte compressé avec cette clé.
   - PGP chiffre la clé secrète IDEA précédemment créee au moyen de la clé publique RSA du destinataire.

3. **Envoi et réception du message :** l'expéditeur envoie le couple (message chiffré, clé de session chiffrée) au destinataire. Ce dernier récupère d'abord la clé de session, en utilisant sa clé privée, puis il déchiffre le message grâce à la clé de session.

Notons qu'il y a d'autres logiciels aussi performants, mais PGP est le plus connu. Correctement utilisé, il est sûr, même contre les meilleurs cryptanalystes du monde.

7.2.3 IDEA

IDEA a été mis au point en Suisse en 1992. Il est basé sur une clé de 128 bits. Sa structure est semblable à celle du DES. Il utilise une clé de 128 bits, réalise un chiffrement

par blocs de 64 bits, en opérant 8 rondes d'une même fonction qui utilise seulement 3 opérations :

- Ou exclusif,
- Addition modulo 2^{16},
- Multiplication modulo 2^{16}.

Il est très robuste grâce à la largueur de la clé utilisée. Il n'existe pas à l'heure actuelle de machine capable de casser IDEA. Certaines implémentations matérielles permettent une vitesse de chiffrement/déchiffrement considérable. Cet algorithme est connu essentiellement car il constitue la partie "cryptographie à clé secrète" du célèbre PGP.

7.3 Proposition d'un cryptosystème hybride basé sur le problème SAT

7.3.1 Introduction

Dans cette partie nous proposons un algorithme de chiffrement hybride, qui utilise le RSA pour chiffrer une clé secrète, tandis que le message clair est additionné modulo 2 à un vecteur binaire qui est la solution d'un problème SAT.
Avant de décrire cet algorithme, on introduit le problème de satisfaisabilité SAT et ses versions $2-SAT$ et $3-SAT$.

Considérons un ensemble fini X à n variables booléennes (prenant deux valeurs vrai ou faux, notées 1 ou 0 respectivement) $x_1, x_2, \ldots, x_n$. Une formule logique ou booléenne $F(x_1, x_2, \ldots, x_n)$ est une expression logique formée à partir des variables booléennes en utilisant les trois opérateurs : opérateur de négation noté par le symbole $\rceil$ ou barre, opérateur de conjonction noté par $\wedge$ (le et logique) et l'opérateur de disjonction noté $\vee$ (le ou logique).
Un littéral est soit une variable soit la négation d'une variable comme x ou $\bar{x}$. Une clause est la disjonction d'un ou plusieurs littéraux. Une formule logique est dite sous forme

normale conjonctive (FNC) si elle est la conjonction d'une ou plusieurs clauses, comme le montre la formule F ci-dessous :

$$F = (x_1 \vee x_5) \wedge (x_1 \vee \bar{x}_2 \vee x_4) \wedge (\bar{x}_3 \vee x_6) \wedge (x_3 \vee x_4 \vee \bar{x}_5 \vee x_6).$$

Une formule est dite satisfaisable s'il est possible d'affecter une valeur vrai ou faux à chacune des variables de telle façon que la formule ait la valeur vrai. La formule F ci-dessus est satisfaisable car elle prend la valeur 1 pour l'affectation $x_1 = x_2 = 1$, et $x_3 = x_4 = x_5 = x_6 = 0$.

Le problème de satisfaisabilité SAT est un problème de décision qui consiste à savoir si une formule donnée est satisfaisable. Le problème de satisfaisabilité $k - SAT$ consiste à savoir si une formule donnée sous forme normale conjonctive avec k littéraux par clause est satisfaisable.

La formule suivante est un exemple du problème $3 - SAT$:

$F = (x_1 \vee x_3 \vee x_5) \wedge (x_1 \vee x_2 \vee x_4) \wedge (x_3 \vee x_4 \vee x_6)$.

Dans le cas général le problème $k - SAT$ $(k \geq 3)$ est NP-Complet, en particulier il a été prouvé que le problème $2 - SAT$ est polynomial.

Nous présentons dans cette section l'algorithme du à Aspvall, Plass et Tarjan [5], que nous appelons APT (nom condensé de ces concepteurs) résolvant le problème 2-SAT par construction d'un graphe orienté et recherche de chemins dans ce graphe. Pour cela donnant tout d'abord les définitions nécessaires.

7.3.2 Définitions

Graphe et graphe orienté

Un graphe fini G = (V, E) est défini par l'ensemble fini $V = \{v_1, v_2, ..., v_n\}$ $(|V| = n)$ dont les éléments sont appelés sommets, et par l'ensemble fini $E = \{e_1, e_2, ..., e_m\}$ $(|E| = m)$ dont les éléments sont appelés arêtes.

Une arête e de l'ensemble E est définie par une paire non-ordonnée de sommets, appelés les extrémités de e. Si l'arête e relie les sommets v_1 et v_2, on dira que ces sommets sont adjacents. On appelle ordre d'un graphe le nombre de sommets n de ce graphe.

Un graphe orienté est un graphe dont les arêtes sont orientées, on parle alors de l'origine et de l'extrémité d'une arête.

Une boucle est une arête orientée dont l'origine et l'extrémité sont confondues.

Graphe connexe

Un graphe est dit connexe si l'on peut relier deux sommets quelconques du graphe par une suite continue d'arêtes ou d'arcs.

Chemin

Un chemin est une suite ordonnée $(v_1, v_2, ..., v_n)$ de sommets reliés par des arcs. La longueur du chemin est le nombre d'arcs qu'il contient.

7.3.3 L'algorithme BinSat pour résoudre le problème 2-SAT

Nous présentant dans cette section l'algorithme BinSat proposé par Aspvall, Plass et Tarjan en 1979 pour la résolution du 2-SAT. Cet algorithme est basé sur les théoèmes suivants.

Théorème 7.3.1 *Le problème 2-SAT est polynomial.*

Preuve. *On va montrer comment résoudre ce problème efficacement en utilisant la recherche de chemins dans un graphe.* ■

Théorème 7.3.2 (42) *Etant donné un graphe G = (V, E) et deux sommets $s, t \in V$, trouver un chemin de s à t est un problème facile.*

Construction du graphe

On considère le graphe ayant $2n$ sommets, dont n sont étiquetés par les variables, et les n autres par les négations des variables. On représente les m clauses de la formule par $2m$ arcs dans le graphe de la façon suivante : chaque clause de la forme $(x_i \vee x_j)$ peut être vue comme l'implication $\bar{x}_i \Longrightarrow x_j$ ou $\bar{x}_j \Longrightarrow x_i$, donc on ajoute pour chaque clause ainsi notée les arcs $(\bar{x}_i, x_j)$ et $(\bar{x}_j, x_i)$.

Exemple 7.3.1 *Au problème :* $(\bar{x}_1 \vee x_2) \wedge (\bar{x}_2 \vee x_3) \wedge (x_1 \vee \bar{x}_3) \wedge (x_3 \vee x_2)$*, on associe le graphe suivant :*

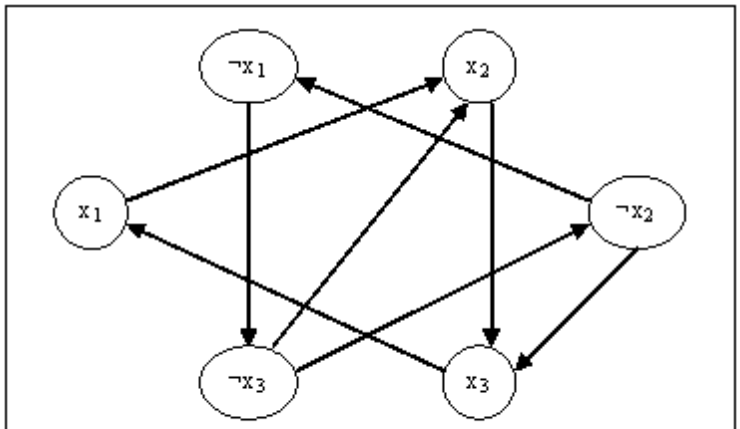


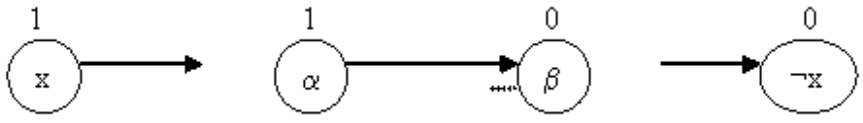
Figure 7.1 : Le graphe associé au 2-SAT

Lemme 7.3.1 *Une 2-FNC est non satisfaisable si et seulement si il existe une variable x telle que :*

1. *il existe un chemin de x à $\bar{x}$ dans le graphe,*
2. *il existe un chemin de $\bar{x}$ à x dans le graphe.*

Preuve. Supposons qu'il existe des chemins, de x à $\bar{x}$ et de $\bar{x}$ à x et qu'il existe aussi une affectation satisfaisant F.

Si x est valué à 1 (similaire pour x valué à 0), on a :



alors $(\bar{\alpha} \vee \beta) = 0$, contradiction. ■

Il s'agit alors de tester pour toute variable x_i l'existence d'un chemin de x_i à $\bar{x}_i$, si c'est le cas pour une variable, cela signifie que la formule n'est pas satisfaisable. Si aucun chemin de ce type n'est trouvé, elle est satisfaisable, et pour trouver une valuation solution, tant que tous les sommets du graphe ne sont pas valués, on effectue les étapes suivantes :

- valuer un des sommets non encore valués à 1,
- propager cette valeur par connexité dans le graphe,
- pour tous les sommets ainsi valués, valuer à 0 leur négation.

Exemple 7.3.2 *Résolvons le 2-SAT précédent. Comme il n'existe aucun chemin de* x_i *à* $\bar{x}_i$*, la formule est satisfaisable. On affecte* 1 *à* x_1*, qui est propagée à* x_2 *puis à* x_3*, ce qui nous permet d'affecter* 0 *à* $\bar{x}_1$*,* $\bar{x}_2$ *et* $\bar{x}_3$*.*

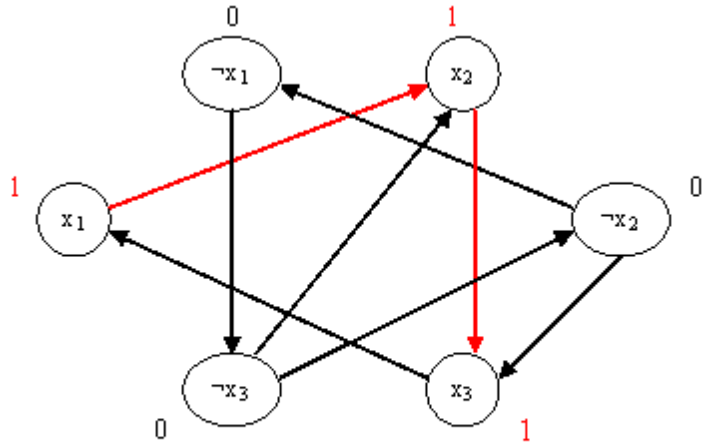


Figure 7.2 : Solution du 2-SAT

7.3.4 Description de la méthode

Décrivons maintenant notre cryptosystème en 3 étapes.

Génération des clés

Au départ on gènère un 2-SAT aléatoire comme suit :

- On se donne l variables booléennes $x_1, x_2, ..., x_l$.
- On tire m couples de variables au sort.
- On ajoute des "$\vee$" entre les variables.

- On nie chaque variable avec une probabilité 1/2.
- On ajoute des "$\wedge$" entre toutes les clauses.

On résout le 2-SAT ainsi obtenu avec l'algorithme BinSat présenté ci-dessus. Soit $S = (s_1, s_2, ..., s_l)$ sa solution, celle-ci va constituer la clé secrète.

Après avoir créé le $2 - SAT$, on va le camoufler sous un problème $3 - SAT$ qui est difficile, ce dernier doit avoir la même solution S. C'est la raison pour laquelle on ajoute à chaque clause du $2 - SAT$ initial une variable nulle pour ne pas changer sa valeur de vérité.

L'algorithme $C-3SAT$ suivant permet de faire cette transformation en $O(m)$ opérations :

Algorithme C-3SAT;

$p := 1$

Pour $k := 1$ **haut** m **faire** (*Traitement de la $k^{ème}$ clause $C_k = x_i \vee x_j$*)

$trouve := faux$ (*trouve indique si une variable est ajoutée à la $k^{ème}$ clause*)

Tant que non$(trouve)$ **faire**

Si $p < i$ **alors**

- On ajoute à C_k la variable d'indice p à la première position
- Si $s_p = 0$ on ajoute x_p à C_k sinon on ajoute $\bar{x}_p$ à C_k
- V[k]:=1 (*V[k] est la position de la variable ajoutée à C_k*)
- $Trouve := vrai$

Si non

Si $(p < j)$ **et** $(p \neq i)$ **alors**

- On ajoute à C_k la variable d'indice p à la deuxième position
- Si $s_p = 0$ on ajoute x_p à C_k sinon on ajoute $\bar{x}_p$ à C_k
- V[k]:=2
- $Trouve := vrai$

Si non

- On ajoute à C_k la variable d'indice p à la troisième position
- Si $s_p = 0$ on ajoute x_p à C_k sinon on ajoute $\bar{x}_p$ à C_k
- V[k]:=3;
- $Trouve := vrai$

Fin Si

Fin Si

Si $Trouve = faux$ **alors** $p := p + 1$ (*C'est le cas où $p = i$ ou $p = j$*)

Fin Tant que

Si $p < l$ **alors** $p := p + 1$ Si non $p := 1$

Fin Pour

Le problème $3-SAT$ étant obtenu, il est ensuite publié (il représente la clé publique). Le vecteur $V = (v_1, v_2, ..., v_m)$ où $v_i \in \{1, 2, 3\}$ représente la position de la variable ajoutée

à la $i^{ème}$ clause.

On pose $a = v_1v_2...v_m$, l'entier qui regroupe les v_i l'un à côté de l'autre et après avoir calculé $\acute{a} = a^e \bmod n$, où (n, e) est la clé publique RSA du destinataire, les vecteurs V et a sont détruits tandis que $\acute{a}$ est transmis avec le message chiffré.

Le chiffrement

Etant donné un message binaire $M = m_1m_2...m_l$ (si le message a une longueur supérieur à l , il sera décomposé en blocs de taille l) et une clé secrète S, le message chiffré $C = c_1c_2...c_l$ est obtenu en additionnant modulo 2 M et S :

$$c_i = \begin{cases} 0 & \text{si } m_i = s_i \\ 1 & \text{sinon} \end{cases}$$

Ainsi on envoie $(C, \acute{a})$.

Le déchiffrement

Le destinataire calcule a à l'aide de sa clé privée d : $a = \acute{a}^d \bmod n$, déduit le vecteur V et supprime les variables ajoutées du $3-SAT$, il obtient alors le $2-SAT$ correspondant et par la suite sa solution $S = (s_1, s_2, ..., s_l)$ en utilisant l'algorithme BinSat. En fin il déduit le message initial par : $m_i = 0$ si $c_i = s_i$ et 1 sinon.

Exemple 7.3.3 ***Génération des clés :***
Soit le 2-SAT :

$$F = (\bar{x}_1 \vee x_2) \wedge (x_3 \vee x_4) \wedge (\bar{x}_1 \vee x_5) \wedge (x_2 \vee x_3) \wedge (\bar{x}_5 \vee x_6) \wedge (x_4 \vee x_6).$$

L'application de l'algorithme BinSat donne la solution $S = (1, 1, 1, 1, 1, 1)$.

Maitenant on déroule l'algorithme $C-3SAT$, ce qui donne les nouvelles clauses :

$$\begin{cases} C_1 = (\bar{x}_1 \vee x_2 \vee \bar{\mathbf{x}}_3) \\ C_2 = (\bar{\mathbf{x}}_1 \vee x_3 \vee x_4) \\ C_3 = (\bar{x}_1 \vee \bar{\mathbf{x}}_2 \vee x_5) \\ C_4 = (\bar{\mathbf{x}}_1 \vee x_2 \vee x_3) \\ C_5 = (\bar{\mathbf{x}}_1 \vee \bar{x}_5 \vee x_6) \\ C_6 = (\bar{\mathbf{x}}_1 \vee x_4 \vee x_6) \end{cases}$$

Ainsi le vecteur $V = (3, 1, 2, 1, 1, 1)$ et $a = 312111$.

Chiffrement :

Soit la clé publique $(n, e) = (36581, 5)$. Pour chiffrer a à l'aide du RSA on le décompose en deux blocs $a_1 = 3121$ et $a_2 = 11$, ainsi $\acute{a}_1 = a_1^e \bmod n = (3121)^5 \bmod 36581 = 26481$ et $\acute{a}_2 = a_2^e \bmod n = (11)^5 \bmod 36581 = 14727$ donc $\acute{a} = 2648114727$.

On publie alors le $3-SAT$, á et on détruit V et a. Si l'on désire envoyer le message $M = 011000110110$, on le décompose en deux blocs de six bits chacun soient $m_1 = 011000$ et $m_2 = 110110$ ensuite on calcule $C = 100111001001$ et on envoie : $(C, \acute{a}) = (100111001001, 2648114727)$.

Déchiffrement :

Le destinataire après avoir reçu $(C, \acute{a})$ calcule $a_1 = \acute{a}_1^d \bmod n = (26481)^{14477} \bmod 36581 = 3121$ et $a_1 = \acute{a}_1^d \bmod n = (14727)^{14477} \bmod 36581 = 11$ par conséquent il a $a = 312111$. Il supprime la $3^{ème}$ variable de la première clause du $3-SAT$, la $1^{ère}$ variable de la deuxième, et ainsi de suite, il obtient le $2-SAT$ correspondant donc sa solution $S = (1, 1, 1, 1, 1, 1)$. Il calcule le message M de la façon suivante : si $c_i = s_i$ alors $m_i = 1$ si non $m_i = 0$ sachant que S est suffisamment répétée, il obtient $M = 011000110110$.

Remarque 7.3.1 *On remarque que la solution S du 2-SAT peut ne pas être unique, pour y remidier on propose aux interlocuteurs de valuer à chaque fois la variable du plus petit indice non encore valuée.*

Chapitre 8

Présentation du logiciel *"RLBS-CRYPTTOOL"* réalisé pour le chiffrement

Sommaire

8.1 Introduction

Dans le but de faire savoir ce qu'est le chiffrement et ce qu'il peut apporter dans notre vie de tous les jours, nous avons réalisé le logiciel RLBS-CRYPTTOOL (Rezkallah L, Bouroubi S Crypt Tool) qui est un outil de chiffrement développé avec Delphi 5, il regroupe les méthodes de chiffrement classiques et modernes les plus connues présentées dans ce mémoire, ce logiciel permet la génération des clés, le chiffrement et le déchiffrement des messages au format texte ainsi que leur sauvegarde, il intègre aussi un simulateur ENIGMA.

Une partie du travail réalisé a consisté en l'implémentation d'un type numérique permettant de travailler sur de trés grands entiers, il a fallu implémenter les fonctions arithmétiques (Addition, Soustraction, Multiplication, Division, PGCD, Inversion modulaire et Exponentiation modulaire) sur ce type de manière efficace.

Ce chapitre est alors consacré à la présentation de l'interface du logiciel et à l'explication qui facilite son utilisation.

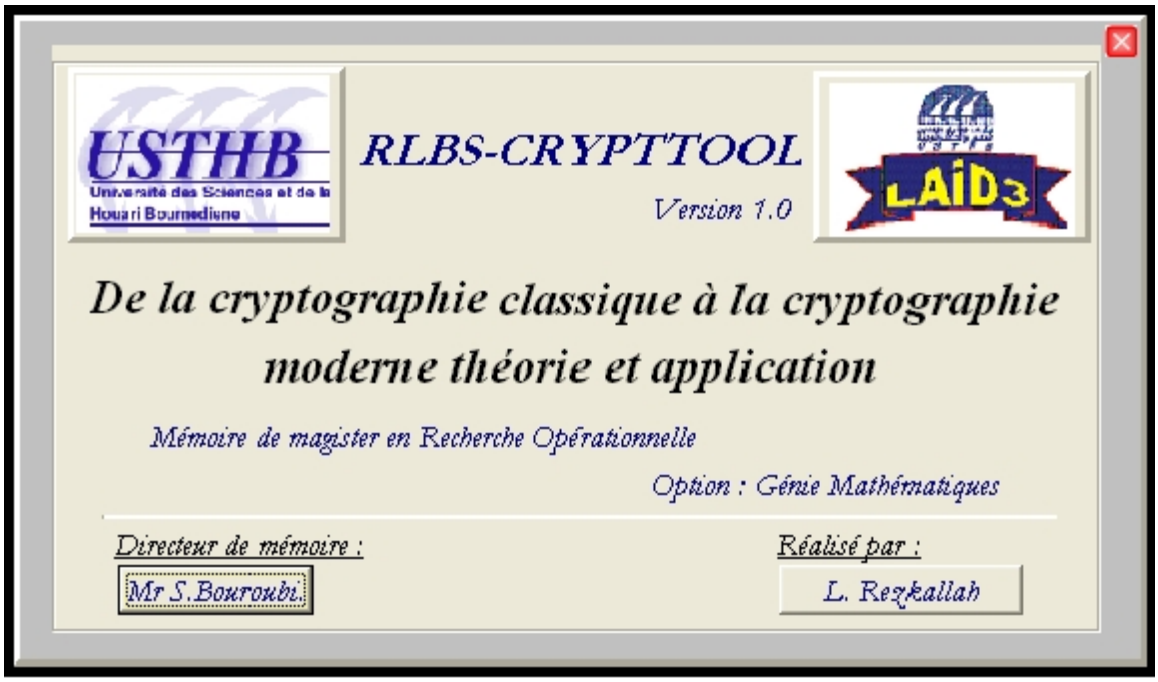


Figure 8.1 : La fenêtre A propos de RLBS-CRYPTTOOL

8.2 Fenêtre principale

*Figure 8.2 :*Fenêtre principale RLBS-CRYPTTOOL

La fenêtre principale de RLBS-CRYPTTOOL contient les menus qui permettent d'utiliser les méthodes de chiffrement classiques et modernes via les menus, par esemple :

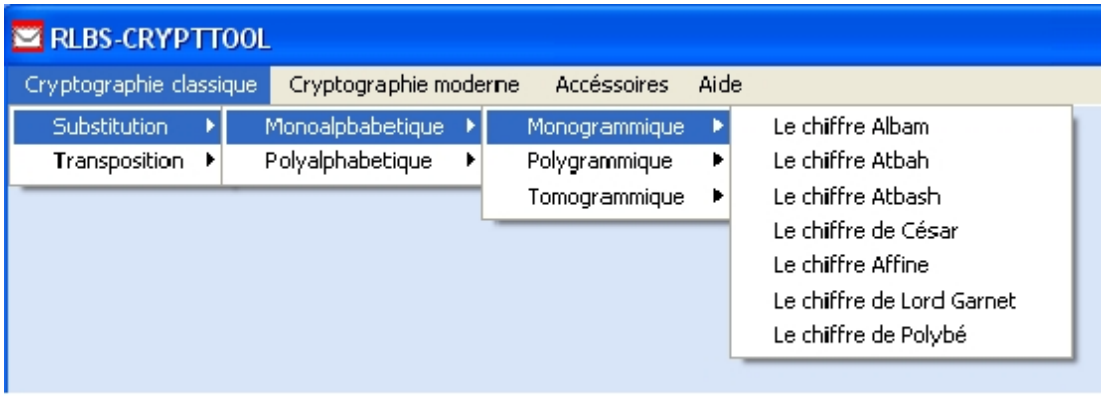


*Figure 8.3 :*Le menu cryptographie classique

et

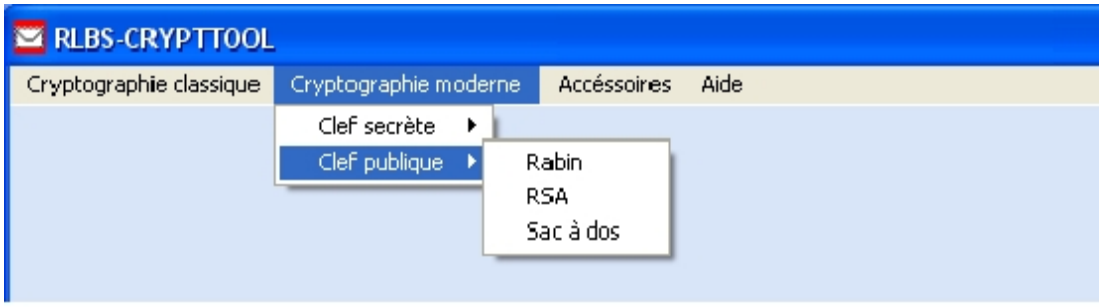


Figure 8.4 : Le menu cryptographie moderne

Le menu Accessoires quant à lui permet d'envoyer des messages chiffrés avec, si vous le souhaitez, un fichier attaché et d'accéder à un outil de calcul sur les grands nombres entiers, voici les fiches correspondantes :

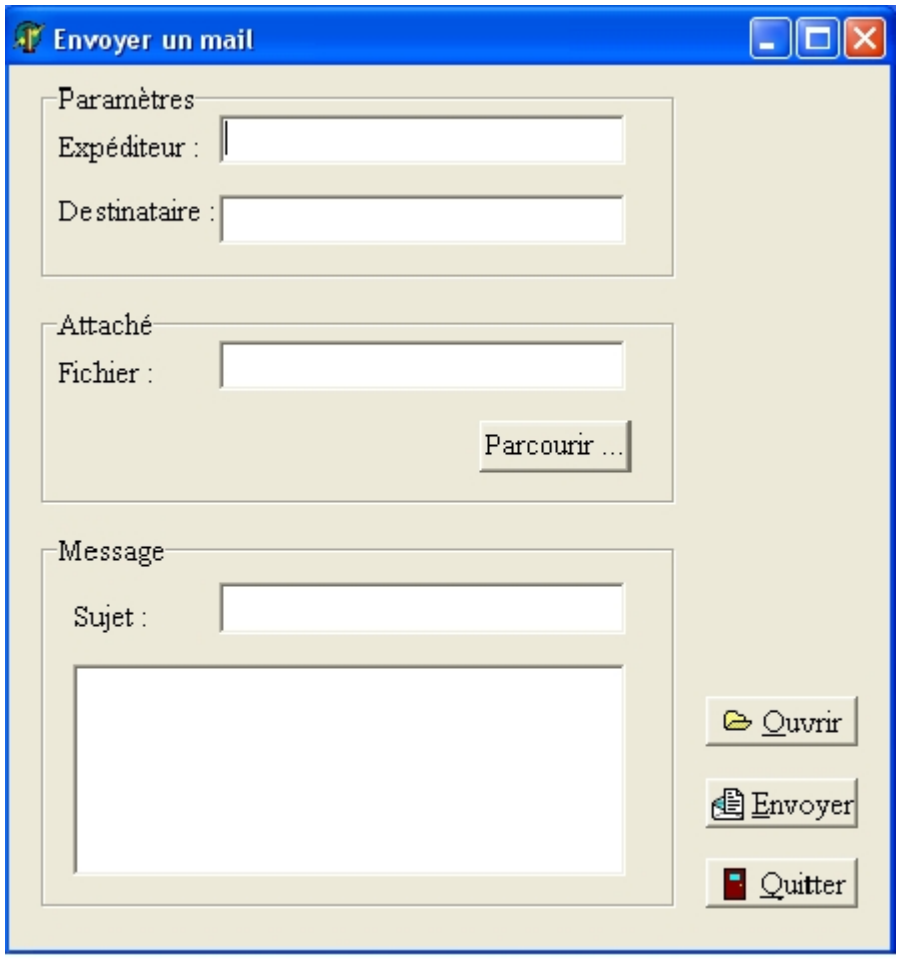


Figure 8.5 : Envoyer un message chiffré

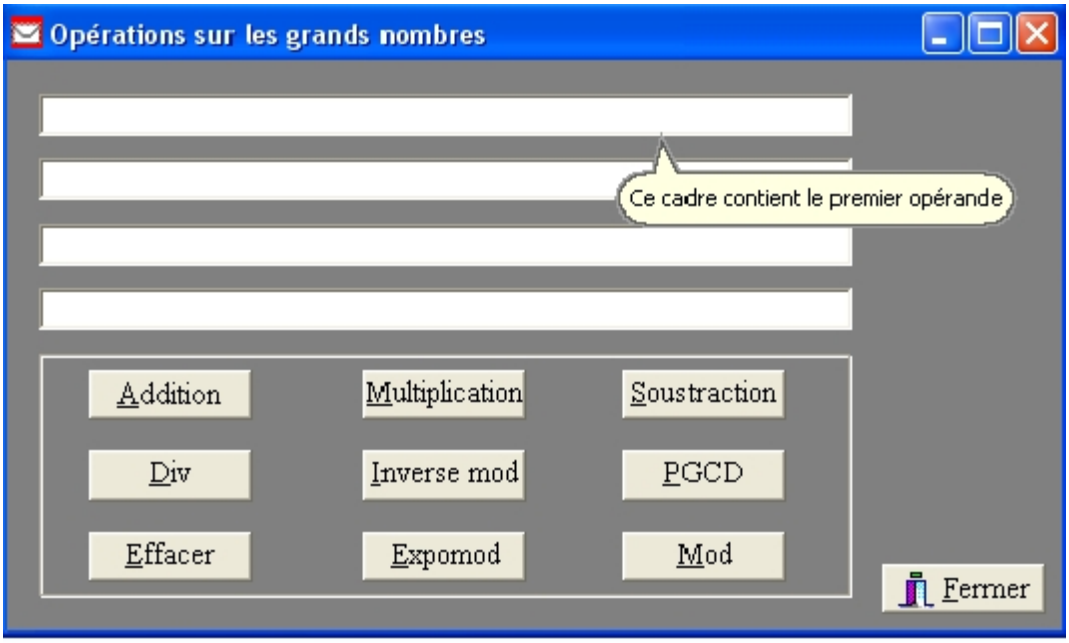


Figure 8.6 : La fiche opérations sur les grands nombres

Le dernier menu, Aide, contient deux sous menus. "A propos" qui donne des informations sur le logiciel et Terminologie qui comme son nom l'indique contient une "Terminologie" qui contient une terminologie sur la cryptographie.

Notons que les fenêtres de ce logiciel s'utilisent de la même manière pour le chiffrement et le déchiffrement des messages, présentant par exemple la fiche de chiffrement DES.

8.3 La fenêtre DES

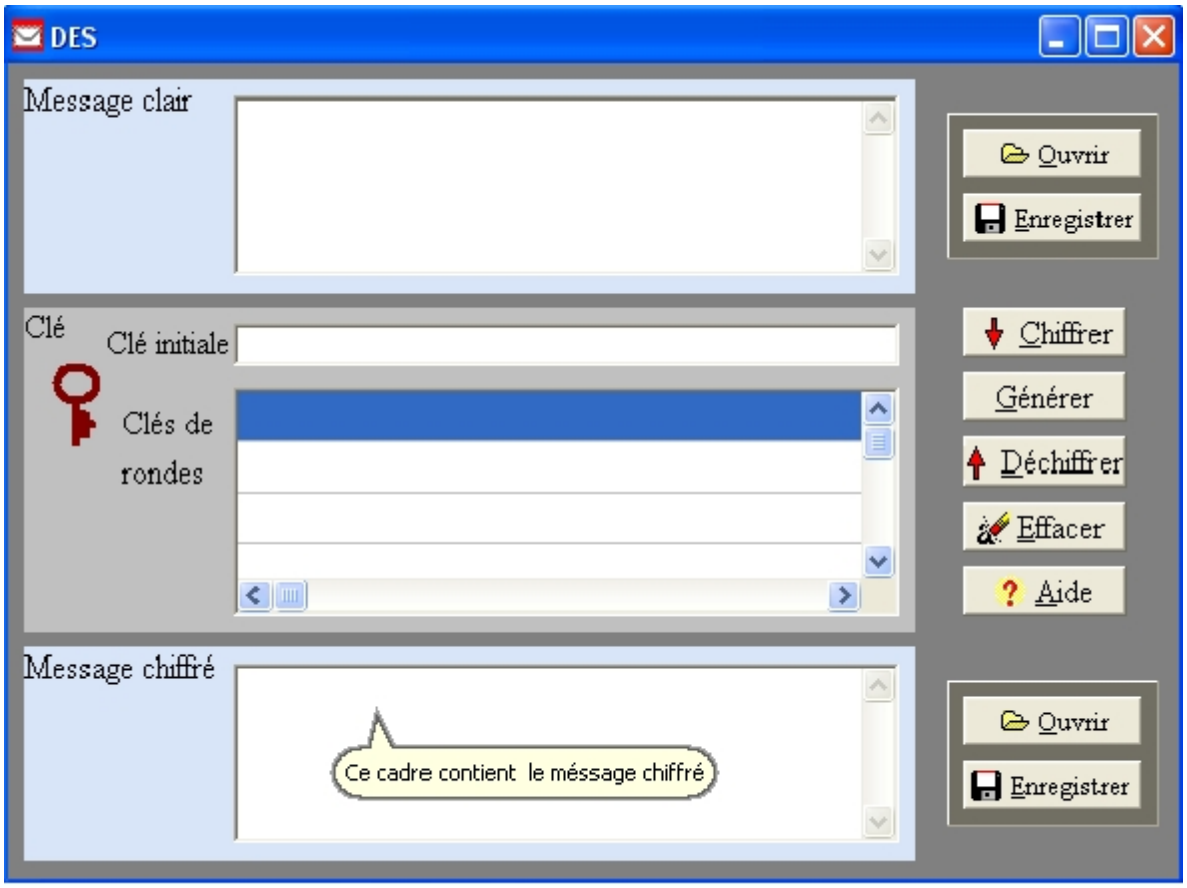


Figure 8.7 : La fenêtre DES

Cette fenêtre est réservée au chiffrement et au déchiffrement, mais avant d'effectuer une de ces opérations, vous devez être en possession d'une clé. Détaillons maintenant le rôle de chaque composant de la fiche DES :

- Le cadre message clair est destiné à contenir le texte à chiffrer ou le texte déchiffré. Vous pouvez soit saisir directement le texte dans le champ prévu à cet effet, soit choisir un fichier sur le disque en utilisant le bouton Ouvrir situé en face. Ce dernier permet d'ouvrir un fichier sauvegardé au format texte sur le disque dur de votre ordinateur ou sur un support externe.

  Le bouton Enregistrer situé en dessous du bouton Ouvrir vous permet de sauvegarder le contenu du champ Message clair dans un fichier texte.

- Cliquer sur le bouton Générer pour générer aléatoirement une clé initiale à 64 bits et 16 clés de rondes à 48 bits chacune.

- Après avoir choisi la clé de chiffrement, cliquez sur le bouton Chiffrer pour lancer l'opération de chiffrement du texte situé dans le champ Message clair. Après un court instant, le texte chiffré s'affiche dans le cadre Message chiffré. Ce bouton affiche un message d'erreur si aucune clé n'est chargée.

- Le bouton Déchiffrer lance l'opération de déchiffrement (après avoir saisi les clés de chiffrement dans les champs correspondants) du texte situé dans le cadre Message chiffré. Aprés un moment, le message clair s'affiche dans le cadre Message clair. Ce boutonaffiche un message d'erreur si aucune clé n'est chargée.

- Les boutons Ouvrir et Enregistrer, cités en face du cadre Message chiffré, permettent respectivement d'ouvrir un message texte à déchiffrer et d'enregistrer un message chiffré.

- N'hésitez pas à consulter l'aide accessible via le bouton Aide.

Conclusion générale

Dans ce mémoire, nous nous somme intéressés à la cryptographie classique et moderne, sa théorie et ses applications.

En premier lieu, nous avons présenté les méthodes de cryptographie les plus connues et les principaux objectifs qu'elle permet d'atteindre, ainsi que les notions de la théorie de la complexité nécessaires.

En second lieu nous avons fait le lien entre la théorie de la complexité et la cryptographie en proposant deux cryptosystèmes, le premier est symétrique et est basé sur la difficulté de la factorisation des grands nombres entiers, le second est hybride et est fondé sur la difficulté de la résolution du problème SAT.

Enfin, en dernier lieu, la réalisation du logiciel *RLBS-CRYPTTOOL* a permis une utilisation simple et rapide des méthodes de chiffrement présentées, et ce, via une interface interactive. Néanmoins pour une extension future du programme, il serait souhaitable de prendre en considération quelques recommandations :

- L'interface graphique pourrait être améliorée.

- Changer le langage de programmation, par exemple choisir JAVA pour une utilisation plus large (voir internet).

Le travail présenté dans ce rapport de mémoire ouvre plusieurs perspectives :

1) Construction de cryptosystèmes à clé publique ou mixtes basés sur des problèmes NP-Complets. Dans le même cadre, nous envisageons un cryptosystème basé sur le problème du voyageur de commerce. Et ce en considérant un graphe G qui soit un cycle hamiltonien, qu'on cache dans un autre graphe G' en ajoutant des arêtes à l'aide d'une fonction à sens unique.

2) Etude des attaques possibles contre les méthodes de chiffrement moderne.

3) Etudier la complexité des différents algorithmes de chiffrement exposés.

4) Etude de la sensibilité du disque tournant à l'attaque du mot probable et aux autres attaques connues.

Bibliographie

[1] A. CANTEAUT, Analyse et conception de chiffrements à clef secrète, HDR (Habilitation à Diriger des Recherches), Université Paris 6, http://www-rocq.inria.fr/Codes/Anne.canteaut/canteaut-hdr.pdf

[2] A. CANTEAUT, F. LEVY-DIT-VEHEL, "La cryptologie moderne", INRIA Ecole Nationale Supérieure, http://www-rocq.inria.fr/~canteaut/

[3] A. DEL VAL, "On 2-SAT and Renamable Horn E.T.S", Université de Madrid, 2000

[4] A.J. Menezes, P.C. Van Oorschot et S.A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1997

[5] B. ASPVALL, M.F. PLASS et R.E. TARJAN, "A linear time algorithm for testing the truth of certain quantified boolean formulas", Information Processing Letters, 1979, http://geodisi.u-strasbg.fr/~daurat/sujets/master1/tarjan.pdf

[6] B. MAZUR, "Pourquoi les nombres premiers ?", Les dossiers de la recherche, N° 20 Août 2005

[7] B. SCHNEIER, Cryptographie appliquée, International Thomson Publishing, Paris, 1997

[8] C. PAPADIMITRIOU, Computational Complexity, Addison-Wesley, 1994

[9] C. ROUSSEAU, "Les dessous de la cryptographie à clé publique", Université de Montréal, Octobre 2005

[10] C.E. SHANNON, "Communication Theory of Secrecy Systems", Bell System Technical Journal, Vol 28, pp. 656-715, Octobre 1949

[11] D. KAHN, The Codebreakers: the story of secret writing, MacMillan publishing, 1996

[12] D. STEHLE, "RSA, Cryptosystèmes sacs à dos et Réseaux Euclidiens", Istanbul, Avril 2005, http://www.loria.fr/~stehle/

[13] E. BIHAM., A. SHAMIR, Differential cryptanalysis of Data Encryption Standard, Springer-Verlag, 1993

[14] E. GIOAN, Introduction à la cryptographie, maîtrise d'informatique, Université de La Réunion, 2004

[15] F. ARNAULT, Théorie des nombre et cryptographie, Cours de D.E.A, Université de Limoges, Mai 2002

[16] F. GAINES HELEN, Cryptanalysis, a study of ciphers and their solution, Dover Publications Inc, 1956

[17] F. GUILLEUX, Sécurité informatique, cryptographie, certificats et signature électronique, CRU, octobre 2004, www.cru.fr

[18] F. LEPREVOST, Espionnage industriel, cryptographie et relations internationales, Université du Luxembourg, Laboratoire d'Algorithmique, Paris, mars 2006

[19] F. LEPREVOST, S. VARRETTE, Introduction à la cryptographie à clef publique, Université du Luxembourg, CESI-LACS, Luxembourg Laboratoire ID-IMAG, Grenoble, France 2005

[20] F. RAYNAL, Codage Etude d'outils pour la dissimulation d'information, approches fractales, protocoles d'évaluation et protocoles cryptographiques, Thèse, Mars 2002

[21] G. DUBERTRET, Initiation à la cryptographie, Vuibert Informatique, Paris, 2000

[22] G. ZEMOR, Cours de cryptographie, Cambridge University Press, Novembre 2000

[23] H. FEISTEL, Block cipher cryptographic system, U.S. patent 3,798,359, 19 mars 1974

[24] H. FEISTEL, Cryptography and computer privacy, Scientific American, 228, mai 1973

[25] J. BUCHMANN, "La factorisation des grands nombres", Pour la science N° 251, Septembre 1998, http://www.pourlascience.com/numeros/pls-251/art-12.htm

[26] J. DAEMEN et V. RIJMEN, The Design of Rijndael : AES - The Advanced Encryption Standard, Springer-Verlag, 2002

[27] J. STERN, L. Granboulan, P. Nguyen et D. Pointcheval, Conception et preuves d'algorithmes cryptographiques, Cours de magistère M.M.F.A.I. Ecole normale supérieure, 2004

[28] J.Y. ENJALBERT, "Cryptographie utilisation par l'administration électronique", Université Lille 2

[29] L. ALBERT, Cours et Exercices d'Informatique, Vuibert 1998

[30] L. MONIER, "Evaluation and comparison of two efficient primality testing algorithms", Therical Computer Science, Vol 11, 1980

[31] J.P. GAULIER, Analyse des algorithmes finalistes concourant pour le futur standard AES, Mémoire Ingénierie et Intégration Informatique Systèmes d'Information

[32] M. AGRAWAL, N. Kayal et N. Saxena, "PRIMES is in P", Annals of Mathematics 160 (2004), 781–793

[33] M. DIDIER, Les codes secrets décryptés, City éditions, 2007

[34] M. FINIASZ, Nouvelles constructions utilisant des codes correcteurs d'erreurs en cryptographie à clef publique, Thèse de Doctorat, INRIA, octobre 2004

[35] N. KOBLITZ, A Course in Number Theory and Cryptography, Second Edition, Springer-Verlag, May 1994

[36] M.O. RABIN, Probabilistic algorithms for testing primality, Journal of numbers theory, 12 (1980), 128-138

[37] N. GHOUALMI-ZINE, "La Cryptographie de l'informatique à la physique ?", Département Informatique Faculté des sciences de l'ingénieur Université Badji Mokhtar, Annaba Algérie

[38] N. JOMBART, "Comment générer des nombres aléatoires avec un ordinateur", Confidentiel Sécurité n°66, février 2000

[39] O. GOLDREICH, Modern Cryptography, Probabilistic Proofs and Pseudorandomness, Springer-Verlag, 1999

[40] P.A. FOUQUE, Primalité, Factorisation et Logarithme Discret, école normale supérieure, Département d'Informatique, MPRI M1, 2006

[41] P. GUILLOT, "Introduction à la cryptographie", Université Paris 8 – Vincennes – Saint-Denis, 2 rue de la Liberté, 93526 Saint-Denis CEDEX, France, http ://ufr6.univ-paris8.fr/lit-math/maths/

[42] P. LACOMME, Algorithmes de graphes, Eyrolles, 2^e édition 2003

[43] P. ZIMERMMANN, "Cryptographie et réseau", Pour la science N° 260, Juin 1999

[44] P. ROUCHON, MATHEMATIQUES DISCRETES, Ecole Nationale Supérieure des Mines de Paris, Novembre 2003

[45] R.C. MERKLE, M.E. Hellman, Hiding information and signatures in trapdoor knapsacks. IEEE transactions on information theory, 24, 1978

[46] R.L. RIVEST, A. SHAMIR, et L.M. ADLEMAN, A method for obtaining digital signatures and public-key cryptosystems, Communications of the ACM, 21, 1978

[47] R. SHAMIR, A polynomial time algorithm for breaking the merkle-hellman cryptosystem. IEEE transactions on information theory, 30, 1984

[48] S. ABRAHAM, Elementary cryptanalysis, published by The American Association of America, 1966

[49] S. AL FAKIR, Algèbre et théorie des nombres, Editions Ellipses, 2003

[50] S. DOUGLAS, Cryptographie - Théorie et pratique, Vuibert Informatique, Paris, 2001

[51] S. BURCKEL, "A recursive algorithm for SAT", Laboratoire de Logique, Algorithmique et Informatique de Clermont Université d'Auvergne, BP 86 63172 Aubière

[52] S. HERVET, La cryptographie Appliquée au Web, DEA Informatique, Université de Picardie Jules Verne, France, 1998

[53] S. SINGH, Histoire des codes secrets, éditions Jean-Claude Lattès, 1999

[54] S. VARRETTE, Fonctions de Hachage et Signatures Electroniques, Cours "Cryptographie & Securité Réseau" Master Info, Université de Yaoundé, http://www-id.imag.fr/~svarrett/

[55] T. GENE, Message Authentication with One-Way Hash Functions, ACM Computer Communication Review, Vol. 22, 1992

[56] V. CORTIER, Vérification automatique des protocoles cryptographiques, Thèse pour obtenir le grade de Docteur de l'école Normale Supérieure de Cachan, mars 2003

[57] W. DIFFIE, M.E. HELLMAN, "New directions in cryptography", IEEE Transactions on Information Theory, 22, 1976

[58] W.R. ALFORD, A. Granville et C. Pomerance, "There are infinitely many Carmichael numbers", Annals of Mathematics 140, 1994

Sites internet

www.techno-science.net

http://www.enseignement.polytechnique.fr/profs/informatique/Guillaume.Poupard/PI

http://www.cryptologie.com/

http://csrc.nist.gov/encryption/aes/rijndael/

http://www.esat.kuleuven.ac.be/ rijmen/rijndael/

http://www.Securiteinfo.com

http://www.science.ch

http://www.cs.mcgill.ca/~jford/crepeau/CRYPTO/Biblio QC.html.

http://www.lsp.ups-tlse.fr/Chafai/

http://www.france-science.org

http://diabo.free.fr/enigma/

http://www.authsecu.com

http://www.umich.edu/~umich/fm-34-40-2/ Traité de cryptanalyse

http://www.ebabylone.com/encyclopedie_Histoire_de_la_cryptanalyse.html

http://www.ebabylone.com/cryptographie.html

http://home.us.net/~encore/Enigma/enigma.html

http://www.alanturing.net

http://mantis.free.fr/articles

http://www.merkle.com

http://openpgp.vie-privee.org/

http://www.apprendre-en-ligne.net

http://www.security-labas.org

http://www.bibmath.net

http://www.uqtr.ca

http://www.securite.org

http://www.securiteinfo.com

http://fr.wikipedia.org

http://www.commentcamarche.net

http://www.ssi.gouv

http://mbourgeois.developpez.com