الجمهوريـة الجزائـريـة الديمقراطيـة الشعبيـة

**République Algérienne Démocratique et Populaire**

وزارة التعليـم العالـي والبحـث العلمـي

**Ministère de l'Enseignement Supérieur et de la Recherche Scientifique**

**Nº Réf :……**

# Centre Universitaire de Mila

**Institut des Sciences et de la Technologie** **Département Mathématiques et Informatique**

**Mémoire préparé En vue de l'obtention du diplôme de licence**

**En Filière : Informatique générale**

# Thème :

# Implémentation de l'algorithme RIJNDAEL pour la sécurité des données


**Préparé par :**

*1*-Lahkiri Abderraouf

*2*-Djamaa Houssam Eddin

*3*- Seraoui Mouhammed Wadjih

**Encadré par :**

Mme Deffas Zineb M.A.A


**Année universitaire : 2013/2014**

# *Remerciements*

*C'est avec l'aide de Dieu qu'a vu les jours ce présent travail.*

***A notre encadreur Mme Deffas Zineb***

*Nous avons eu le privilège de travailler parmi votre équipe et d'apprécier vos qualités et vos valeurs.*

*Votre sérieux, votre compétence et votre sens du devoir nous ont énormément marqués.*

*Veuillez trouver ici l'expression de notre respectueuse considération et notre profonde admiration pour toutes vos qualités scientifiques et humaines.*

*Ce travail est pour nous l'occasion de vous témoigner notre profonde gratitude.*

***Wadjih ,Houssam et Abderraouf***

# *Dédicaces*

## *Je dédie cette travaille à …*✍

- ✓ *A mon père.*
- ✓ *A ma mère.*
- ✓ *A mon frère ayoube et ma sœur aya.*
- ✓ *A mon trinôme Wadjih et Houssam.*
- ✓ *Aux enseignants du cycle d'étude au centre universitaire de Mila.*
- ✓ *Aux mes cousins Adam et Haytem et Hatem et Tayma, Diyaa et Islâm et Dyna,Nourhane et Mohammed.*
- ✓ *A toute mes amies d'étude.*
- ✓ *Aux étudiants qui me donner l'aide pour finir ce travaille Amel Meriem et Nassim avec mes remerciements.*

***ABDERRAOUF***

## *Je dédie cette travaille à …✍*

- ✓ *A mon père.*
- ✓ *A ma mère.*
- ✓ *A toute ma famille.*
- ✓ *A mon trinôme Abderraouf et Houssam.*
- ✓ *Aux enseignants du cycle d'étude au centre universitaire de Mila.*
- ✓ *A toute mes amies d'étude.*

***M.WADJIH***

## *Je dédie cette travaille à …✍*

- ✓ *A mon père.*
- ✓ *A ma mère.*
- ✓ *A mes souers.*
- ✓ *A mon trinôme Abderraouf et Wadjih.*
- ✓ *A mes amis Oussama et Amir.*
- ✓ *Aux enseignants du cycle d'étude.*
- ✓ *A toute mes amies d'étude.*

***HOUSSAM. Eddine***

# *Résumé*

Depuis l'Antiquité, les hommes expriment le besoin de transmettre des informations de manière sécurisée en les rendant inintelligibles pour toute personne étrangère à l'échange. Cette nécessité s'accentue encore plus dans un contexte militaire et en temps de guerre. L'ensemble de ces techniques de sécurité sont regroupées dans un domaine particulier : la cryptographie.

Avec la popularité grandissante des réseaux, des échanges de données, et donc des transmissions entre individus, Le cryptage et le décryptage sont devenus de plus en plus indispensables comme moyen de confidentialités et de protection de l'information contre tout système d'espionnage. Au travers de ce travaille, nous essaierons de vous résumer l'étude des différentes classes d'algorithmes cryptographiques ainsi que la réalisation d'un système cryptographique à clé secrète Rijndael, qui fait partie des systèmes symétriques (la même clé est utilisé pour crypter et décrypter) et qui exploite des méthodes cryptographiques traditionnelles.

# Table des matières

# Liste des figures

# Liste des tableaux

# Introduction générale

Dès que les hommes apprirent à communiquer, ils durent trouver des moyens d'assurer la confidentialité d'une partie de leurs communications : l'origine de la cryptographie remonte sans doute aux origines de l'homme.

Dans notre monde moderne, où diverses méthodes de communication sont utilisées régulièrement, le besoin de confidentialité est plus présent que jamais à une multitude de niveaux. Par exemple, il est normal qu'une firme désire protéger ses nouveaux logiciels contre la piraterie, que les institutions bancaires veuillent s'assurer que les transactions sont sécuritaires et que tous les individus souhaitent que l'on protège leurs données personnelles.

Le besoin de communications sécuritaires a donné naissance à la science que nous appelons Cryptographie.

En effet, la cryptographie est un terme générique désignant l'ensemble des techniques permettant de chiffrer des messages c'est-à-dire de les rendre inintelligible dans le but de masquer leur contenue, de les rendre incompréhensible aux personnes non autorisées et d'empêcher leur modification ou leur utilisation illégale, autrement dit garantir la confidentialité, l'intégrité ainsi que leur imputabilité, elle ne protège pas les communications en tant que telles mais plutôt leur contenu.

Plusieurs algorithmes ont été proposé pour le cryptage, ces algorithmes sont classifié selon différents critères, par exemple en peut les classifier selon l'apparition (classiques et modernes), ou selon la nature de la clé (publique ou secrète).

Puisque cette science est très large, nous essayons d'implémenté un algorithme moderne à clé secrète RIJINDAEL.

Et pour réaliser ce but, dans ce qui suit Nous nous tournerons dans un premier temps vers les techniques cryptographiques qui ont marqué l'histoire, suivis par les techniques actuelles du monde de l'informatique.

Par ailleurs, explique en détaille l'algorithme que nous avons choisi (l'algorithme AES nommée RIJNDAEL).

Enfin la description du langage et l'environnement de développement utiliser pour l'implémentation Puis présente l'interface graphique de notre application suivie par des exemples.

# Chapitre 1 : Etat de l'art sur la cryptographie

# 1. Introduction

Dans ce chapitre, nous présentons un aperçu global sur la notion de la cryptographie.

Nous commençons par l'historique, puis, nous donnons quelques définitions de la cryptographie et leurs fonctions. Par la suite, nous présentons une classification des algorithmes cryptographique et on parle sur quelque autre algorithmes associés, enfin nous terminons ce chapitre par la cryptanalyse et ses différents types utiliser.

# 2. La cryptographie au cours de l'histoire

De l'antiquité à la renaissance, l'art de la cryptographie met en œuvre, pour cacher la substance d'un texte, une combinaison plus ou moins élaborée de substitutions et de permutations dès l'origine, la cryptographie a été utilisée à des fins diplomatiques puis militaires.

## 2.1 L'origine des codes secrets

En Egypte, 2000 ans avant notre ère un scribe avait gravé des hiéroglyphes transformés sur la pierre tombale de Khumhotep II pour rendre inintelligible la description de sa vie, A la même époque, un général Grec, Enée le tacticien, consacre un chapitre de son ouvrage Commentaires sur la défense des places fortes à la sécurité des communications et donne quelques méthodes pour chiffrer.

Jules César dans la Guerre des Gaules décrit un procédé de substitution aujourd'hui bien connu: Il consiste pour chiffrer à décaler d'un nombre de rangs convenu la lettre « Claire » dans l'alphabet usuel. Si la clé est 3, a est remplacé par d et b par etc.…

**Figure 1 : Le scytale**

A Sparte, au IV ème siècle avant notre ère, les communications entre les chefs des armées et les commandants étaient chiffrées à l'aide d'une scytale un bâton sur lequel on enroule une lanière en spires jointives (figure1).

**Figure 2 : Disque de Phaistos été retrouvé**

En Crète, un disque de Phaistos, datant de 1700 avant notre ère, comportant un texte chiffré sur les deux faces à Ce texte n'est encore à ce jour décrypté de manière sure (figure2). [1]

## 2.2 La renaissance: L'éveil cryptographique

Abu Bakr ben Wahshiyya publie plusieurs alphabets secrets utilisés à des fins de magie, dans son livre "Kitab shauk almustaham fi ma'arifat rumuz al aklam" (Le livre de la connaissance longuement désirée des alphabets occultes enfin dévoilée).

Blaise de Vigenère écrit son Traité des chiffres ou secrètes manières d'écrire.

Il présente entre autres un tableau du type Trithème, que l'on dénomme Aujourd'hui à tort carré de Vigenère. On considéra longtemps ce chiffre comme indécryptable, légende si tenace que même en 1917, plus de cinquante après avoir été cassé, le Vigenère était donné pour "impossible à décrypter" par la très sérieuse revue Scientifiques American.

## 2.3 L'essor des communications

En 1917 Gilbert S. Vernam, travaillant pour AT&T, a inventé une machine de chiffre poly alphabétique pratique capable d'employer une clé qui est totalement aléatoire et ne se répète jamais - un masque jetable.

C'est seul le chiffre, dans nos connaissances actuelles, dont on a prouvé qu'il était indécryptable en pratique et en théorie. Ce procédé ne fut cependant jamais utilisé par l'armée car il exigeait de voir produire des millions de clés différentes (une par message), ce qui est impraticable. Par contre, il fut utilisé par les diplomates allemands dès 1921.

Boris Caesar Wilhelm Hagelin (1892-1983) propose à l'armée suédoise la machine B-21, qui fut pendant une décennie la machine la plus compacte capable d'imprimer des messages chiffrés. Pendant la seconde guerre mondiale, les Alliés fabriquèrent une autre machine de Hagelin, la Hagelin C-36 (appelée M-209 aux États-Unis), à 140 000 exemplaires. Après la guerre, Boris Hagelin créa à Zoug, en Suisse, Crypto AG, qui est aujourd'hui encore l'un des principaux fabricants d'équipements cryptographiques.

Crypto AG =develops and produces security systems for all common information and communication technologies.

La machine Enigma ne fut pas un succès commercial mais elle fut reprise et améliorée pour devenir la machine cryptographique de l'Allemagne nazie. Elle a été cassée par le mathématicien polonais Marian Rejewski, qui s'est basé seulement sur un texte chiffré et une liste des clés quotidiennes obtenues par un espion.Pendant la guerre, les messages furent régulièrement décryptés par Alan Turing**,** Gordon

Welchman et d'autres à Bletchley Parc, en Angleterre, à l'aide des premiers ordinateurs (les fameuses bombes).

## 2.4 La cryptographie moderne : de 1970 à nos jours

Les ordinateurs et le réseau Internet font entrer la cryptographie dans son ère moderne. La grande invention de ces dernières décennies fut la cryptographie à clés publiques. Le futur sera peut-être la cryptographie quantique, définitivement indécryptable.

Au début des années 1970, Horst Feistel a mené un projet de recherche à l'IBM Watson Research Lab qui a développé le chiffre Lucifer, qui inspira plus tard le chiffre DES et d'autres chiffres.

Un avantage de ce type d'algorithmes est que chiffrement et déchiffrement sont structurellement identiques.

Whitfield Diffie et Martin Hellman publient *New Directions in Cryptography*, Introduisant l'idée de cryptographie à clé publique. Ils donnent une solution Entièrement nouvelle au problème de l'échange de clés. Ils avancent aussi l'idée d'authentification à l'aide d'une fonction à sens unique.

Ils terminent leur papier avec une observation :

"L'habileté dans la cryptanalyse a toujours été lourdement du côté des professionnels, mais l'innovation, en particulier dans la conception des nouveaux types de systèmes cryptographiques, est venue principalement d'amateurs.

En 1976 DES, pour Data Encryption Standard ("standard de cryptage de données") est un algorithme très répandu à clé privée dérivé du chiffre Lucifer de Feistel (de chez IBM) dans sa version à 64 bits. Il sert à la cryptographie et l'authentification de données. Il a été jugé si difficile à percer par le gouvernement des Etats-Unis qu'il a été adopté par le ministère de la défense des Etats-Unis qui a contrôlé depuis lors son exportation. Cet algorithme a été étudié intensivement et est devenu l'algorithme le mieux connu et le plus utilisé dans le monde à ce jour.

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

RSA signifie Rivest-Shamir-Adleman, en l'honneur de ses trois inventeurs : Ron Rivest, Adi Shamir et Leonard Adleman qui l'ont inventé en 1977. Le brevet de cet algorithme appartient à la société américaine RSA Data Security, qui fait maintenant

partie de Security Dynamics et aux Public Key Parteners, (PKP à Sunnyvale, Californie, Etats-Unis) qui possèdent les droits en général sur les algorithmes à clé publique. RSA est un algorithme à clé publique qui sert aussi bien à la cryptographie de documents, qu'à l'authentification. Grâce au fait qu'il était à clé publique, et au fait qu'il était très sûr, l'algorithme RSA est devenu un standard de facto dans le monde.
L'implémentation fut achevée en 1978 par Rivest, Shamir et Adleman. Depuis ce système de chiffrement est appelé RSA, qui sont les initiales de ces trois chercheurs.
En 1990 Xuejia Lai et James Massey publient A Proposal for a New Block Encryptions Standard, un algorithme de cryptage des données International (l'IDEA: International Data Encryption Algorithm) - pour remplacer le DES. L'IDEA emploie une clé de 128 bits et utilise des opérations convenant bien à tout type d'ordinateurs, permettant donc une programmation plus efficace. Il s'agit d'un des meilleurs algorithmes de chiffrement, si ce n'est le meilleur. Personne n'a dévoilé à ce jour avoir cassé d'une manière ou d'une autre le moindre bloc de texte chiffré par IDEA. Il est actuellement exploité par la société Mediacrypt.
En 1991 Phil Zimmermann sort sa première version de PGP (Pretty Good Privacy) en réponse à la menace du FBI d'exiger l'accès au message clair des citoyens. PGP offre une haute sécurité au citoyen et cela gratuitement. PGP est en effet un freeware et est devenu rapidement une norme mondiale.
Pretty Good Privacy (PGP)=programme gratuit de protection du courrier électronique conçu en 1991 par Philip Zimmermann Sa philosophie est que tout individu a droit à la confidentialité, notamment les organisations des droits de l'homme dans des pays soumis à la dictature. C'est dans cette optique qu'il a créé PGP et qu'il l'a mis à disposition gratuitement sur Internet. Cela lui a valu de sérieux ennuis avec la justice américaine, car les logiciels de cryptage sont considérés comme du matériel de guerre et sont interdits à l'exportation. [2]

## 3. C'est quoi la cryptographie ?

### 3.1 Définition de la cryptographie

La cryptographie est l'art de rendre inintelligible, de crypter, de coder un message à ceux qui ne sont pas habilités à en prendre connaissance ou est un ensemble des principes méthodes et techniques dont l'application assure le « crypter » et le « décrypter » des données. La signification de « Crypter » et « décrypter » est donné successivement comme suit:

- ✓ **Crypter:** brouiller l'information, la rendre "incompréhensible".
- ✓ **Décrypter:** rendre le message compréhensible.

## 3.2 La cryptographie et la cryptologie

La cryptologie est la science des messages secrets, elle est composée de deux composantes complémentaires :

**1. La cryptographie** : étude et conception des procédés de chiffrement des informations.

**2. La cryptanalyse** : analyse des textes chiffrés pour retrouver l'information dissimulée.

Le mot cryptologie est souvent utilisé comme synonyme de cryptographie.

## 3.3 Principes de la cryptographie (principe de kirchoff)

• La sécurité du chiffre ne doit pas dépendre de ce qui ne peut pas être facilement changé.

•la sécurité repose sur le secret de la clé, et non sur le secret de l'algorithme (19ème siècle).

• Le déchiffrement sans la clé est impossible (*à l'échelle humaine*).

• Trouver la clé à partir du clair et du chiffré est impossible (*à l'échelle humaine*). [3]

## 3.4 L'objectif fondamental de la cryptographie

Est de permettre à deux personnes de communiquer au travers d'un canal peu sûr (téléphone, réseau informatique ou autre) sans qu'un opposant puisse comprendre ce qui est échangé.

## 3.5 Où s'applique ?

• le titulaire d'un compte en banque qui veut retirer de l'argent, ou qui paye par carte bancaire.

• un avion au moment de la procédure d'atterrissage, un véhicule suivi par GPS.

• sur un champ de bataille ses propres troupes et celles de l'ennemi,...à entrainé le développement des protocoles d'identification largement basés sur les principes des codes asymétriques.

• Les protocoles de preuves sans apport de connaissance.

• La signature aveugle.

La nécessité de conserver l'anonymat dans certaines situations (secret médical par exemple, secret de la vie privée, etc....) Peut être assuré grâce au

• Le transfert inconscient.

Il y a encore bien d'autres applications.

## 3.6 Les postulats de sécurité associés à la cryptographie

### 3.6.1 Confidentialité

Un mécanisme pour transmettre des données de telle sorte que seul le destinataire autorisé puisse les lire.

### 3.6.2 Intégrité des données

Un mécanisme pour s'assurer que les données reçues n'ont pas été modifiées durant la transmission, frauduleusement ou accidentellement.

### 3.6.3 Authentification

Un mécanisme pour permettre d'authentifier les utilisateurs de façon à limiter l'accès aux données, serveurs et ressources aux seules personnes autorisées (un mot de passe par un nom de login ou un certificat numérique).

### 3.6.4 Non-répudiation

Un mécanisme pour enregistrer un acte ou un engagement d'une personne ou d'une entité de telle sorte que celle-ci ne puisse pas nier avoir accompli cet acte ou pris cet engagement. Ce mécanisme se décompose:

- non-répudiation d'origine l'émetteur ne peut nier avoir écrit le message.
- non-répudiation de réception le receveur ne peut nier avoir reçu le message.
- non-répudiation de transmission l'émetteur du message ne peut nier avoir envoyé le message.

## 4. Classification des algorithmes cryptographiques

On peut effectuer une classification des algorithmes cryptographiques selon déférents critères, par exemple en peut les classifier selon l'apparition (algorithmes cryptographiques classiques et modernes), ou selon la nature de la clé (publique ou secrète), dans ce chapitre nous essayons de les classifier en prenant en compte tout ces critères.

## 4.1 La cryptographie classique

La cryptographie classique 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. Les meilleurs systèmes (de cette classe d'algorithmes) répètent ces deux opérations de base plusieurs fois. Cela suppose que les procédures (de chiffrement ou déchiffrement) soient gardées secrètes (figure 3).

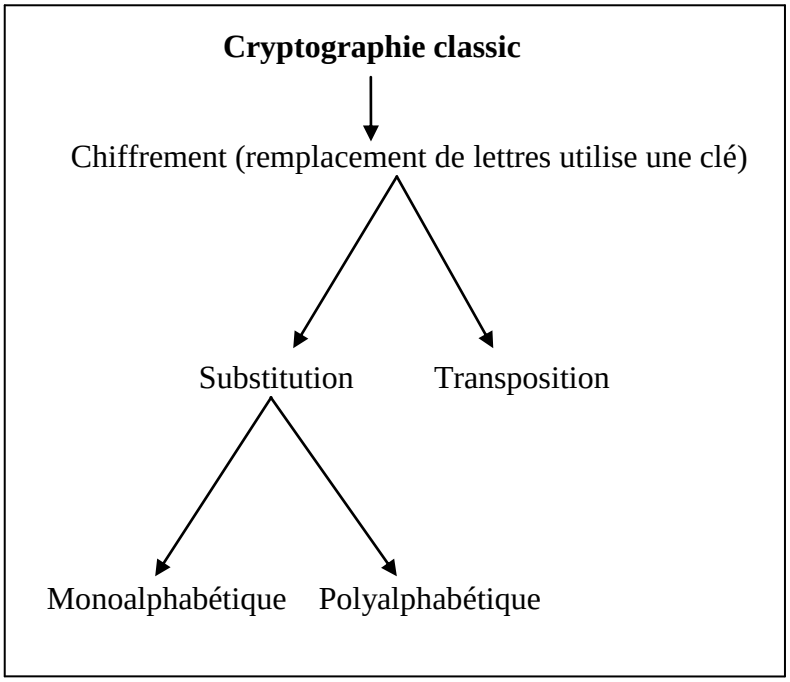



**Figure 3 :** La cryptographie classique**.**

### 4.1.1 Chiffrement par substitution

#### 4.1.1.1 Système de César

L'un des systèmes les plus anciens et les plus simples est le codage par substitution mono - alphabétique (ou alphabets désordonnés). Il consiste à remplacer chaque lettre par une lettre différente. Il existe donc grâce à cet technique 26 façons de coder un message, ce qui fait que ce système a été longtemps utilisé par les armées pendant l'antiquité. Ce procédé très fiable à l'époque est tout de même problématique car il nécessite que les interlocuteurs se souviennent tous deux de la clé. De plus, il est évident que la sûreté de ce codage est quasi nulle et qu'il pourrait être déchiffré par n'importe quelle personne qui y mettrait le temps nécessaire.

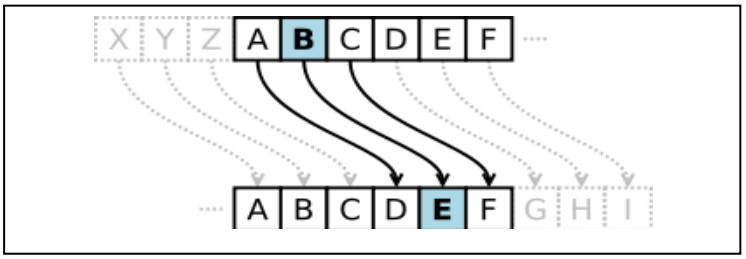



**Figure 4 : Décalage selon le chiffrier de César**

Exemple:

Si on utilise un décalage de 3, A est remplacé par D, B devient E, et ainsi de suite ...

Original: HELLO WORLD codé: KHOOR ZRUOG

| **Texte Clair** | *H* | *E* | *L* | *L* | *O* | *W* | *O* | *R* | *L* | *D* |
|---|---|---|---|---|---|---|---|---|---|---|
| **Texte Codé** | *K* | *H* | *O* | *O* | *R* | *Z* | *R* | *U* | *O* | *G* |

**Tableau 1:Chiffrement de césar**

La méthode la plus ancienne admise par l'histoire (par substitution alphabétique) est le non moins connu code de César, consistant en un décalage simple de lettres. Par substitution si l'on remplace le A par le C, alors le B devient D, le D un F, etc.… César utilisait ce code simple pour transmettre via un message des consignes à ces généraux d'armées sans qu'il puisse être exploité par un quelconque ennemi dans le cas ou le message serait intercepté.

Malheureusement il n'y a que 26 façons différentes de chiffrer à l'aide de ce code ce qui en fait un code très peu sûr. Mais ce qui est d'autant plus insolite, c'est le fait que ce code de « César » est encore utilisé de nos jours sur Internet avec le ROT13 (rotation de 13 lettres) qui consiste à cacher des messages afin qu'ils ne soient pas lus involontairement, comme par exemple s'ils dévoilent le dénouement d'un film ou encore qui donne la réponse à une devinette.

### 4.1.1.2 Système de Vigénère

Un autre système de cryptographie des plus anciens est cette fois-ci, la substitution poly alphabétique, qui utilise plusieurs alphabets décalés pour crypter un message. L'algorithme de substitution poly alphabétique le plus connu est le chiffre de Vigenère, mis au point par Blaise de Vigenère en 1586, qui fut utilisé pendant plus de 3 siècles. Son chiffre consiste à utiliser le chiffre de César, mais en changeant le décalage à chaque fois. Il utilise alors un carré composé de 26 alphabets alignés, décalés de colonne en colonne d'un caractère. Il place également au dessus de ce carré, un alphabet pour la clé et à sa gauche un autre alphabet pour le texte à coder. Il suffit alors, pour chiffrer un message, de choisir un mot de longueur quelconque, de l'écrire sous le message à coder (de façon répétée s'il le faut) et de regarder dans le tableau l'intersection de la lettre à coder et de la lettre de la clé. Pour mieux

comprendre le fonctionnement du Carré de Vigenère nous vous proposons cet exemple :

Supposons que nous voulons coder le texte «CARRE DE VIGENERE» avec la clé « MALICE ». On commence par écrire la clé sous le texte à coder :

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Texte Clair | C | A | R | R | R | E | D | E | V | I | G | E | N | R | E |
| Texte Codé | M | A | L | I | C | E | M | A | L | I | C | E | M | A | L |

**Tableau 2:Chiffrement de vigénere**

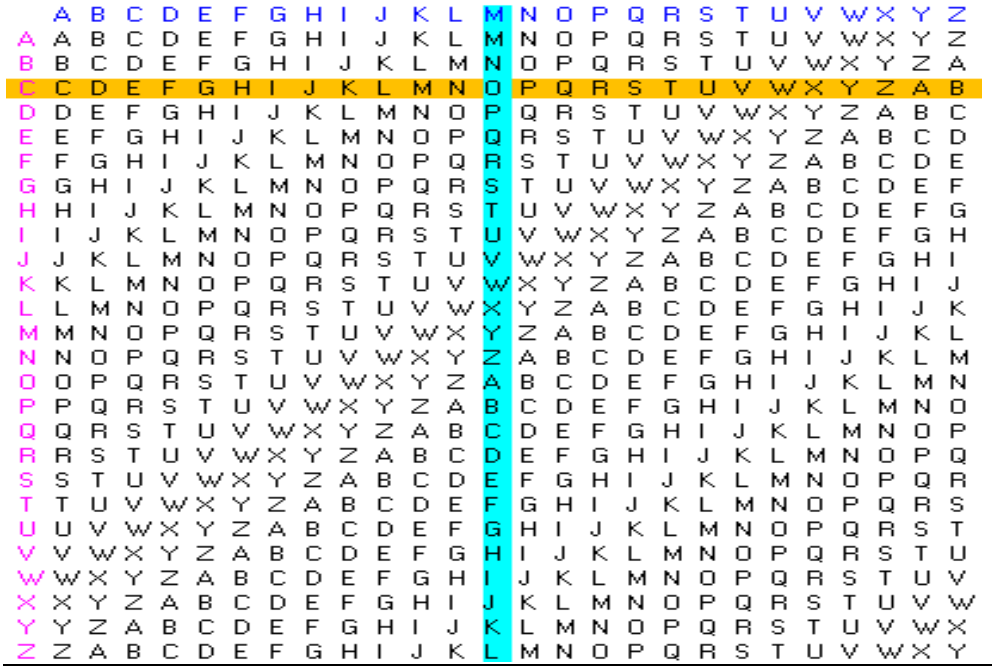


**Figure 5:Carré de vigénere**

Pour coder la lettre C, la clé est donnée par la lettre M. On regarde dans le tableau l'intersection de la ligne donnée par le C, et de la colonne donnée par le M. On trouve O. Puis on continue, jusqu'à ce qu'on ait fini de chiffrer notre texte.

En chiffrant le Texte « Carre de Vigenere », on obtient donc le texte « OAUZGHG VTOGRQRP ».

Cet algorithme de cryptographie ainsi que celui de César sont les premiers des algorithmes à clé privée. [4]

### 4.1.2 Code de permutation ou transposition

On partage le texte en blocs, on garde le même alphabet mais on change la place des lettres à l'intérieur d'un bloc (on les permute).

Un exemple historique dont le principe est encore utilisé dans les codes à clé secrète (DES, AES) est la méthode de la grille (principe de la scytale utilisé par les spartiates).

On veut envoyer le message suivant:

RENDEZ VOUS DEMAIN MIDI VILLETANEUSE

L'expéditeur et le destinataire du message se mettent d'accord sur une grille de largeur fixée à l'avance (ici une grille de 6 cases de large).

L'expéditeur écrit le message dans la grille en remplaçant les espaces entre les mots par le symbole □ . Il obtient:

| | | | | | |
|---|---|---|---|---|---|
| R | E | N | D | E | Z |
| □ | V | O | U | S | □ |
| D | E | M | A | I | N |
| □ | M | I | D | I | □ |
| V | I | L | L | E | T |
| A | N | E | U | S | E |

Il lit le texte en colonne et obtient ainsi le message crypté

R□ D□ VAEVEMINNOMILEDUADLUESIIESZ□ N□ TE

Pour augmenter la sécurité les deux interlocuteurs peuvent décider l'ajout d'une clé.

Le but est de pouvoir changer facilement le cryptage d'un message tout en gardant le même algorithme de codage. Pour cela on rajoute une clé secrète constituée par l'ordre de lecture des colonnes.

Exemples 1. On choisit la clé: CAPTER

On numérote les colonnes en fonction du rang des lettres du mot CAPTER dans l'alphabet c'est-à-dire 2, 1, 4, 6, 3, 5

Et on lit les colonnes dans l'ordre indiqué.

EVEMINR□ D □ DADUADLUZ□N□ TENOMILEESIIES

On a 6! Codes différents.

Pour décoder le message précédent on range en colonne sur la grille en suivant l'ordre des colonnes donné par le mot de code. [5]

## 4.2 La cryptographie moderne

Pour assurer les objectifs de la cryptographie nous pouvons utiliser des algorithmes basés sur des clés. Ces algorithmes sont définis par plusieurs types de cryptographie ou cryptosystèmes :

### 4.2.1 Cryptographie symétrique

Le chiffrement symétrique (aussi appelé chiffrement à clé privée ou à clé secrète) consiste à utiliser la même clé pour le chiffrement et le déchiffrement (Figure 6). Le chiffrement consiste alors à effectuer une opération entre la clé privée et les données à chiffrer afin de rendre ces dernières inintelligibles. Le déchiffrement consiste à réaliser l'opération inverse c'est-à-dire récupérer le message d'origine à partir du message chiffré en utilisant la clé secrète. [6]

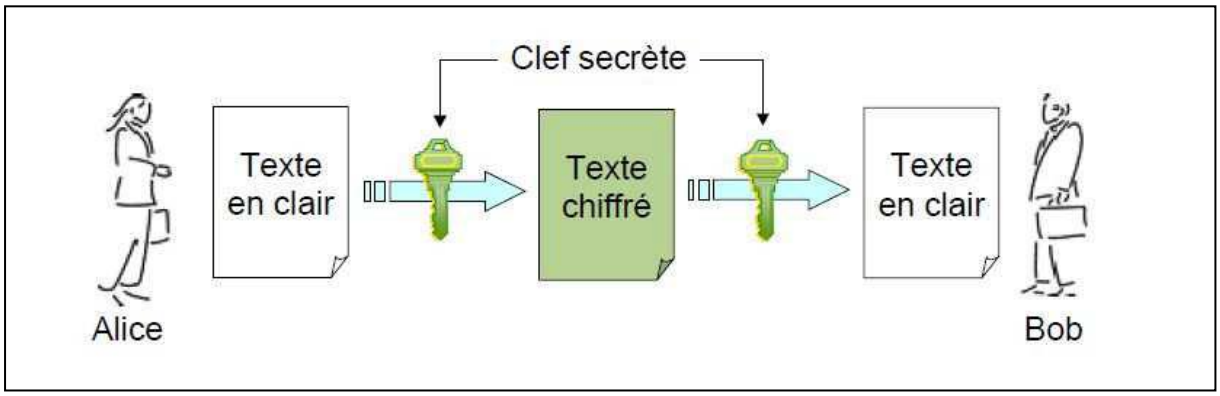



**Figure 6: La cryptographie symétrique**

Les algorithmes symétriques sont plus rapides et facilement implémentés sur le matériel. Ils sont des simples opérations de substitution et de transposition et sont mieux adaptés pour une utilisation sur le réseau Internet filaire et en mobilité. Cependant ces systèmes ne sont pas entièrement adéquats pour résoudre tous les problèmes de sécurité, Le processus de chiffrement et déchiffrement dépend de la même clé secrète partagée entre l'émetteur et le récepteur, L'émetteur et le récepteur doivent donc se mettre d'accord sur la clé secrète avant qu'ils puissent communiquer d'une façon sécurisée. Un canal sûr d'échange de clé est difficile à établir.

Les cryptosystèmes symétriques ont un problème de gestion de clé.

En effet lorsqu'un grand nombre de personnes désirent communiquer ensemble, le nombre de clés augmente de façon importante et la situation devient impraticable. Pour n participants à la communication, on aura besoin de : n (n − 1) /2 clés secrètes enregistrés.

Il y a deux catégories de systèmes à clé privée : les chiffrements par blocs et les chiffrements de flux.

#### 4.2.1.1 Chiffrement par bloc (Block Cipher)

Une des deux grandes catégories de chiffrements modernes en cryptographie symétrique. Il consiste à un découpage des données en blocs de taille généralement fixe (souvent une puissance de deux comprise entre 32 et 512 bits). Les blocs sont ensuite chiffrés les uns après les autres. Il est possible de transformer un chiffrement de bloc en un chiffrement par flot en utilisant un mode d'opération comme ECB (chaque bloc chiffré indépendamment des autres) ou CFB (on chaîne le chiffrement en effectuant un XOR entre les résultats successifs).

Un exemple de taille de bloc et de clé utilisés par les algorithmes les plus connus:

- DES : blocs de 64 bits, clé de 56 bits.
- IDEA : blocs de 64 bits, clé de 128 bits.
- AES : blocs de 128 bits, clé de 128 à 256 bits.

Pour cette catégorie, nous allons présenter deux algorithmes très connus DES et AES et IDEA en mettant l'accent sur l'AES au deuxième chapitre car il est devenu le standard recommandé pour le chiffrement symétrique.

##### 4.2.1.1.1 DES (Data Encryption Standard)

DES a été développé à la fin des années 1970s comme un standard du gouvernement US pour protéger l'information sensible. Le DES est officiellement défini dans la publication FIPS 46-3 et il est public. DES est encore supporté dans des outils de cryptographie pour les O.S des dispositifs mobiles et les cartes à puce.

C'est un chiffrement qui transforme des blocs de 64 bits avec une clé secrète de 56 bits. Il a une conception basée sur le schéma de Feistel au moyen de permutations et de substitutions. L'attaque de force brute est possible sur une clé DES. Cette attaque a été réalisée par DES Cracker de EFF-Electronic Frontier Fondation en juin 1998 (Elle a trouvé une clé DES dans moins de 3 jours).

Avec la technologie actuelle (des CPU très rapides et moins chers), il est hautement Possible de cracker DES. La solution a été au premier temps d'adopter le triple DES (TDES ou 3DES) : 3 applications de DES à la suite avec 2 clés différentes (112 bits) (Figure 7).

TDES utilise une taille de blocs est 64 bits et de clé comprise entre 128 bits et 192 bits.

TDES est suffisant en sécurité mais il est trois fois plus lent que DES. Un nouvel algorithme remplace DES, c'est l'AES (Advanced Encryption Standard).

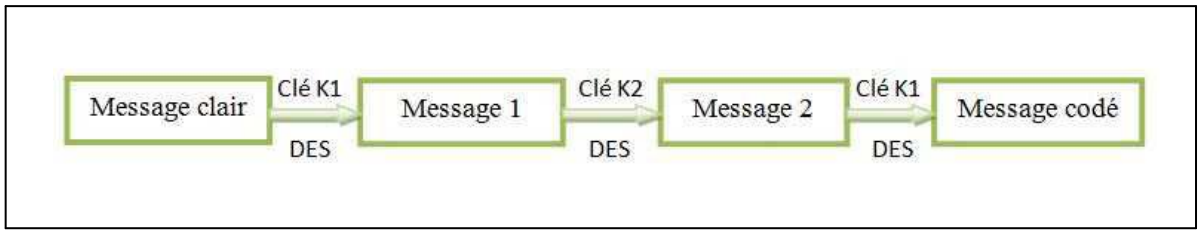



**Figure 7 : Algorithme TDES.**

#### 4.2.1.1.2 L'algorithme IDEA

IDEA pour International Data Encryption Algorithm est un algorithme de chiffrement conçu par Xuejia Lai et James Massey de **l'Institut Technologique Fédéral Suisse** et présenté pour la première fois en 1991.

L'algorithme IDEA est un algorithme symétrique de chiffrement par bloc de 64 bits (soient 8 octets) fonctionnant avec une clé de 128 bits. La particularité de cet algorithme est que les opérations qu'il utilise en interne s'adaptent très facilement à une programmation informatique sur machine de 16 bits. « IDEA » est utilisé au sein du « PGP » et est un des remplaçants potentiels du « DES ».

#### 4.2.1.1.3 AES (Advanced Encryption Standard)

En 1997, le NIST (National Institue of Standards and Technology) a lancé un appel d'offre pour un algorithme de chiffrement symétrique avec un bloc de taille 128 bits et supporte des clés de 128, 192 et 256 bits. Les critères d'évaluation de l'offre comprenaient la sécurité, la puissance de calcul, les contraintes de la mémoire des petits dispositifs (comme la carte à puce), la plateforme logicielle et matérielle et la flexibilité. Un total de 15 algorithmes ont été envoyés et 5 ont été sélectionné parmi ces 15 algorithmes.

L'algorithme de Rijndael a été sélectionné pour devenir l'AES en 2001.

Rijndael a été conçu par Joan Daemen et Vincent Rijmen, deux chercheurs de la Belgique. Les autres algorithmes sont : Serpent, Twofish, RC6, et MARS, L'AES n'utilise pas le schéma de Feistel mais il utilise des opérations mathématiques comme des substitutions, des permutations et des XOR s, Il a plusieurs rounds identiques (de 10 à 14) et leur nombre dépend de la taille de la clé. L'AES opère au niveau octet ce qui permet une implémentation efficace au niveau matérielle et logicielle. L'AES est un standard, donc libre d'utilisation, sans restriction d'usage ni brevet. NIST spécifie

actuellement AES dans le document FIPS 197, comme le nouveau standard de chiffrement symétrique.

AES est approuvé pour utilisation par les organisations du gouvernement U.S pour protéger l'information sensible non classifiée.

AES a trois niveaux forts de sécurité : 128 bits, 192 bits et 256 bits. La sécurité 128 bits fournira au mois 30 ans de protection. AES ne fournit pas seulement une sécurité supérieure à TDES mais il délivre aussi une meilleure performance. Une meilleure sécurité et une meilleure performance rendent l'AES une alternative plus attractive que TDES et un bon choix pour un algorithme de chiffrement symétrique.

AES est également un candidat particulièrement approprié pour les dispositifs mobiles limités en ressources de calcul et de stockage. Le monde de la 3G (3ème génération de dispositifs mobiles) a adopté l'algorithme AES pour son schéma d'authentification.

**Autres finalistes d'AES :**

Durant la compétition avec AES, Serpent et Twofish avaient une performance faible comparée à Rijndael. RC6 et MARS ont des problèmes de sécurité et d'efficacité. RC6 a été cassé à au moins de 17 rounds sur 20 pendant la compétition avec AES. MARS était plus couteux en calcul pour l'implémenter comparé aux autres finalistes d'AES.

#### 4.2.1.2 Chiffrements de flux (Stream Cipher)

Les algorithmes de chiffrement de flux peuvent être définis comme étant des algorithmes de chiffrement par blocs, où le bloc a une dimension unitaire (1 bit, 1 octet, etc.) ou relativement petite. Leurs avantages principaux viennent du fait que la transformation (méthode de chiffrement) peut être changée à chaque symbole du texte clair et du fait qu'ils soient extrêmement rapides. De plus, ils sont utiles dans un environnement où les erreurs sont fréquentes car ils ont l'avantage de ne pas propager les erreurs (diffusion). Ils sont aussi utilisés lorsque l'information ne peut être traitée qu'avec de petites quantités de symboles à la fois. Quelques algorithmes de cryptographie symétrique par flot:

A5 : utilisé dans les téléphones mobiles de type GSM pour chiffrer la communication par radio entre le mobile et l'antenne-relais la plus proche.

RC4, le plus répandu, conçu par Ronald Rivest, utilisé notamment par le protocole WEP, un algorithme récent de Eli Biham – E0 utilisé par le protocole Bluetooth.

#### 4.2.1.3 Considérations de sécurité pour la cryptographie symétrique

Lorsqu'on utilise la cryptographie à clé symétrique, nous devons considérer certains Principes qui constituent les critères de base pour implémenter une sécurité fiable et efficace.

##### 4.2.1.3.1 Choix de l'algorithme de chiffrement symétrique, la taille du bloc et la clé symétrique :

Il y a plusieurs méthodes d'attaques pour trouver des vulnérabilités dans les crypto systèmes symétriques en mode bloc. Les problèmes de collision sont plus dominants dans les cryptages de bloc de 64 bits comparés à ceux de 128 bits Il est donc fondamental d'utiliser des chiffrements de bloc de 128 bits.

L'utilisation de la clé symétrique n'est pas la même pour chaque crypto système, et certaines initialisations des clés peuvent casser la sécurité offerte par l'algorithme. Par exemple, DES a un problème de clé faible et semi-faible (weak and semi-weak key). DES a 4 clés faibles et 16 clés semi-faibles. Le développeur est obligé de tester pour voir si la clé utilisée est faible ou non.

Un autre algorithme symétrique RC4 a une contrainte : la même clé ne doit jamais être utilisée pour chiffrer deux suites d'octets différentes parce que la sortie du générateur est XOR avec la suite d'octets. On trouve cette faiblesse de sécurité dans le standard WEP (Wired Equivalent Privacy) qui utilise RC4 pour chiffrer les communications sans fil sur les réseaux 802.11. Le problème avec le WEP est la taille faible de la clé. Les produits WEP implémentent une clé partagée de 64 bits, 40 bits pour la clé secrète et 24 bits pour le vecteur d'initialisation (IV-Initial Vector). Le WEP n'alloue pas suffisamment de bits pour le vecteur d'initialisation, donc la même valeur de l'IV sera réutilisée, et plusieurs paquets seront chiffrés avec la même clé.

Donc une règle fondamentale dans les crypto systèmes symétrique : Chaque crypto-système symétrique utilise une structure de clé différente.

L'AES est l'algorithme le plus efficace et le plus rapide pour les plateformes logicielles et matérielles, et est le mieux approprié sur les dispositifs mobiles et PDAs. Il est donc recommandé d'utiliser l'AES pour le chiffrement/déchiffrement symétrique.

##### 4.2.1.3.2 Gestion de clé :

Il est fondamental d'utiliser un générateur fort de nombre pseudo aléatoire (PRNG Pseudo Random Number Generator) pour générer des sorties d'octets

aléatoires utilisées comme clés. Pour l'échange de la clé, on peut utiliser un centre de distribution de clé ou une cryptographie à clé asymétrique pour l'échange de la clé secrète. Généralement les applications utilisent un algorithme symétrique pour chiffrer les données, et un algorithme asymétrique pour chiffrer la clé secrète.

Cette méthode mixte symétrique et asymétrique est mieux sécurisée et plus efficace. Si la clé est utilisée pour une communication entre deux parties sur un réseau, il est recommandé d'utiliser une clé de session parce qu'elle limite la chance à un attaquant de trouver la clé exacte.

Si une clé est exigée pour chiffrer des fichiers locaux sur une machine, alors les clés générées par mot de passe (PBE-Password Based Encryption) peuvent être utilisées, mais il faut s'assurer que le mot de passe lui-même ne soit pas compromis [08]. Donc une règle fondamentale de la cryptographie symétrique : Utiliser des clés de session pour chiffrer une communication réseau et des clés PBE (basés sur des mots de passe) pour protéger les données sensibles sur la machines.

#### 4.2.1.3.3 Authentifier le chiffrement symétrique

Les modes de chiffrement protègent les données de l'écoute mais ils ne fournissent aucune authentification. La fonction de déchiffrement déchiffre juste les données, elle peut déchiffrer un texte chiffré déjà modifié, en un certain plaintext qui est différent du plain text d'origine.

Dans un tel cas, il est recommandé de produire un code d'authentification du message MAC (Message Authentification Code), du texte chiffré et ajouter ce MAC au texte chiffré.

Utiliser le chiffrement en mode bloc au lieu du chiffrement par flot : Il est recommandé d'utiliser le mode de chiffrement en bloc (CBC ou CTR) avec un vecteur d'initialisation générée d'une manière aléatoire. Un vecteur d'initialisation mal choisi peut causer une faiblesse de sécurité (le cas de RC4 avec le protocole WEP).

### 4.2.1.4 Avantages et Inconvénients de la cryptographie symétrique

**4.2.1.4.1 Avantages :** Les avantages de la cryptographie symétrique sont:

- Système rapide du chiffrement/déchiffrement;
- Clés relativement courtes (128 ou 256 bits);
- Bonnes performances et sécurité bien étudié;

**4.2.1.4.2 Inconvénients:** Les inconvénients de la cryptographie symétrique sont:

- Gestion des clés difficiles (nombreux clés) ;

- Point faible = l'échange de la clé secrète ;
- Dans un réseau de N entités susceptibles de communiquer secrètement il faut distribuer N*(N-1)/2 clés. [7]

### 4.2 .2 Cryptographie asymétrique

La cryptographie asymétrique se base sur des problèmes mathématiques complexes (factorisation de grands nombres entiers ou équation de logarithme discrète). La cryptographie asymétrique se base sur le principe de deux clés: clé publique, clé privée. La clé publique est mise à la disposition de quiconque désire chiffrer un message (cette clé peut être connu par tout le monde). Ce dernier ne pourra être déchiffré qu'avec la clé privée, qui doit être confidentielle et connue seulement par son propriétaire.

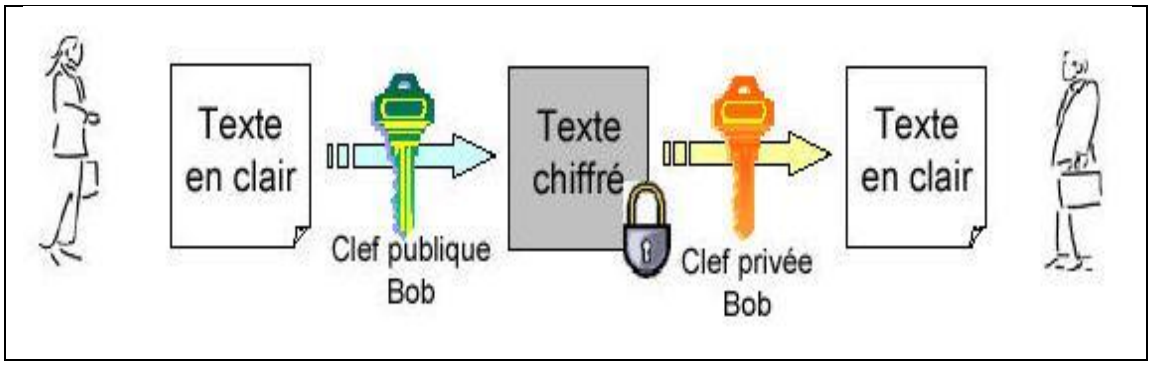



**Figure 8 : Chiffrement Asymétrique**

Un chiffrement asymétrique est défini par trois algorithmes :

1. Algorithme de génération des clés.
2. Algorithme de chiffrement.
3. Algorithme de déchiffrement.

Certains algorithmes asymétriques comme RSA offre aussi des opérations pour la génération de signature numérique et sa vérification.

L'utilisation d'une paire de clés publique/privée permet d'assurer la confidentialité, l'authentification, l'intégrité et l'échange de la clé secrète. Cependant les algorithmes asymétriques ne réalisent tous ces fonctions.

Le tableau 3 donne un résumé des opérations cryptographiques pouvant être réalisés par les algorithmes asymétriques les plus connus.

| Algorithme | Chiffrement/déchiffrement | Signature num | Echange de clé |
|---|---|---|---|
| RSA | Oui | Oui | Oui |
| Diffie-Hellman | Non | Non | Oui |
| DSA | Non | Oui | Non |
| EC-Elliptic Curves | Oui | Oui | Oui |

**Tableau 3:Les applications des crypto système asymétrique**

La Cryptographie à clé publique basé sur des problèmes mathématiques difficiles à résoudre :

### 4.2.2.1 Problème de la factorisation

Le problème de la factorisation consiste à retrouver la décomposition en facteurs premiers d'un entier obtenu de manière secrète par multiplication de deux nombres premiers de taille comparable. Un tel nombre composé est classiquement appelé « module ».L'utilisation du problème de la factorisation est en général effectuée au moyen du crypto systèmes RSA. [7]

### 4.2.2.2 RSA

L'algorithme le plus célèbre d'algorithme à clé publique a été inventé en 1977 par RonRiv et Adi Shamir et Len Adleman, à la suite de la publication de l'idée d'une cryptographie à clé publique par Diffie et Hellman. Il fut appelé RSA, des initiales de ces inventeurs.

RSA est basé sur la difficulté de factoriser un grand nombre en produit de deux grands facteurs premiers. L'algorithme fonctionne de la manière suivante :

Imaginons que Bob souhaite recevoir d'Alice des messages en utilisant RSA.

#### 4.2.2.2.1 Génération des clés

**A.** p et q, deux grands nombres premiers sont générés au hasard grâce à un algorithme de test de primalité probabiliste, avec n = pq.

**B.** Un nombre entier e premier avec (p-1) (q-1) est choisi. Deux nombres sont premiers entre eux s'ils n'ont pas d'autre facteur commun que 1.

**C.** L'entier d est l'entier de l'intervalle [2, (p-1) (q-1)] tel que ed soit congrue à 1 modulo (p-1) (q-1), c'est-à-dire tel que ed-1 soit un multiple de (p-1) (q-1).

#### 4.2.2.2.2 Distribution des clés

Le couple (n, e) constitue la clé publique de Bob. Il la rend disponible à Alice en lui envoyant ou en la mettant dans un annuaire. Le couple (n, d) constitue quand à lui sa clé privée.

#### 4.2.2.2.3 Chiffrement du message

Pour crypter le message Alice représente le message sous la forme d'un ou plusieurs entiers M compris entre 0 et n-1.Elle calcule C= Me mod n grâce à la clé publique (n, e) de Bob et envoie C à Bob.

#### 4.2.2.2.4 Déchiffrement du message

Bob reçoit C et calcule grâce à sa clé privée Cd mod n.

Il obtient ainsi le message initial M.

Exemple :

Bob choisit p = 17 et q = 19, n = p x q = 323 et e = 5.

Sa clé privée est alors d=173 car 173 x 5 = 1 (mod (16 x 18))

Supposons qu'Alice veuille lui envoyer le message « BONJOUR » en se servant du tableau suivant pour transformer les lettres en nombre :

| A | B | C | D | E | F | G | H | I | J | Q | L | M | N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |

| O | P | K | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 |

Cela donne :

| B | O | N | J | O | U | R |
|---|---|---|---|---|---|---|
| 2 | 15 | 14 | 10 | 15 | 21 | 18 |

Après avoir chiffré en remplaçant chaque nombre b par (be mod n) on obtient :

| 32 | 2 | 29 | 193 | 2 | 89 | 18 |
|---|---|---|---|---|---|---|

Qu'Alice envoie à Bob.

Bob réalise pour chaque nombre b du message bd mod n pour trouver :

| 2 | 15 | 14 | 10 | 15 | 21 | 18 |
|---|---|---|---|---|---|---|
| B | O | N | J | O | U | R |

Qui est bien le message initial.

RSA est basé sur la difficulté de factoriser n. En effet celui qui arrive à factoriser n peut retrouver facilement la clé secrète de Bob connaissant seulement sa clé publique. C'est pourquoi dans la pratique la taille des clés est au minimum de 512 bits. [4]

#### 4.2.2.3 SSL

Le protocole SSL (Secure Sockets Layers, que l'on pourrait traduire par « couche de transport sécurisé »), est un procédé développé par Netscape ayant pour but de sécuriser les transactions effectuées sur Internet. De nombreux sites de commerces de nos jours sont sécurisés avec SSL (afin de communiquer sûrement avec leurs clients et d'obtenir le paiement de leurs ventes). Ce système repose à la fois sur les algorithmes à clé publique (RSA, Diffie- Hellmann), sur les algorithmes à clé privée et sur les certificats électroniques (que nous traiterons plus tard) afin de garantir au maximum la sécurité de la transmission de données avec un tel site. Cependant un utilisateur quelconque ignore en principe totalement qu'il utilise SSL, car celui-ci agit de manière transparente.

En fait SSL est indépendant des protocoles De communications, il agit directement entre la gestion des commandes et la gestion du transport des données (il agit comme une couche supplémentaire de protection).

De cette façon un utilisateur se connectant à un site de commerce sécurisé via un navigateur Internet enverra des données chiffrées (code de Carte Bleue..) par SSL s'en même s'en préoccuper. Seule l'apparition d'un petit cadenas s'affichant dans le navigateur et l'url commençant par https:// (le «s» signifiant secured) pourront permettre à un utilisateur averti de se rendre compte de l'intervention de SSL dans un tel échange.

L'utilisation de SSL est donc assez simple étant donné qu'elle se fait seule : son fonctionnement l'est aussi. La transaction par protocole SSL est en fait basé sur un échange de clé entre un client et un serveur. La transaction sécurisée se fait selon un schéma bien défini. Pour commencer, le client se connecte à un site de commerce (grâce à un navigateur utilisant SSL) en lui demandant de s'identifier (ce qui évitera de nombreux problèmes comme nous le verrons dans le chapitre concernant les attaques sur SSL). Le client envoie aussi une liste des crypto systèmes qu'il connaît. Ensuite le serveur s'identifie en envoyant un certificat d'authentification contenant la clé publique du serveur ainsi que le nom du crypto système qui lui convient (souvent celui dont la longueur de clef est la plus long).

A la réception, le client doit alors vérifier la validité du certificat envoyé par le marchant (serveur). Il choisi en suite une clé secrète (de manière aléatoire) et l'envoie au serveur sous forme cryptée grâce à la clef publique du serveur (c'est la que RSA intervient). Cette nouvelle clé est appelée clé de session. Le serveur est enfin capable de déchiffrer le reste des transactions par le biais de la clé de session. Le serveur ainsi que le client sont donc tous deux en possession d'une clé commune, permettant alors la confidentialité des données échangées.

Une dernière authentification est possible, celle du client. Elle permettrait encore une plus grande sécurité, mais elle est en fait très rarement utilisée dans les utilisations courantes de SSL.

L'utilisation du système SSL ne cesse de s'accroître. En outre il semble important de rappeler que SSL est indépendant du protocole utilisé (couche supplémentaire), c'est-à dire qui peut non seulement sécuriser des transactions effectuées sur le Web mais aussi des connexions par FTP, POP, TELNET, IMAP, SMTP, etc.… [4]

### 4.2.2.4 El Gamal

C'est un algorithme à clé publique présent à la base de la norme U.S. de signature électronique. Il fut inventé par Tahar El Gamal en 1984. Il est basé sur la difficulté de calculer des logarithmes discrets.

Le problème du logarithme discret consiste à retrouver un entier $\gamma$ tel que h = g puis $\gamma$ mod p. [6]

Le Cryptosystème de El Gamal, ou chiffrement El Gamal (ou encore système d'El Gamal ...) est un algorithme de cryptographie asymétrique fondé sur le problème du logarithme discret. Il a été créé par Taher ElGamal. Cet algorithme est utilisé par le logiciel libre GNU Privacy Guard, de récentes versions de PGP, et d'autres systèmes de chiffrement, et n'a jamais été sous la protection d'un brevet contrairement à RSA. Il peut être utilisé pour le chiffrement, mais aussi la signature électronique, par exemple l'algorithme DSA du NIST.

L'algorithme est décrit pour un groupe multiplicatif $\mathbb{Z}_p^*$, $p$ premier, mais n'importe quel groupe cyclique fini pour lequel le problème du logarithme discret est difficile convient. On suppose que Bob veut envoyer un message à Alice (le chiffrement est asymétrique).

- Alice calcule deux clés, une clé publique et une clé privée : elle choisit d'abord p suffisamment grand pour que le calcul du logarithme discret soit infaisable pratiquement dans le groupe multiplicatif $\mathbb{Z}_p^*$, $g$ un générateur de ce groupe et un entier naturel $s$, $s < p$, puis calcule $h = g^s \bmod p$. L'entier $s$ est la clé secrète, le triplet ($p$, $g$, $h$) la *clé publique*. Cette dernière seule est connue de Bob.
- Le message clair de Bob est supposé être un m dans $\mathbb{Z}_p^*$. Bob choisit aléatoirement un nombre entier $k$ puis calcule (dans $\mathbb{Z}_p^*$) $c_1 = g^k$ et $c_2 = mh^k$. Le message chiffré est le couple ($c_1$, $c2$) que Bob envoie à Alice.
- Alice peut déchiffrer le message reçu en calculant $m = c_2 / {c_1}^s$. En effet :

$$\frac{c_2}{{c_1}^s} = \frac{m \cdot h^k}{g^{ks}} = \frac{m \cdot h^k}{h^k} = m$$

Casser l'algorithme El Gamal est dans la plupart des cas au moins aussi difficile que de calculer le logarithme discret. Cependant, il est possible qu'il existe des moyens de casser l'algorithme sans résoudre le problème du logarithme discret.

#### 4.2.2.5 Avantages et Inconvénients de la cryptographie asymétrique

**4.2.2.5.1 Avantages :** Les avantages de la cryptographie asymétrique sont :

- ✓ Gestion de la secrète facilitée ;
- ✓ Pas de secrète à transmettre ;
- ✓ Nombre de clés à distribuer est réduit par rapport aux clés symétriques ;
- ✓ Très utile pour échanger les clés ;
- ✓ Permet de signer des messages facilement ;

**4.2.2.5.2 Inconvénients :** Les inconvénients de la cryptographie asymétrique sont:

La relation clé publique /clé privée impose :

- ✓ Des clés plus longues (1024 à 4096 bits) ;
- ✓ Gestion de certificats de clés publiques ;
- ✓ Lenteur de calcul;
- ✓ Pas d'authentification de la source.

### 4.2.3 Les algorithmes associés

#### 4.2.3.1 La cryptographie hybride

Cette technique a été introduite afin de profiter des avantages des deux techniques précédemment citées, c'est à dire la rapidité de traitement des messages

codés par cryptographie symétrique et la puissance du chiffrement des messages par cryptographie asymétrique.

Le principe est assez simple. La communication entre A et B se fait par système cryptographique symétrique, ce qui rend rend la communication assez rapide à chiffrer et déchiffrer. Mai sella lacune de la sécurité de transmission de la clé symétrique de chiffrement/déchiffrement est palliée par un chiffrement de cette clé, qui lui est asymétrique.

#### 4.2.3.2 PGP

L'email est l'un des services réseau les plus utilisés. Le contenu des messages n'est pas sécurisé, il peut donc être inspecté durant son transit ou par les utilisateurs privilégiés sur le système de destination.

On souhaite donc la mise en place de services de sécurité tels que la confidentialité, l'authentification, l'intégrité et la non-répudiation. Ces quatre services sont apportés par le logiciel PGP.

Développé par Phil Zimmermann, PGP fournit un service de confidentialité et d'authentification qui peut être employé pour des applications de type courrier électronique et stockage de dossiers.

Lorsqu'un utilisateur crypte du texte en clair avec PGP, ces données sont d'abord compressées. Cette compression des données permet de réduire le temps de transmission par modem, d'économiser l'espace disque et, surtout, de renforcer la sécurité cryptographique. La plupart des cryptanalystes exploitent les modèles trouvés dans le texte en clair pour casser le chiffrement. La compression réduit ces modèles dans le texte en clair, améliorant par conséquent considérablement la résistance à la cryptanalyse. Toutefois, la compression est impossible sur les fichiers de taille insuffisante ou supportant mal ce processus.

PGP crée ensuite une clé de session qui est une clé secrète à usage unique. Cette clé correspond à un nombre aléatoire, généré par les déplacements aléatoires de votre souris et les séquences de frappes de touches. Pour crypter le texte en clair, cette clé de session utilise un algorithme de cryptage conventionnel rapide et sécurisé. Une fois les données codées, la clé de session est cryptée vers la clé publique du destinataire. Cette clé de session cryptée par clé publique est transmise avec le texte chiffré au destinataire.

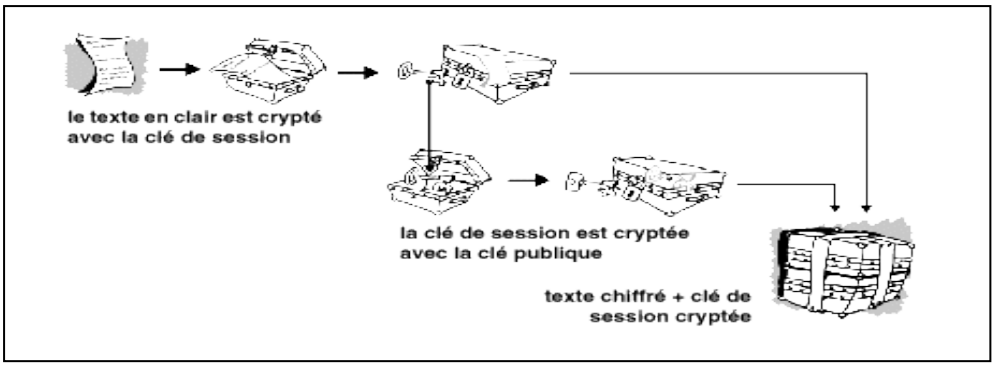



**Figure 9 : Fonctionnement du cryptage PGP**

Le processus de décryptage est inverse. La copie de PGP du destinataire utilise sa clé privée pour récupérer la clé de session temporaire qui permettra ensuite de décrypter le texte crypté de manière conventionnelle.

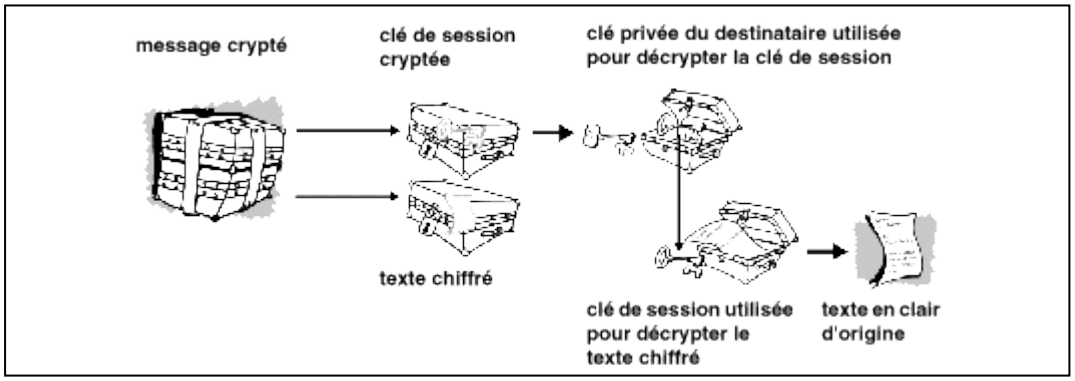



**Figure 10: Fonctionnement du décryptage PGP**

Ces deux méthodes de cryptage associent la facilité d'utilisation du cryptage de clé publique à la vitesse du cryptage conventionnel. Le cryptage conventionnel est environ 1 000 fois plus rapide que le cryptage de clé publique. De plus, le cryptage de clé publique résout non seulement le problème de la distribution des clés, mais également de la transmission des données. Utilisées conjointement, ces deux méthodes améliorent la performance et la distribution des clés, sans pour autant compromettre la sécurité. [8]

### 4.2.3.3 Fonctions de hachage

Lors d'échanges de messages cryptés, il est important de pouvoir s'assurer que le message n'a pas été altéré ou modifié par un tiers pendant l'envoi. Les fonctions de hachage permettent alors de s'assurer de l'intégrité du message.

Une fonction de hachage calcule l'empreinte y (ou digest) d'un message x. Cette fonction F doit être une fonction à sens unique c'est-à-dire qu'il doit être facile de trouver y à partir de x, mais très difficile de trouver x à partir de y. Elle doit aussi

être très sensible pour qu'une petite modification du message entraîne une grande modification de l'empreinte.

En envoyant le message accompagné de son empreinte, le destinataire peut ainsi s'assurer de l'intégrité du message en recalculant le résumé à l'arrivée et en le comparant à celui reçu.

Si les deux résumés sont différents, cela signifie que le fichier n'est plus le même que l'original : Il a été altéré ou modifié par une tierce personne.

Les fonctions de hachage les plus répandus sont MD5 et SHA-1 qui sont basés tous les deux sur MD4, MD5 générant des empreintes de 128 bits et SHA-1 de 160 bits (seul MD5 sera décrit, ces deux fonctions ayant un fonctionnement similaire).

Nous verrons plus en avant qu'une fonction de hachage joue un rôle dans la signature électronique, méthode qui permet d'authentifier l'expéditeur. [4]

#### 4.2.3.4 Signatures

Les fonctions de hachage permettent de s'assurer de l'intégrité d'un message mais un autre problème se pose : comment être certain que personne n'a usurpé l'identité de l'expéditeur pour vous envoyer un message ? Ou que l'expéditeur ne va pas nier vous l'avoir envoyé ?

C'est le rôle de la signature numérique, celle-ci fournissant donc les services d'intégrité des données, d'authentification de l'origine des données et de non-répudiation.

La façon la plus simple de signer un message est d'utiliser la cryptographie asymétrique pour le chiffrer en utilisant sa clé privée : seul le possesseur de cette clé peut générer la signature et toute personne ayant accès à la clé publique correspondante peut la vérifier. Mais cette méthode est très lente et en pratique elle n'est que peu utilisée.

La méthode réellement utilisée repose non pas sur le chiffrement du message lui-même mais sur l'empreinte (empreinte issue d'une fonction de hachage comme MD5 par exemple) de celui-ci. En effet, cette méthode est beaucoup plus rapide du fait de la quantité réduite des données à chiffrer.

Une signature numérique est plus sûre qu'une signature papier car la signature change à chaque message. Elle est de ce fait inimitable (sans la connaissance de la clé secrète bien entendue). [4]

### 4.2.3.5 Certificats numériques

Lors de l'utilisation des systèmes de cryptographie de clé publique, les utilisateurs doivent constamment vérifier qu'ils cryptent vers la clé du bon utilisateur, ce qui constitue un problème. Dans un environnement où le libre échange de clés via des serveurs publics est sécurisé, toute attaque menée par une personne intermédiaire, encore appelée un intercepteur, représente une menace éventuelle. Dans ce type d'attaque, une personne place une fausse clé comportant le nom et l'ID utilisateur du destinataire. Les données cryptées (et interceptées) vers le détenteur réel de cette clé erronée sont dorénavant entre de mauvaises mains.

Dans un environnement de clé publique, il est essentiel de s'assurer que la clé publique vers laquelle vous cryptez les données est celle du destinataire concerné et non une contrefaçon. Vous pouvez crypter uniquement vers les clés qui vous ont été distribuées physiquement. Supposons maintenant que vous devez échanger des informations avec des personnes que vous ne connaissez pas, comment savoir que vous êtes en possession de la bonne clé ?

Les certificats numériques ou certificats simplifient la tâche qui consiste à déterminer si une clé publique appartient réellement à son détenteur supposé.

Un certificat correspond à une référence. Il peut s'agir par exemple de votre

Permis de conduire, de votre carte de sécurité sociale ou de votre certificat de naissance. Chacun de ces éléments contient des informations vous identifiant et déclarant qu'une autre personne a confirmé votre identité. Certains certificats, tels que votre passeport, représentent une confirmation de votre identité suffisamment importante pour ne pas les perdre, de crainte qu'une autre personne ne les utilise pour usurper votre identité.

Un certificat numérique contient des données similaires à celles d'un certificat Physique. Il contient des informations associées à la clé publique d'une personne, aidant d'autres personnes à vérifier qu'une clé est authentique ou *valide*.

Les certificats numériques permettent de contrecarrer les tentatives de substitution

De la clé d'une personne par une autre.

Un certificat numérique se compose de trois éléments :

• Une clé publique.

• Des informations sur le certificat. (Informations sur l'« identité » de l'utilisateur,

T elles que son nom, son ID utilisateur, etc.)

• Une ou plusieurs signatures numériques.

#### 4.2.3.6 La cryptographie quantique

La cryptographie quantique repose sur un échange des clés secrètes par un canal quantique, en faisant appel à des photons (quantum key distribution ou QKD). La particularité de cette méthode est de reposer sur une loi physique permettant de détecter toute interception des clés, et donc de contrôler l'intégrité des données chiffrées et déchiffrées grâce à celles-ci. Un objet quantique ne peut en effet être observé sans être altéré. Une fois les clés échangées, les flux sont chiffrés par des mécanismes classiques, en faisant par exemple appel à l'algorithme AES.

Pourquoi utiliser le système de cryptographie quantique pour transmettre une clé, et non le message en lui-même ?

Pour deux raisons essentielles :

- Les bits d'informations communiqués par les mécanismes de la cryptographie quantique ne peuvent être qu'aléatoires. Ceci ne convient pas pour un message, mais convient parfaitement bien à une clé secrète, qui doit être aléatoire.
- Même si le mécanisme de la cryptographie quantique garantit que l'espionnage de la communication sera toujours détecté, il est possible que des bits d'informations soient interceptés par l'espion avant que celui-ci ne soit détecté.

  Ceci est inacceptable pour un message, mais sans importance pour une clé aléatoire qui peut être simplement jetée en cas d'interception.

## 5. La cryptanalyse

### 5.1 Définition

La cryptanalyse s'oppose, en quelque sorte, à la cryptographie.

En effet, si Déchirer consiste à retrouver le clair au moyen d'une clé, cryptanalyse c'est tenter de se passer de cette dernière et retrouver le message caché sans connaitre la clé privée de l'algorithme.

### 5.2 Types d'attaques

Il existe quatre types d'attaques générales :

#### 5.2.1 Attaque par analyse de messages chiffrés [Ciphertext only attack]

Dans le cadre de cette attaque, le cryptanalyste dispose d'une série de messages chiffrés, éventuellement chiffré avec la même clé.

Le but est de trouver le message clair pour un grand nombre de messages chiffrés, voir de déterminer la clé (ou les clés).

### 5.2.2 Attaque par message clair connu [Known plaintext attack]

Cette attaque est aussi appelée attaque par message clair/message chiffré.

Le cryptanalyste possède un ensemble de messages clairs et de messages chiffrés correspondants.

Le problème est alors de retrouver la clé à partir de ces données ou un algorithme pour déchiffrer n'importe quel autre message (probablement chiffré avec la même clé).

### 5.2.3 Attaque à texte (clair ou chiffré) choisi [Choosen plaintext ou Choosen ciphertext attack]

Le cryptanalyste peut soit choisir des messages clairs et leurs équivalents chiffrés (attaque à message claire choisi), soit choisir des messages chiffrés et leurs équivalents clairs. C'est l'une des attaques les plus dangereuses, car le cryptanalyste peut choisir des formatages particuliers de textes, permettant d'obtenir des informations supplémentaires sur la clé.

### 5.2.4 Attaque adaptative à texte (clair ou chiffré) choisi [Adaptative Choosen plaintext ou Adaptative Choosen ciphertext attack]

Cette attaque est un cas spécial du précédent, où le cryptanalyste peut fabriquer la paire (Ci+1;Mi+1) en fonction du résultat obtenu à l'étape i.

Remarque

Un algorithme est dit cassé quand il est possible de retrouver la clé en effectuant moins d'opérations qu'en utilisant la force brute.

Un algorithme cassé est "moins sûr", mais pas inutile pour autant. Il faudra veiller à son utilisation si on souhaite assurer la confidentialité des données, mais rien n'empêche de l'utiliser à d'autres fins.

## 5.3 Les différentes attaques

### 5.3.1 L'Analyse des fréquences

L'analyse de fréquences est une méthode de cryptanalyse consistant à examiner la fréquence des lettres employées dans un message chiffré. Cette méthode est fréquemment utilisée pour décoder des messages chiffrés par substitution. L'analyse fréquentielle est basée sur le fait que, certaines lettres ou combinaisons de

lettres apparaissent avec une certaine fréquence. Par exemple, en français, le E est la lettre la plus utilisée, suivie du A et du S. Inversement, le W est peu usité.

### 5.3.2 L'indice de Coïncidence

L'indice de Coïncidence (IC) est la probabilité que deux lettres choisies aléatoirement dans un texte soient identiques. Il fut inventé par William Friedman et publié en 1920, dans l'article "The Index of Coincidence and its Applications in Cryptography"

L'indice de coïncidence vise à déterminer si un texte crypté provient d'une substitution monoalphabétique ou polyalphabétique.

### 5.3.3 L'attaque par mot probable

Consiste à supposer l'existence d'un mot probable dans le message chiffré. Il est donc possible d'en déduire la clé du message si le mot choisi est correct. Ce type d'attaque a été mené contre la machine Enigma durant la Seconde Guerre mondiale.

### 5.3.4 L'attaque par force brute

Est une méthode utilisée en cryptanalyse pour trouver un mot de passe ou une clé. Il s'agit de tester, une à une, toutes les combinaisons possibles.

### 5.3.5 L'attaque par paradoxe des anniversaires

Le paradoxe des anniversaires est un résultat probabiliste qui est utilisé dans les attaques contre les fonctions de hachage. Ce paradoxe permet de donner une borne supérieure de résistance aux collisions d'une telle fonction. Cette limite est de l'ordre de la racine de la taille de la sortie, ce qui signifie que, pour un algorithme comme MD5 (empreinte sur 128 bits), trouver une collision quelconque avec 50 % de chance nécessite $2^{64}$ hachages d'entrées distinctes.

### 5.3.6 Cryptanalyse différentielle

La cryptanalyse différentielle est une analyse statistique des changements dans la structure de la méthode de chiffrement après avoir légèrement modifié les entrées. Avec un très grand nombre de perturbations, il est possible d'extraire la clé. Cette attaque date de 1990 (présentée à la conférence *Crypto 90*). Elle est due à Eli Biham et Adi Shamir.

Nous n'allons pas plus développer cette attaque, qui serait bien trop longue à décrire en détail dans ce travail.

### 5.3.7 Cryptanalyse linéaire

La cryptanalyse linéaire est une technique inventée par Mitsuru Matsui, chercheur chez Mitsubishi Electric. Elle date de 1993 et fut développée à l'origine pour casser l'algorithme de chiffrement symétrique DES. Ce type de cryptanalyse se base sur un concept antérieur à la découverte de Matsui : les expressions linéaires probabilistes. Ces dernières ont été étudiées par Henri Gilbert et Anne Tardy-Corfdir dans le cadre d'une attaque sur FEAL.

## 6. Conclusion

Dans ce chapitre, nous avons tenté de dresser l'art de la cryptographie, et son importance dans la protection des informations pendant le transfert. Ensuite, nous avons sabordé la classification des algorithmes cryptographique avec des exemples sur chaque classe d'algorithme, enfin quelques concepts sur la cryptanalyse.

Le chapitre suivant est consacré à la présentation détaillé de l'algorithme Rijndael que nous avons choisi.

# Chapitre 2 : l'algorithme Rijndael

## 1. Introduction

Après l'étude des différentes classes des algorithmes cryptographique dans le chapitre précédent. Nous prenons dans ce chapitre l'algorithme Rijndael en détail, nous expliquerons les différentes étapes de l'opération de chiffrage et déchiffrage et comment sa marche et on finit ce chapitre avec les avantages et les inconvénients de cet algorithme.

## 2. A la découverte de Rijndael

L'AES, ou Advanced Encryption Standard (Standard de Chiffrement Avancé), provient d'une demande du bureau américain des standards, le NIST, qui cherchait en 1997 à remplacer le standard précédent, nommé le DES (Data Encryption Standard). En effet, celui-ci était en place depuis 1976 et nécessitait donc une mise à niveau. En 1998, le bureau avait reçu 15 propositions d'algorithmes, qui furent réduites à 5 finalistes après une première sélection :

- MARS (IBM)
- RC6 (RSA Laboratories)
- Rijndael (J. Daemen & V. Rijmen)
- Serpent (E. Biham et al.)
- Twofish (B. Schneier et al.)

Le NIST veut réaliser les points suivants :

- Un chiffrement par bloc.
- Longueur de la clé: 128, 192, ou 256 bits.
- Longueur de bloc: 128 bits.
- La mise en œuvre possible sur des cartes à puce.
- Libre de droits.
- Sécurité.
- La mise en œuvre efficace à la fois dans le matériel et le logiciel.
- Longueur de code et de la mémoire.
- Rapidité.

S'ensuivit plusieurs années de recherche sur tous ces algorithmes pour trouver des failles et des vulnérabilités, et pendant lesquels il s'est avéré que tous les cinq offraient un niveau de sécurité tout à fait acceptable. Au final, l'algorithme Rijndael fut sélectionné, sur des critères non seulement de sécurité mais aussi de performance, d'efficacité, de flexibilité et d'applicabilité.

Rijndael a été conçu par Joan Daemen et Vincent Rijmen, deux chercheurs de la Belgique, dans le but de devenir un candidat à l'Advanced Encryption Standard (AES) du NIST.

**Figure 11 : Dr. Vincent Rijmen et Dr. Joan Daemen**

Et ce fut donc le 26 novembre 2001, dans la publication U.S. FIPS PUB 197 (FIPS 197), que Rijndael fut officiellement adopté comme étant l'algorithme de l'AES.Il deviendra **le standard de chiffrement de l'administration fédérale américaine** le 26 mai 2002, et de manière plus significative, le premier algorithme de chiffrement ouvert à tous approuvé par la National Security Agency (NSA) pour le chiffrement des informations top secret. Celui-ci est un système de chiffrement dit ''par blocs'' car les messages sont chiffrés par blocs entier de 128,192 ou 256 bits.

Il existe plusieurs versions du système utilisant des clés de 128, 192 ou 256 bits.

Trois critères principaux ont été respectés dans sa conception :

- Résistance face à toutes les attaques connues.
- simplicité dans la conception.
- rapidité du code sur la plus grande variété de plates-formes possible.

Dans ce qui suit, on essaie de réaliser l'algorithme de **« RIJNDAEL »** qui représente le standard **AES** en considérant la norme de base qui est une clé de 128 bits.

## 3. Présentation du système Rijndael

Tout d'abord un aperçu de l'algorithme est donné Rijndael. La figure 12 montre les différentes phases de l'algorithme Rijndael. Il commence par une ronde initiale suivie par un certain nombre de tours standard, il se termine par la ronde finale. Seuls quatre opérations différentes sont nécessaires de calculer ces tours et un calendrier de clé.

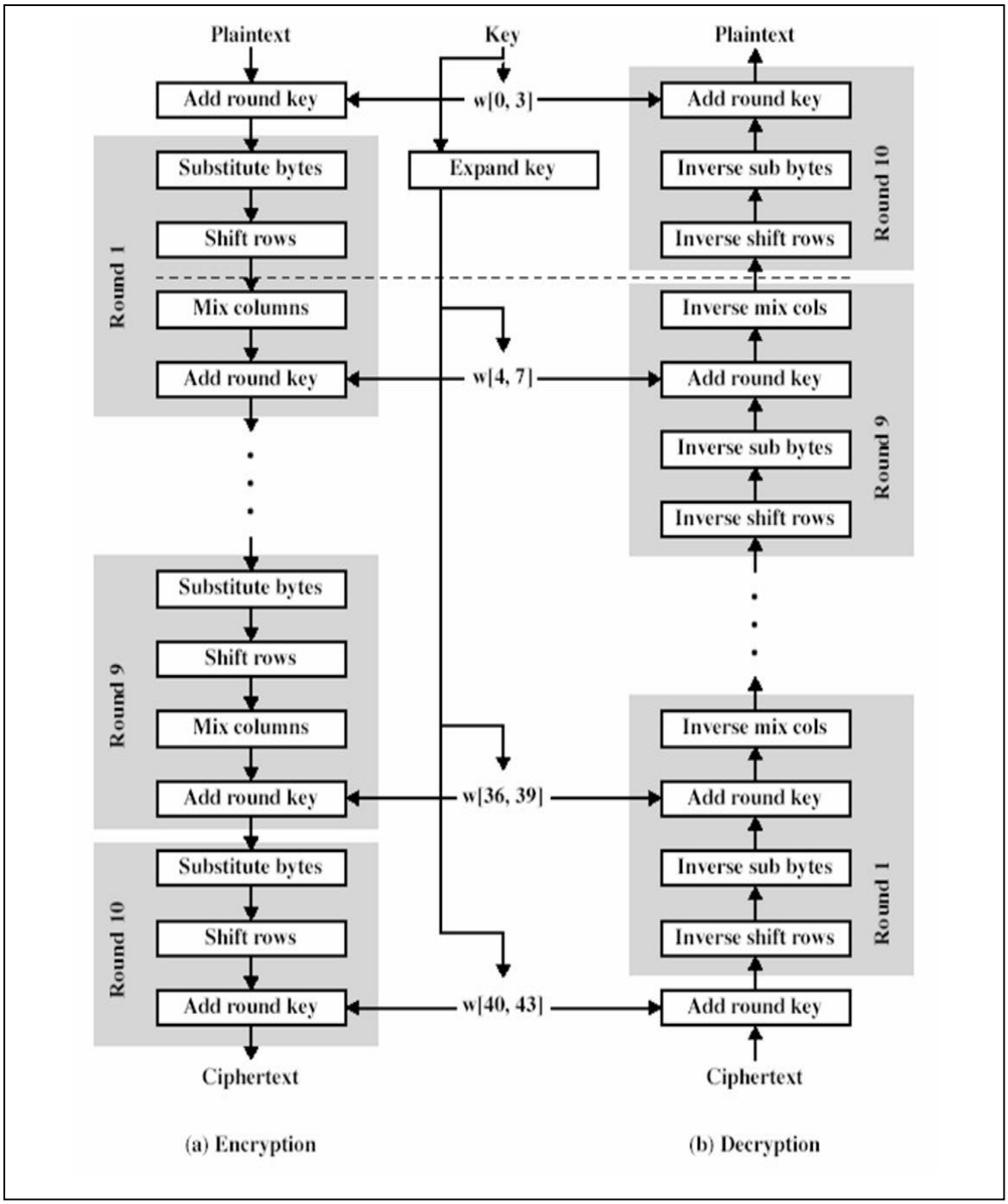



**Figure 12 : Schéma de fonctionnement de l'algorithme Rijndael.**

Il est possible d'utiliser dans Rijndael différentes longueurs de clés en fonction du niveau de sécurité ce qui est nécessaire pour l'application.

Rijndael est défini comme un bloc de chiffrement avec la clé de longueurs de 128, 192 ou 256 bits. Les possibles longueurs de blocs d'entrée sont de 128, 192 ou 256.

Pour l'algorithme Rijndael, L'algorithme AES est exactement le même que le Rijndael algorithme, mais elle ne fait que définir une longueur de bloc de 128 bits.

| | LONGUEUR DE CLE (MOT) | NOMBRE DE RONDES Nr(TOURS) |
|---|---|---|
| AES /128 | 4 | 10 |
| AES/192 | 6 | 12 |
| AES/256 | 8 | 14 |

**Tableau 4 : Numéros de rondes (un mot =32bits)**

Avec clé de 128 bit: $2^{128}$= 3.4x 1038 clés possibles.

Avec clé de 192 bit: 2192 = 6.2x 1057 clés possibles.

Avec clé de 256 bit: 2256 = 1.1x 1077 clés possibles.

Dans la description de la Rijndael le résultat chiffré intermédiaire sera appelé l'État. Notations de matrice peut être utilisée pour représenter l'état. La structure de la matrice est telle qu'il y ait toujours quatre rangées et le nombre de colonnes est variable en fonction sur le nombre de bits choisis pour la longueur de bloc et une longueur de clé.

Une clé de 128 bits par exemple est un (4,4) avec une matrice d'un octet dans chaque élément:

| | | | |
|---|---|---|---|
| $K_{0,0}$ | $K_{0,1}$ | $K_{0,2}$ | $K_{0,3}$ |
| $K_{1,0}$ | $K_{1,1}$ | $K_{1,2}$ | $K_{1,3}$ |
| $K_{2,0}$ | $K_{2,1}$ | $K_{2,2}$ | $K_{2,3}$ |
| $K_{3,0}$ | $K_{3,1}$ | $K_{3,2}$ | $K_{3,3}$ |

**Figure 13 : Matrice d'un clé de 128 bits**

Un bloc de longueur de 256 bits est représenté dans un (4,8) matrice.

Un bloc de longueur de 192 bits est représenté dans un (4,6) matrice

Le nombre de colonnes dans le bloc d'entrée est appelée le Nb, qui est égal au bloc longueur divisée par 32. Le paramètre Nk est utilisé pour désigner le nombre de colonnes dans la clé.

Il est possible de combiner toutes les longueurs de bloc avec toutes les clés de longueur différente, Par exemple, considérons l'entrée et la clé suivante:

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bloc en entrée | 32 | 43 | F6 | A8 | 88 | 5A | 30 | 8D | 31 | 31 | 98 | A2 | E0 | 37 | 07 | 34 |
| clé | 2B | 7E | 15 | 16 | 28 | AE | D2 | A6 | AB | F7 | 15 | 88 | 09 | CF | 4F | 3C |

Bloc: $\begin{bmatrix} 32 & 88 & 31 & E0 \\ 43 & 5A & 31 & 37 \\ F6 & 30 & 98 & 07 \\ A8 & 8D & A2 & 34 \end{bmatrix}$ et la clé sera: $\begin{bmatrix} 2B & 28 & AB & 09 \\ 7E & AE & F7 & CF \\ 15 & D2 & 15 & 4F \\ 16 & A6 & 88 & 3C \end{bmatrix}$

## 3.1 Algorithme de chiffrement

Pour comprendre le fonctionnement de Rijndael, il paraît judicieux de commencer par l'envisager dans son ensemble (schéma bloc). Les différentes opérations seront détaillées successivement par la suite.

La fonction de chiffrement se divise en trois : une transformation initiale avec la clé (l'étape "Add Round Key", bloc XOR clé), une série de rounds puis une transformation finale.

Les exemples illustrant les mécanismes utilisent une clé de 128 bits, par soucis de simplicité.

### 3.1.1 Mise en forme des entrées

Comme tout système cryptographique symétrique, Rijndael dispose de deux entrées, à savoir le texte clair à chiffrer (plaintext) ainsi que la clé de cryptage (key). Si la longueur de la clé est fixe (128 bits), la longueur du texte clair est variable d'une application à l'autre.

Rijndael étant un algorithme de bloc, il doit commencer par découper le texte clair en blocs de128 bits.

#### 3.1.1.1 Initial Round (Transformation initiale)

C'est la première étape et la plus simple car elle ne compte qu'une seule opération, AddRoundKey. Le texte à chiffrer est découpé en blocs de 128 bits et est disposé sous forme matricielle. Les matrices ont 4*4 éléments et chaque élément comporte 8 bits (1 octet), donc chaque bloc comporte 4*4*8=128 bits.

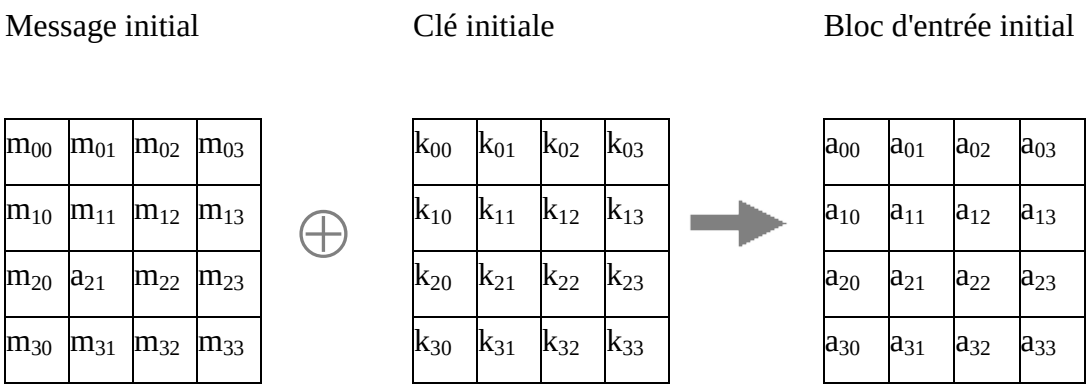



**Figure 14 : Transformation initiale**

Prenons un exemple (selon la table ASCII, §…) avec la phrase «Hello Miss Janet» : 48 65 6C 6C 6F 20 4D 69 73 73 20 4A 61 6E 65 74 qui est codée en hexadécimal. Pour rappel, 6C en hexadécimal signifie $6*16+12*16^0=6*16+12=108$ en notation décimale, avec A=10, B=11, C=12, D=13, E=14, F=15 (FF=255 en notation décimale). En notation binaire, sur 8 bits, 6C s'écrit 0110 1100. Pour simplifier, nous allons représenter les chiffres sous forme hexadécimale. Donc il suffit de représenter la suite de chiffres sous forme matricielle :

| | | | |
|---|---|---|---|
| **48** | **65** | **6C** | **6C** |
| **6F** | **20** | **4D** | **69** |
| **73** | **73** | **20** | **4A** |
| **61** | **6E** | **65** | **74** |

Qui représente le texte en clair (State). Maintenant prenons une clé codée sur 128 bits («les grandes cles», en notation hexadécimale que nous représenterons sous forme de matrice (key)

| | | | |
|---|---|---|---|
| **6C** | **75** | **23** | **20** |
| **67** | **72** | **61** | **6E** |
| **64** | **65** | **73** | **20** |
| **63** | **6C** | **65** | **73** |

Les matrices étant formées, le chiffrage peut commencer. La première étape consiste à combiner la matrice State (le bloc de texte clair) avec la clé. Cette opération s'appelle

AddRoundKey. Elle consiste à additionner modulo 2 (OU-exclusif ou XOR) chaque octet de la matrice State avec son homologue de la matrice key (clé originale).

Par exemple 48 ⊕ 6C = 0011 0000 ⊕ 0110 1100 = 0101 1100 = 5C. On obtient ainsi la nouvelle matrice state (son nom indique qu'elle désigne l'état actuel du bloc en cours de chiffrage). Elle constitue la matrice d'entrée de l'étape suivante. La nouvelle matrice state sera :

| | | | |
|---|---|---|---|
| **5c** | **…** | **…** | **…** |
| **…** | **…** | **…** | **…** |
| **…** | **…** | **…** | **…** |
| **…** | **…** | **…** | **…** |

Mais maintenant comme nous sommes paresseux et que nous ne voulons pas refaire tous les calculs, nous allons continuer avec des matrices qui ont été déjà entièrement calculées:

| | | | |
|---|---|---|---|
| **32** | **88** | **31** | **E0** |
| **43** | **5A** | **31** | **37** |
| **F6** | **30** | **98** | **07** |
| **A8** | **8D** | **A2** | **34** |

**Tableau 5 : Matrice (Texte en clair)**

| | | | |
|---|---|---|---|
| **2B** | **28** | **AB** | **09** |
| **7E** | **AE** | **F7** | **CF** |
| **15** | **D2** | **15** | **4F** |
| **16** | **A6** | **88** | **3C** |

**Tableau 6 : Matrice Key State (clé)**

Nouvelle matrice State, après avoir subi l'opération AddRoundKey :

| | | | |
|---|---|---|---|
| **19** | **A0** | **9A** | **E9** |
| **3D** | **F4** | **C6** | **F8** |
| **E3** | **E2** | **8D** | **48** |
| **BE** | **2B** | **2A** | **08** |

### 3.1.1.2 Round 1 – Round 9 (Rondes 1 à 9)

Cette partie du processus de chiffrage dépend de la taille de la clé utilisée. Comme on envisage la version 128 bits de l'algorithme, cette deuxième étape compte 9 itérations. Chacune de ces 9 itérations effectue successivement les quatre opérations détaillées ci-dessous.

### 3.1.1.2.1SubBytes

| hex | | y | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
| x | 0 | 63 | 7c | 77 | 7b | f2 | 6b | 6f | c5 | 30 | 01 | 67 | 2b | fe | d7 | ab | 76 |
| | 1 | ca | 82 | c9 | 7d | fa | 59 | 47 | f0 | ad | d4 | a2 | af | 9c | a4 | 72 | c0 |
| | 2 | b7 | fd | 93 | 26 | 36 | 3f | f7 | cc | 34 | a5 | e5 | f1 | 71 | d8 | 31 | 15 |
| | 3 | 04 | c7 | 23 | c3 | 18 | 96 | 05 | 9a | 07 | 12 | 80 | e2 | eb | 27 | b2 | 75 |
| | 4 | 09 | 83 | 2c | 1a | 1b | 6e | 5a | a0 | 52 | 3b | d6 | b3 | 29 | e3 | 2f | 84 |
| | 5 | 53 | d1 | 00 | ed | 20 | fc | b1 | 5b | 6a | cb | be | 39 | 4a | 4c | 58 | cf |
| | 6 | d0 | ef | aa | fb | 43 | 4d | 33 | 85 | 45 | f9 | 02 | 7f | 50 | 3c | 9f | a8 |
| | 7 | 51 | a3 | 40 | 8f | 92 | 9d | 38 | f5 | bc | b6 | da | 21 | 10 | ff | f3 | d2 |
| | 8 | cd | 0c | 13 | ec | 5f | 97 | 44 | 17 | c4 | a7 | 7e | 3d | 64 | 5d | 19 | 73 |
| | 9 | 60 | 81 | 4f | dc | 22 | 2a | 90 | 88 | 46 | ee | b8 | 14 | de | 5e | 0b | db |
| | a | e0 | 32 | 3a | 0a | 49 | 06 | 24 | 5c | c2 | d3 | ac | 62 | 91 | 95 | e4 | 79 |
| | b | e7 | c8 | 37 | 6d | 8d | d5 | 4e | a9 | 6c | 56 | f4 | ea | 65 | 7a | ae | 08 |
| | c | ba | 78 | 25 | 2e | 1c | a6 | b4 | c6 | e8 | dd | 74 | 1f | 4b | bd | 8b | 8a |
| | d | 70 | 3e | b5 | 66 | 48 | 03 | f6 | 0e | 61 | 35 | 57 | b9 | 86 | c1 | 1d | 9e |
| | e | e1 | f8 | 98 | 11 | 69 | d9 | 8e | 94 | 9b | 1e | 87 | e9 | ce | 55 | 28 | df |
| | f | 8c | a1 | 89 | 0d | bf | e6 | 42 | 68 | 41 | 99 | 2d | 0f | b0 | 54 | bb | 16 |

**Figure 15 : S-Box**

La substitution se fait de la manière suivante :

1. Prendre la matrice state trouvée à la sortie de la ronde initiale (initial round).
2. Pour chaque élément de cette matrice, procéder comme suit :

❖ Le premier caractère hexadécimal de l'élément indique une ligne de la S-Box tandis que le deuxième indique une colonne.

❖ L'octet se trouvant à l'intersection ligne-colonne dans la S-Box est celui qui doit être substitué à celui de la matrice State.

Exemple :

On a 19 comme premier élément donc 1 représente la ligne et 9 la colonne. La valeur de remplacement du premier élément sera donc d4. Le deuxième élément vaut A0 et sera remplacé par E0. Ainsi la matrice state issue de cette opération (SubBytes) est la suivante:

| | | | |
|---|---|---|---|
| **D4** | **E0** | **B8** | **$1^E$** |
| **27** | **BF** | **B4** | **41** |
| **11** | **98** | **5d** | **52** |
| **AE** | **F1** | **E5** | **30** |

### 3.1.1.2.2 ShiftRows

Cette opération consiste à décaler des lignes dans la matrice State. En ce sens, elle garantit le principe de diffusion de Shannon. De faibles changements dans le texte clair impliquent de grands changements dans le texte chiffré. Les décalages ne modifient pas les valeurs des bytes, mais changent leur ordre.

Les décalages se font comme suit :

• La première ligne n'est pas décalée.

• La deuxième ligne est décalée de 1 octet vers la gauche.

• La troisième ligne est décalée de 2 octets vers la gauche.

• La quatrième ligne est décalée de 3 octets vers la gauche.

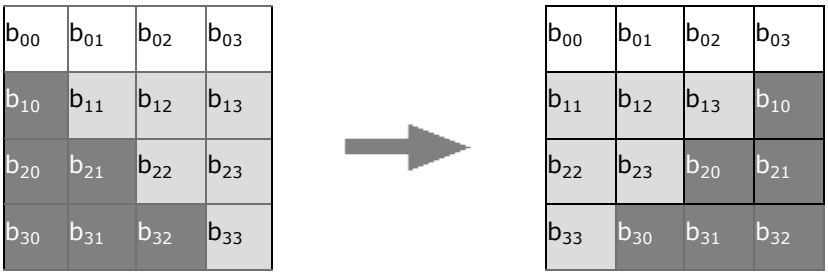



**Figure 16 : Décalage des lignes (ShiftRows)**

La matrice State issue de cette opération (ShiftRows) est la suivante :

| | | | |
|---|---|---|---|
| **D4** | **E0** | **B8** | **1E** |
| **BF** | **B4** | **41** | **27** |
| **5D** | **52** | **11** | **98** |
| **30** | **AE** | **F1** | **E5** |

### 3.1.1.2.3 MixColumns

Cette opération consiste à multiplier une matrice constante avec la matrice State :

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{D4} & \mathbf{E0} & \mathbf{B8} & \mathbf{1E} \\ \mathbf{BF} & \mathbf{B4} & \mathbf{41} & \mathbf{27} \\ \mathbf{5D} & \mathbf{52} & \mathbf{11} & \mathbf{98} \\ \mathbf{30} & \mathbf{AE} & \mathbf{F1} & \mathbf{E5} \end{pmatrix} = \begin{pmatrix} ? & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \end{pmatrix}$$

Pour obtenir le premier résultat (ligne1, colonne1), on procède comme suit :

Result [1,1] = 02•D4 ⊕ 03•BF ⊕ 01•5D ⊕ 01•30 = ?

La particularité de ce calcul est que les opérations d'addition et de multiplication se font dans le corps de Galois GF(28). Si les additions se réalisent par un simple OU-exclusif (XOR,⊕), les multiplications sont un peu plus compliquées. Néanmoins il existe une technique assez simple pour les calculer, dont l'implémentation est aisée. Remarquez que nous n'avons que des multiplications par **01** (le résultat reste inchangé dans ce cas), par 02 et par 03 (éléments de la matrice constante). [9]

**Multiplication par 02**

Considérons un polynôme dans GF(28) qui a la forme suivante :

$F(x) = b_7 x^7 + b_6 x^6 + b_5 x^5 + b_4 x^4 + b_3 x^3 + b_2 x^2 + b_1 x^1 + b_0 x^0$

Si nous multiplions ce polynôme par x :

$x \times F(x) = (b_7 x^8 + b_6 x^7 + b_5 x^6 + b_4 x^5 + b_3 x^4 + b_2 x^3 + b_1 x^2 + b_0 x^1) \bmod (m(x))$

Avec $m(x) = x^8 + x^4 + x^3 + x^1$

- Si $b_7 = 0$ alors le résultat est un polynôme de degré moindre que 8 et aucun calcul compliqué n'est nécessaire : $(b_6 b_5 b_4 b_3 b_2 b_1 0)$

- Si = $b_7 = 1$ alors (sans démonstration),

$x \times F(x) = (b_7 x^8 + b_6 x^7 + b_5 x^6 + b_4 x^5 + b_3 x^4 + b_2 x^3 + b_1 x^2 + b_0 x^1) + (x^4 + x^3 + x + 1)$

Ce qui signifie alors que le résultat sera : $(b_6 b_5 b_4 b_3 b_2 b_1 0) \oplus (00011011)$

Nous remarquons que la multiplication par x (ou '02' en hexadécimal) revient à faire un décalage de 1 bit vers la gauche, suivi d'un XOR avec (00011011) si $b_7=1$.

En appliquant ceci, reprenons nos calculs : 02•D4 = ?

Nous savons que D4 (Hexa)=212 (décimal) = 1101 0100 donc $b_7=1$, le résultat sera donc $(b_6 b_5 b_4 b_3 b_2 b_1 0) \oplus (00011011)$ = 10101000 + 00011011 = 10110011

**Multiplication par 03**

La multiplication par 03 peut être vue comme une multiplication par 02 suivie d'uneaddition modulo 2 du nombre multiplié. On peut illustrer ceci par un petit exemple en notation décimale (N=6) : 3N = 2N+N = 2•6+6 = 12+6 = 18

Voyons maintenant un exemple binaire :

N = 00101110

2N = 01011100

3N = 2N ⊕ N = 01011100 ⊕ 00101110 = 01110010

**Résultat « Result [1,1]»**

Il est maintenant possible de calculer

Result [1, 1] = 02•D4 ⊕ 03•BF ⊕ 01•5D ⊕ 01•30

02•D4 = 10110011

03•BF = 11011010

Car

1•BF = 10111111

2•BF = 01111110 ⊕ 00011011 = 01100101

3•BF = 2•BF ⊕ BF = 01100101 ⊕ 10111111 = 11011010

1•5D = 01011101

1•30 = 00110000

Finalement

Result [1,1] = 10110011 ⊕ 11011010 ⊕ 01011101 (00110000 = 00000100 =04 (hexa)

L'opération MixColumns est particulièrement laborieuse puisque les 15 autres éléments de la matrice doivent tous être calculés de cette manière ! Ceci est bien trop long pour être détaillé. Voici donc le résultat final : la matrice Result (cette matrice devient naturellement la matrice State de la prochaine opération) :

| | | | |
|---|---|---|---|
| **04** | **E0** | **48** | **28** |
| **66** | **CB** | **F8** | **06** |
| **81** | **19** | **D3** | **26** |
| **E5** | **9A** | **7A** | **4C** |

L'opération MixColumns est particulièrement laborieuse puisque les 15 autres éléments de la matrice doivent tous être calculés de cette manière ! Ceci est bien trop long pour être détaillé. Voici donc le résultat final : la matrice Result (cette matrice deient naturellement la matrice State de la prochaine opération) :

#### 3.1.1.2.4 AddRoundKey

Lors du processus de chiffrage, Rijndael transforme la clé (matrice Key). A chaque itération n(Round), une matrice Key différente est utilisée (RoundN Key). Ceci permet d'éliminer les attaques liées à la clé (Biham) en faisant disparaître la symétrie. Pour obtenir les 10 nouvelles clés nécessaires, Rijndael procède à une opération appelée Key Scheduling ou Key Expansion (cette opération sera expliquée au point suivant).

Même si la clé change à chaque ronde, l'opération AddRoundKey reste simple puisqu'elle consiste, comme celle de la première étape, à additionner modulo 2 (XOR) la matrice State et la matrice Key de la ronde en cours (RoundN Key).

AddRoundKey est le simple XOR bit à bit de la matrice d'état avec une matrice de 4*4 Octets fabriquée à partir de la clé de chiffrement pour la ronde en cours.

- Si le nombre de rondes est Nr il faut donc Nr+1 telles matrices, l'algorithme commençant par une addition de clé initiale (Rappel : Nr dépend de la taille de la clé : 128, 192 ou 256)
- La fabrication de ces Nr+1 clés est l'algorithme d'expansion des clés décrit ci-après

### 3.1.1.3 Final round

Cette étape est quasiment identique à l'un des neuf rondes de la deuxième étape. La seule différence est que, dans cette dernière ronde, l'opération MixColumns n'est pas effectuée.

## 3.1.2 Extension de la clé – Key Expansion

Il s'agit ici de voir comment Rijndael opère pour déduire les 10 clés secondaires dont il a besoin pour chiffrer le texte.

**Matrice Key (clé)** ⇨

| | | | |
|---|---|---|---|
| **2B** | **28** | **AB** | **09** |
| **7E** | **AE** | **F7** | **CF** |
| **15** | **D2** | **15** | **44** |
| **16** | **A6** | **88** | **3C** |

On considère encore une matrice Rcon (Round constant word array) construite de la façon suivante : Rcon[j]= (RC[j], 0, 0, 0) avec RC [1]=1, RC[j]=2.RC [j-1] en considérant la multiplication dans le corps de Galois GF(28). Les valeurs RC[j] en hexadécimal sont les suivantes :

| **j** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| **RC[j]** | **01** | **02** | **04** | **08** | **10** | **20** | **40** | **80** | **1b** | **36** |

Ainsi la matrice Rcon vaut :

| **01** | **02** | **04** | **08** | **10** | **20** | **40** | **80** | **1b** | **36** |
|---|---|---|---|---|---|---|---|---|---|
| **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |

Les quatre vecteurs composant la matrice Key sont appelés Words (ce sont des mots de 32 bits). On commence par calculer le cinquième vecteur ($W_i$) :

| $W_{i-4}$ | $W_{i-3}$ | $W_{i-2}$ | $W_{i-1}$ | $W_i$ |
|---|---|---|---|---|
| 2B | 28 | AB | 09 | |
| 7E | AE | F7 | CF | |
| 15 | D2 | 15 | 4F | |
| 16 | A6 | 88 | 3C | |

On prend le vecteur Wi-1 auquel on applique une opération appelée RotWord qui consiste en un simple décalage des quatre bytes du vecteur vers le haut.

| **$W_{i-1}$** | **RotWord($W_{i-1}$)** |
|---|---|
| 09 | CF |
| CF | 4F |
| 4F | 3C |
| 3C | 09 |

Au résultat on applique encore l'opération SubBytes (avec le S-Box):

| **RotWord($W_{i-1}$)** | **Subbyte(RotWord($W_{i-1}$))** |
|---|---|
| CF | 8A |
| 4F | 84 |
| 3C | EB |
| 09 | 01 |

Le vecteur obtenu doit encore être additionné (mod 2) avec le vecteur Wi-4 ainsi que le premier vecteur de la matrice Rcon :

Wi- 4⊕Rcon(colonne1) ⊕ SubBytes{RotWord(Wi-1)} :

$$\begin{pmatrix} 2b \\ 7e \\ 15 \\ 16 \end{pmatrix} \oplus \begin{pmatrix} 01 \\ 00 \\ 00 \\ 00 \end{pmatrix} \oplus \begin{pmatrix} 8a \\ 84 \\ eb \\ 01 \end{pmatrix} = \begin{pmatrix} a0 \\ fa \\ fe \\ 17 \end{pmatrix}$$

Il constitue le premier des quatre vecteurs de la deuxième clé :

| $W_{i-4}$ | $W_{i-3}$ | $W_{i-2}$ | $W_{i-1}$ | **$W_i$** |
|---|---|---|---|---|
| 2B | 28 | AB | 09 | **a0** |
| 7E | AE | F7 | CF | **fa** |
| 15 | D2 | 15 | 4F | **fe** |
| 16 | A6 | 88 | 3C | **17** |

Le deuxième vecteur de la deuxième clé s'obtient plus simplement que le premier. En effet, il est donné par $W_i = W_{i-4} \oplus W_{i-1}$ :

| $W_{i-5}$ | $W_{i-4}$ | $W_{i-3}$ | $W_{i-2}$ | **$W_{i-1}$** | **$W_i$** |
|---|---|---|---|---|---|
| 2B | 28 | AB | 09 | **a0** | **88** |
| 7E | AE | F7 | CF | **fa** | **54** |
| 15 | D2 | 15 | 4F | **fe** | **2c** |
| 16 | A6 | 88 | 3C | **17** | **B1** |

Le troisième vecteur de la deuxième clé s'obtient de la même manière que le deuxième. On a donc à nouveau $W_i = W_{i-4} \oplus W_{i-1}$ :

| $W_{i-6}$ | $W_{i-5}$ | $W_{i-4}$ | $W_{i-3}$ | $\mathbf{W_{i-2}}$ | $\mathbf{W_{i-1}}$ | $\mathbf{W_i}$ |
|---|---|---|---|---|---|---|
| 2B | 28 | AB | 09 | **a0** | **88** | **23** |
| 7E | AE | F7 | CF | **fa** | **54** | **A3** |
| 15 | D2 | 15 | 4F | **fe** | **2c** | **39** |
| 16 | A6 | 88 | 3C | **17** | **B1** | **39** |

Le dernier vecteur de la deuxième clé s'obtient de la même manière que les deuxième et troisième vecteurs, Wi = Wi-4 ⊕ Wi-1:

| $W_{i-7}$ | $W_{i-6}$ | $W_{i-5}$ | $W_{i-4}$ | $\mathbf{W_{i-3}}$ | $\mathbf{W_{i-2}}$ | $\mathbf{W_{i-1}}$ | $\mathbf{W_i}$ |
|---|---|---|---|---|---|---|---|
| 2B | 28 | AB | 09 | **a0** | **88** | **23** | **2A** |
| 7E | AE | F7 | CF | **fa** | **54** | **A3** | **6C** |
| 15 | D2 | 15 | 4F | **fe** | **2c** | **39** | **76** |
| 16 | A6 | 88 | 36 | **17** | **B1** | **39** | **05** |

Les quatre vecteurs de la deuxième clé sont maintenant définis. La matrice Key a doublé de taille. Elle contient pour l'instant les clés des rondes 0 et 1. Les vecteurs Wi-7 à Wi-4 constituent la clé de la ronde 0 et les vecteurs Wi-3 à Wi ceux de la clé de la ronde 1. Il reste à faire 9 fois l'intégralité de cette démarche d'extension de la clé pour trouver les 9 autres clés secondaires. [9]

## 3.2 Algorithme de Déchiffrement

Toutes les opérations réalisées lors du chiffrement sont réversibles. Le générateur de clés de ronde fonctionne exactement de la même manière. Il suffira de reprendre l'algorithme en sens inverse, avec les fonctions inverses.

- Les additions modulo 2, ou ⊕, effectuées lors de l'opération AddRoundKey sont réversibles (En effet, (A ⊕ B) ⊕ B = A).
- L'opération SubBytes est inversée en utilisant la table S-Box inverse (Inverse S-Box). Si par exemple la S-Box indique le byte F7 (ligne 2, colonne 6), alors la Inverse S-Box restituera le byte 26 (ligne F, colonne 7).
- Les décalages de l'opération ShiftRows sont inversés, c'est-à-dire effectués vers la droite.

- La multiplication matricielle de l'opération MixColumns nécessite une inversion de la matrice (Qui rend le déchiffrage plus lent que le chiffrage). Une fois la matrice inverse obtenue, la manipulation est la même que pour l'opération MixColumns faite lors du chiffrage.

  En conclusion on peut dire que l'utilisation de la S-Box constitue une réelle difficulté pour les cryptanalystes. L'opération MixColumns combinée avec ShiftRows fait que, après les nombreuses rondes, tous les bits de sortie dépendent de tous les bits d'entrée. Ceci rend la cryptanalyse difficile.

  L'utilisation des clés secondaires construites par extension de la clé originale, quant à elle, complique les attaques liées à la clé en cassant les symétries. Rijndael est un algorithme sûr et va probablement le demeurer encore une vingtaine d'années. [9]

## 4. Les avantages des algorithmes Rijndael

### 4.1 Flexibilité

Rijndael a une variété de bloc et tailles de clés qui peuvent être appliquées sur presque n'importe quel système. Contrairement à de nombreux algorithmes, les programmeurs peuvent modifier l'ordre d'un algorithme Rijndael pour s'adapter à la situation. Les blocs peuvent être retirés à l'algorithme pour accueillir les petits systèmes et ajoutés à des plates-formes pour s'adapter à de grands programmes.

### 4.2 Sécurité

Avoir le soutien d'une structure algébrique riche permet à Rijndael pour être plus sûr que l'algorithme moyenne. Sous algorithmes Rijndael, les individus peuvent accéder rapidement et facilement les éléments du programme. Cela permet à une personne de détecter et de résoudre les failles de sécurité car ils se produisent au lieu de jours plus tard. Revoir l'utilisation des bits et des analyses complètes du système avant la détection de virus n'est pas nécessaire de Rijndael. L'algorithme est particulièrement utile dans la défense contre le pouvoir et le moment des attaques qui ont des effets néfastes sur les systèmes informatiques**.**

### 4.3 Mémoire

Bien qu'il soit souple et défend contre les attaques, Rijndael ne nécessite pas beaucoup de mémoire pour fonctionner. Dans de nombreux cas, 128 bits est la quantité maximale de mémoire nécessaire pour l'algorithme de travailler. Avec sa fonction de mémoire, Rijndael est le choix idéal pour les programmes de matériel et de logiciel.

## 4.4 L'intérêt du public

Rijndael a une fonction de cryptographie à clé publique. Cette caractéristique permet d'accroître la sécurité et la commodité pour l'utilisateur. Au lieu d'imprimer des documents et des signatures demandant, cryptographie à clé publique permet à l'utilisateur de demander des signatures numériques des partenaires d'affaires. Cela permet d'économiser essentiellement temps et argent tout en permettant une meilleure tenue des dossiers. En outre, les personnes qui choisissent de signer numériquement des documents n'ont pas à s'inquiéter au sujet du vol d'identité, comme l'algorithme Rijndael est difficile à infiltrer.

– Le plus rapide en implantation matérielle.

– Proche des plus rapides en implantation logicielle.

## 5. Inconvénients

La gestion des clés des cryptosystèmes symétriques est assez problématique. Examinons cela sur un exemple. Supposons qu'un émetteur et un récepteur désirent communiquer de façon confidentielle en utilisant l'algorithme Rijndael. ils doivent donc pouvoir communiquer sans que un autre personne intercepter leur message.

Il faut donc trouver un procédé permettant de transmettre la clé $K_{ER}$ sans qu'un autre en prenne connaissance. La solution la plus simple serait que l'Autorité de Certification convoque individuellement l'émetteur et le récepteur pour communiquer les clés. Dans notre monde actuel, cette solution n'est pas concevable. Les solutions utilisées à ce jour utilisent la cryptographie à clés publiques.

## 6. Conclusion

Dans ce chapitre, nous passons en revue et expliquer le fonctionnement de l'algorithme Rijndael en profondeur en s'appuyant sur de nombreux exemples de ce en plus, nous avons abordé la cryptanalyse et les techniques les plus importants utilisés dans cette opération. Enfin, nous avons expliqué les avantages et les inconvénients de Rijndael. Le chapitre suivant est consacré à l'implémentation de l'algorithme.

# Chapitre 3 : implémentation

## 1. Introduction

Au cours de ce chapitre, nous allons faire une entrer au langage java qui représente l'enivrement qui nous le choisis pour l'implémentation de notre application sous l'éditeur de netbeans, ainsi que l'interface graphique de l'application et les résultats obtenus.

## 2. Le concept Java

Java est un langage inventé par les développeurs de Sun. Si les bases de ce langage ont été conçues en 1990, Java reste encore aujourd'hui dans une phase de développement.

Selon les développeurs de Sun, Java (qui signifie café en argot américain) est un langage : simple, orienté-objet, distribué, interprété, robuste sécurisé, neutre vis à vis de l'architecture, portable, à haute performance, multi-threaded et dynamique. Mis à part l'aspect de haute performance qui est TRES loin d'être vérifié, Java dispose bien des fonctions énumérées.

## 3. Comment ca fonctionne en gros ?

Le programmeur écrit un code qui est transformé par le compilateur Java en une série d'instructions qui ressemble à du langage machine (pseudo-code) mais que la machine est incapable d'exécuter. On fournit ce pseudo-code à un programme spécial (un interpréteur) qui exécute l'une après l'autre les instructions du pseudo-code. L'interpréteur peut être un navigateur Web (comme Netscape Navigator) ou un programme indépendant plus ou moins caché dans le système d'exploitation de l'ordinateur. Le pseudo-code utilise la mémoire de la machine, mais un mécanisme (le ramasse-miettes) récupère la mémoire dès que possible ce qui permet ainsi à d'autres programmes de tourner. Le pseudo-code peut faire appel à des bibliothèques de classes qui sont tout simplement des programmes écrits par d'autres programmeurs. Ces bibliothèques de classes permettent au programmeur d'éviter d'avoir à réécrire sans cesse des fonctionnalités classiques. Par exemple, les bibliothèques de classes contiennent des classes permettant de dessiner un bouton, d'ouvrir une connexion avec le réseau, de jouer de la musique... Ces bibliothèques de classes ne se trouvent pas forcement sur la machine où s'exécute le pseudo-code. L'interpréteur va alors les chercher où il faut sur le réseau. En fait, vous avez peut être déjà utilisé Java sans même le savoir, simplement en allant surfer sur le Web. Les programmes exécutés par

les navigateurs sont appelés applets contrairement aux programmes autonomes appelés comme il se doit applications.

## 4. Notion d'objets

Java est un langage orienté objet. Cela signifie qu'il réalise un ensemble de concepts lui permettant de dépasser le cadre du simple langage de programmation structuré (Pascal, C...).

## 5. Instructions de base

De plus, Java, par sa vocation orientée objet, introduit un ensemble d'instructions essentielles permettant :

- ✓ La définition de classes
- ✓ La définition de méthodes
- ✓ La définition d'un constructeur (et destructeur) d'une instance de classe
- ✓ L'appel d'une méthode
- ✓ L'appel d'une instance de classe

## 6. Applications et applets

Java peut être employée dans deux objectifs différents :

- Soit pour produire des applications classiques (standalone) fonctionnant comme n'importe quel logiciel classique de votre ordinateur.
- Soit pour construire des applications distribuées (applets) fonctionnant à travers le Web et interfacées par un navigateur grâce à l'emploi judicieux de code HTML.

## 7. Avantages

Java présente des avantages importants sur les autres langues et les environnements qui le rendent approprié à presque n'importe quelle tâche de programmation.
Les avantages de Java sont les suivants:

- ➢ Java est facile à apprendre.
- ➢ Java a été conçue pour être facile à utiliser et il est donc facile à écrire, compiler, déboguer et apprendre que les autres langages de programmation.
- ➢ Java est orienté objet.

Cela vous permet de créer des programmes modulaires et code réutilisable.

- Java est indépendante de la plateforme.
- L'un des avantages les plus importants de Java est sa capacité de se déplacer facilement d'un système informatique à un autre. La capacité à exécuter le même programme sur de nombreux systèmes différents est essentielle pour le logiciel World Wide Web, et Java succède à cela en étant indépendant de la plateforme à la fois la source et le niveau binaire.
- Java est distribuée.
- Java est conçue pour rendre l'informatique répartie facile avec la fonctionnalité de réseau qui est intrinsèquement intégrées. Écriture de programmes de réseau en Java, c'est comme envoyer et recevoir des données vers et à partir d'un fichier.
- Java est sécurisée.
- Java considère la sécurité dans le cadre de son design. Le langage Java, le compilateur, interprète, et l'environnement d'exécution ont chacun été développés avec la sécurité à l'esprit.
- Java est robuste.

Consistant: la fiabilité. Java met beaucoup d'accent sur la vérification rapide pour d'éventuelles erreurs, que des compilateurs Java sont en mesure de détecter de nombreux problèmes qui seraient d'abord montrer au cours de délai d'exécution dans d'autres langues.

- Java est multi-thread.

Multithread est la possibilité pour un programme à exécuter plusieurs tâches simultanément dans un programme. En Java, la programmation multithread a été intégrée harmonieusement, tandis que dans d'autres langues, système d'exploitation des procédures spécifiques doivent être remises en vue de permettre le multithreading.

En raison de la robustesse de Java, la facilité d'utilisation, des fonctions plate-forme et les caractéristiques de sécurité, il est devenu une langue de choix pour la fourniture de solutions Internet du monde entier.

## 8. JDK

JDK est un logiciel (ensemble des outils officiels) édité par SUN pour le développement nécessaires à la création et à l'exécution de programmes Java. Ils contiennent notamment les compilateurs javac, l'interpréteur java, l'outil d'archivage jar, et plein d'autre choses. Ils sont en générale fourni avec le code source de l'API et pas mal de fichiers d'aides. Dans ce travail nous avons utilisée JDK 7.4].

## 9. Editeur

Il existe plusieurs EDI sur le marché comme par exemple, Jbuilder, Jcreator , Netbeans ou encore Eclipse. Nous avons choisi Netbeans pour notre développement dans le langage Java.

Netbeans est un environnement de développement intégré (EDI). C'est une application qui propose dans le même système de fenêtrage, des outils qui facilite le travail au développeur.

Par exemple, cet environnement propose :

Un éditeur de texte avec coloration syntaxique des éléments clés du langage.

De la même façon, il propose l'auto-complétion par les choix d'écriture disponible de l'instruction en cours d'écriture.

Il propose aussi une fenêtre de compilation ou s'affiche les éventuelles erreurs de compilation et une fenêtre d'exécution qui permet de visualiser les résultats de l'application.

Netbeans a l'avantage d'être distribué en open source et être entièrement gratuit, pour cette raison et pour sa facilité d'installation et d'usage sous Windows et les autres systèmes comme Linux, nous l'avons choisi comme EDI durant l'implémentation de notre travail.

## 10. NetBeans

**C'**est un environnement de développement intégré (EDI) pour Java, placé en open source par Sun en juin 2000 sous licence CDDL et GPLv2 (Common Development and Distribution License). En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, XML, Ruby, PHP et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web).

## 10 .1 Environnement de base

L'environnement de base comprend les fonctions générales suivantes:

- configuration et gestion de l'interface graphique des utilisateurs,
- support de différents langages de programmation,
- traitement du code source (édition, navigation, formatage, inspection..),
- fonctions d'import/export depuis et vers d'autres IDE, tels qu'Eclipse ou JBuilder,
- accès et gestion de bases de données, serveurs Web, ressources partagées,
- gestion de tâches (à faire, suivi …),
- documentation intégrée. [10]

## 11. Description de l'application

Notre application contient:

### 11.1 Les classes

- RijndaelTxt : cette classe est une méthode pour crypter et décrypter l'image.
- RijndaelImg : cette classe est une méthode pour crypter et décrypter un texte.

### 11.2 L'interface graphique

Notre logiciel possède une interface graphique qui facilite son utilisation, représentée par la figure suivant :


**Figure 17: Interface graphique principale**

## 12. Les interfaces graphiques Principales

### 12.1 Barre de Menu

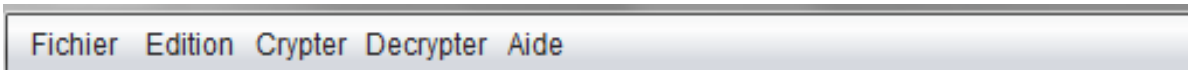
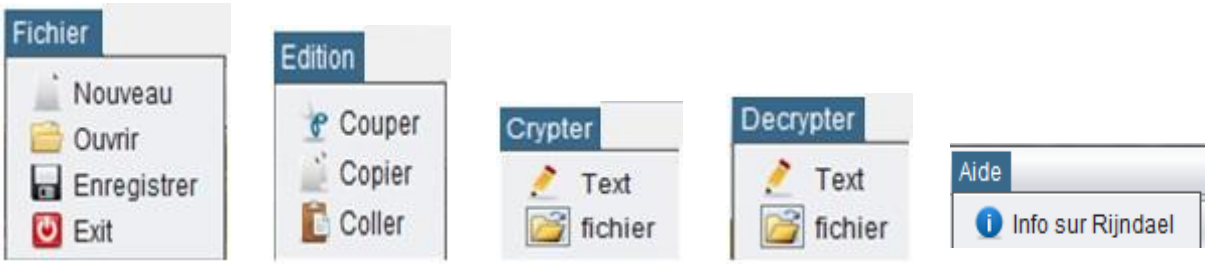



**Figure 18**: **Barre de Menu**

Cette bar est contient cinq menu comme suivant :




### 12.2 Buttons raccourcis


**Figure 19 : Buttons raccourcis**

## 13. Exemple d'application sur un texte

### 13.1 Le chiffrement

1. Saisir le texte clair :

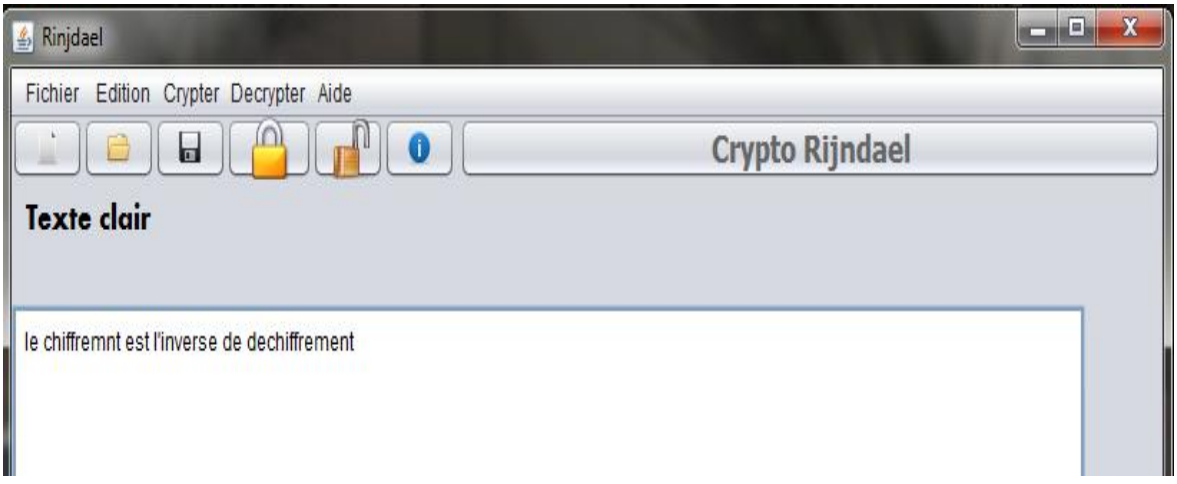



**Figure 20 : Texte Clair**

**2.** choisir le type de cryptage (Crypter Texte).

**3.** entrer la clé de cryptage (16 caracteres).

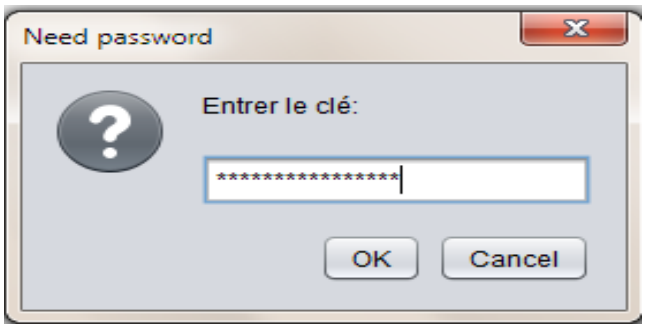



**Figure 21 : Entrer la clé de cryptage**

**4.** Le résultat de cryptage de texte est :

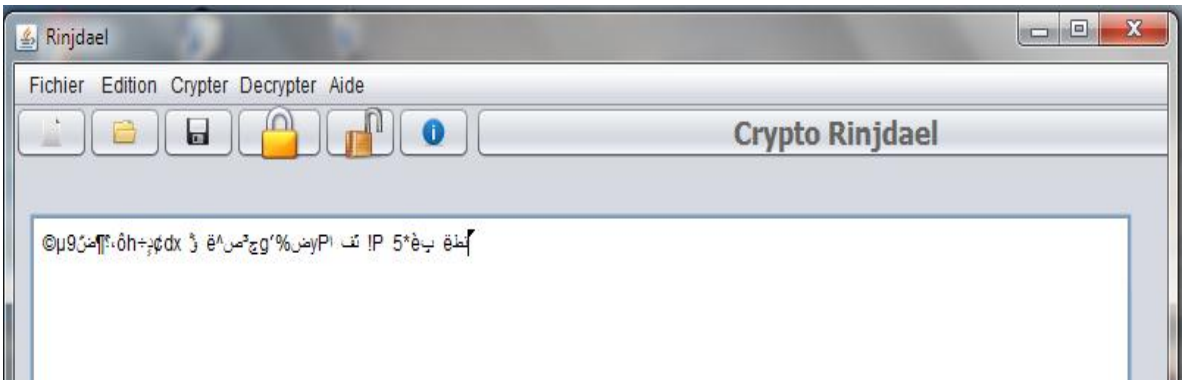



**Figure 22** : **Texte crypté**

## 13.2 Le déchiffrement

**1.** Pour le décryptage nous suivons les mêmes étapes précédentes mais nous choisissons la Command décrypter au lieu de crypter.


**Figure 23** : **Entrer la clé de décryptage**

**2.** Le résultat de décryptage de texte est :

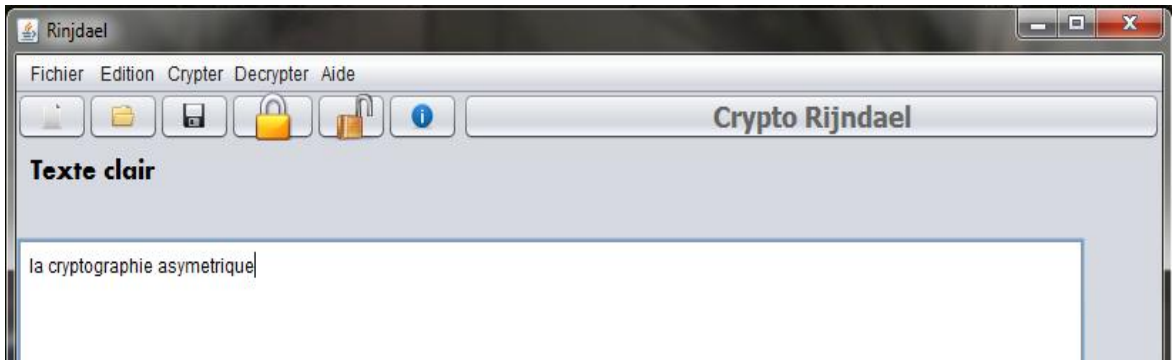



**Figure 24 : Texte clair après le décryptage**

## 14. Exemple d'application sur l'image

### 14.1 Le chiffrement

**1.** choisir l'image claire :

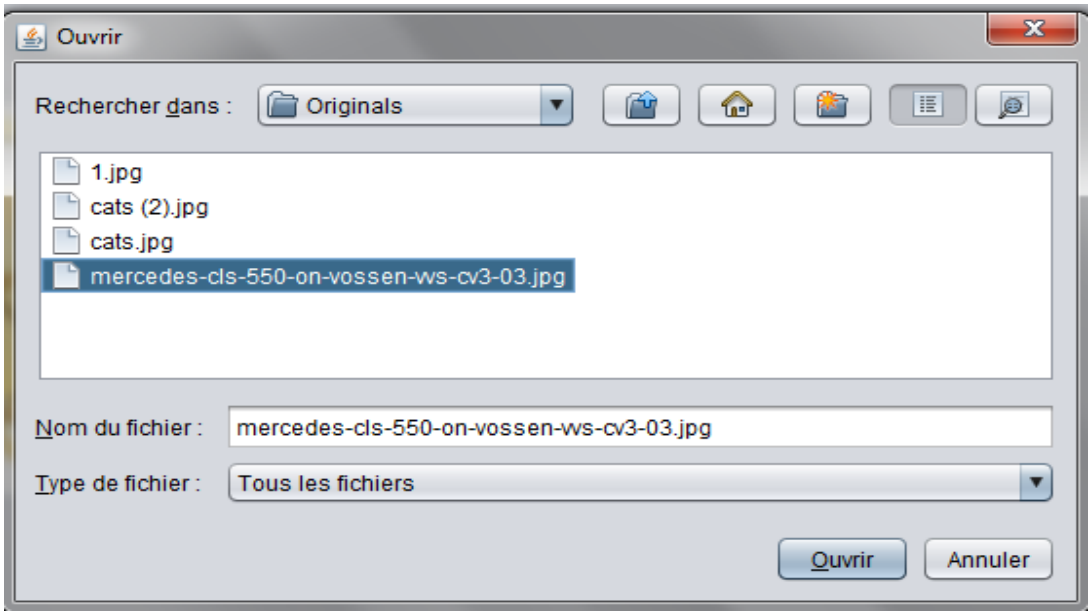



**Figure 25** : **Fenêtre ouvrir pour choisir l'image claire**

**2.** Cliquer sur sous menu crypter pour afficher le message d'entrer la clé :

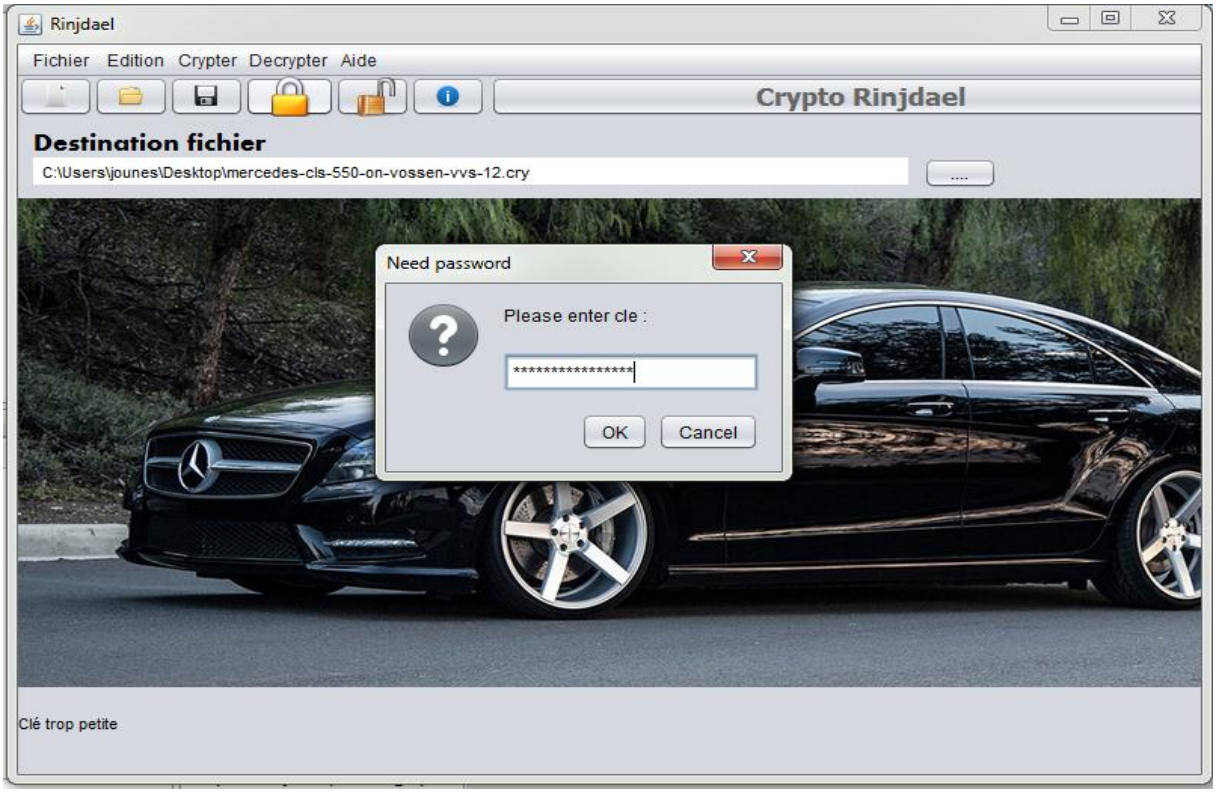



**Figure 26 : Message d'entrer la clé de cryptage**

**3.** Le résultat de cryptage de l'image est un fichier crypté :

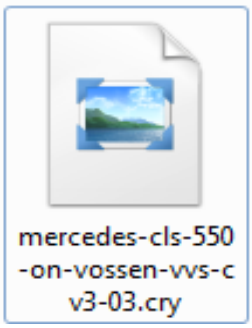



**Figure 27** : **Icône de fichier crypté**

## 14.2 Le déchiffrement

**1.** Pour le décryptage nous suivons les mêmes étapes précédentes mais nous choisissons la commande décrypter au lieu de crypter.

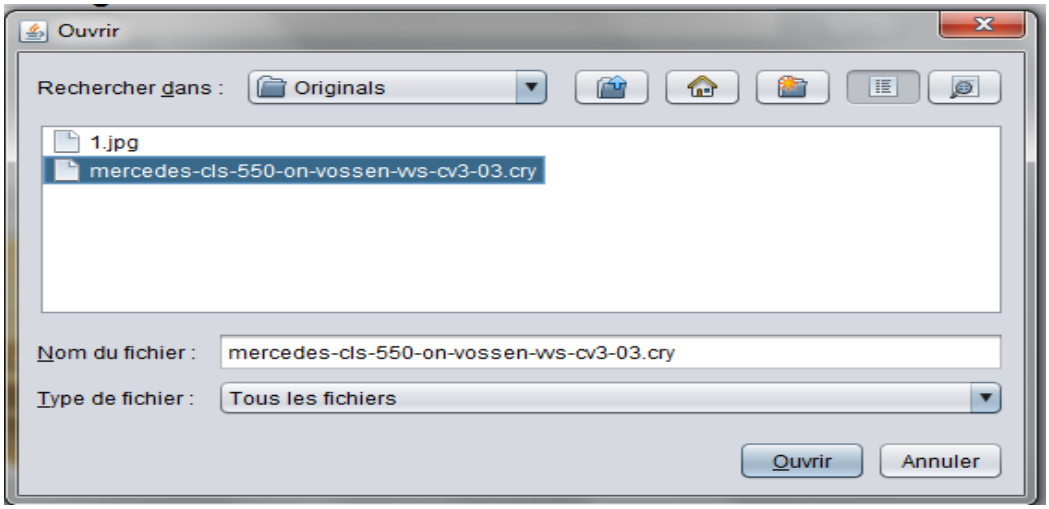



**Figure 28**: **Fenêtre ouvrir pour choisir l'image cryptée**

**2.** Après le choix de l'image crypté l'interface est formée comme suivant :

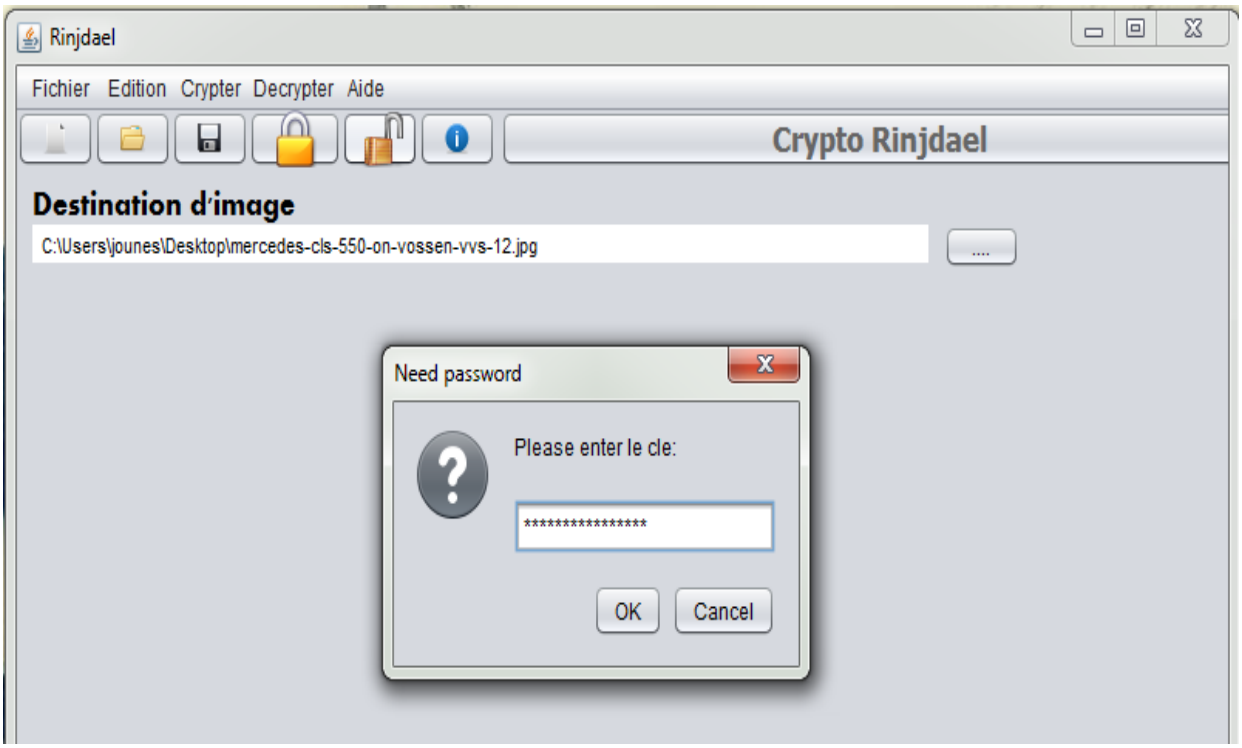



**Figure 29** : **Choisir l'image crypter et entrer la clé**

**1.** Le résultat de cryptage de l'image est une image claire :

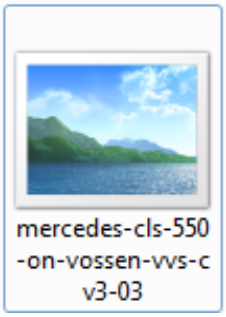



**Figure 30 : L'image claire**

## 15. Conclusion

Ce dernier chapitre était consacré pour la réalisation de l'algorithme Rijndael. Nous avons commencé par la description du langage et l'environnement de développement choisi, puis présenté l'interface graphique de notre application suivie par des exemples de cryptage et décryptage d'un texte et d'une image.

## Conclusion générale

La cryptographie est une science en perpétuelle évolution, la cryptanalyse aidant à trouver les failles d'un système pour toujours avancer. Cette évolution est importante car la cryptographie joue un grand rôle dans la sécurité internationale, tout étant aujourd'hui informatisé.

La Conception de l'Advanced Encryption Standard (AES Rijndael) a haute spécification, où la prise en compte des dispositifs qui seront utilisés, Et des logiciels, ainsi que la mémoire consommée.

En dépit de sa simplicité, mais il n'a pas fourni d'informations suffisantes pour permettre la pénétration des pirates, a évité toutes les méthodes précédentes d'attaque. Les quatre opérations utilisées dans l' Advanced Encryption Standard masquer toute relation entre le texte original et le texte chiffré , et cachent également la relation entre la clé de chiffrement et le texte chiffré , ce qui empêche les attaquants de l'aide des méthodes d'analyse des différents attaques , aussi ne fournit pas d'information sur les statistiques qui dépendent de la répétition lettres , et aussi impossible pour l'attaquant d'essayer toutes les possibilités pour l'utilisateur et la clé de la grande région de la clé .

En réalité, l'Advanced Encryption Standard (AES Rijndael) est la meilleure combinaison de simplicité et de rapidité et de protection.

Dans notre travaille nous avons essayé d'expliquer toutes les notions qui ont une relation avec la cryptographie dans le premier chapitre, dans le deuxième chapitre nous avons vus l'explication des différents étapes de chiffrement et déchiffrement de l'algorithme Rijndael, enfin une introduction à la programmation java et l'éditeur netbeans qui représente l'enivrement pour implémenter notre application ainsi que l'interface graphique de l'application et les résultats obtenus.

# Bibliographie

[1] Philippe Perret, Serge Richard : Cryptographie.

[2] Singh: Simon Singh, "Histoire des codes secrets", LC Lattès, 1999.

[3] Florent BERNARD : Cours de cryptographie.

[4] Enseignant Sandrine JULIA : techniques de cryptographie.

[5] Daniel Barsky : Cours De Cryptographie.

[6] Renaud Dumont : Cryptographie et Sécurité informatique.

[7] Mahammedi Nadjiba ,Mahdadi Houda : Les Concepts Fondamentaux De La Cryptographie Moderne ,Master Mohammedi Mahdadi.

[8] Travaux Personnels Encadrés: Rupture et continuité :

L'évolution de la cryptographie a-t-elle permise l'émergence de techniques inviolables?

[9] Stephan Robert : ELEMENTS DE CRYPTOGRAPHIE, septembre 2005.

[10] Guide Pratique EDI NetBeans.

## Autres livres

- K. Cartrysse et J.C.A. van der Lubbe: The Advanced Encryption Standard: Rijndael. Keijo Ruohonen: Mathematical Cryptology.
- Specification for the Advanced Encryption Standard (AES)", par National Institute of Standards and Technology.
- Alex Biryukov et Dmitry Khovratovich: Related-key Cryptanalysis of the Full AES-192 and AES-256?
- Harris Nover: Algebraic Cryptanalysis of AES – An overview.
- Andrey Bogdanov, Dmitry Khovratovich, et Christian Rechberger: Biclique Cryptanalysis of the Full AES.
- Nadia El Mrabet : Les concepts fondamentaux de la cryptographie.
- Pierre Loidreau : Introduction à la cryptographie.