IN/003-12/11

Université Abou Bekr Belkaid
Tlemcen Algérie

جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme d'Ingénieur d'État en Informatique

Option : Système d'information avancé

Thème

Compression des images par codage Huffman suivie par une opération de chiffrement à clé secrète

Réalisé par :

- *M^elle Hachemi Fouzia*
- *M^elle Khater Khadidja*

- *Présenté le 05 Octobre 2011 devant le jury composé de MM.*
  - *Mr .BENZIYANE.Y* *(Président)*
  - *Mr .BENAISSA .M* *(Encadreur)*
  - *Mr. BENMOUNA .Y* *(Examinateur)*
  - *Mr. BELABED .A* *(Examinateur)*

Année universitaire: 2010-2011

Table des matières

Chapitre II : Compression des images

Chapitre III: Techniques de Cryptages

Chapitre IV: les algorithmes de la cryptographie

Chapitre V: Implémentation et résultat

Liste des figures

Liste des tableaux

Liste des sigles et des abréviations

2D	images à **deux** **d**imensions
AES	**A**dvanced **E**ncryption **S**tandard
Algo	**Algo**rithme
ANSI	**A**merican **N**ational **S**tandard **I**nstitute
ARJ	**A**rchived by **R**obert K. **J**ung,
BMP	**Bitmap**
CBC	**C**ipher **B**lock **C**haining
DAO	**D**essin **A**ssisté par **O**rdinateur
DEA	**D**ata **E**ncryption **A**lgorithm
DES	**D**ata **E**ncryption **S**tandard
DPI	**D**ots **P**er **I**nch
ECB	**E**lectronic **C**ode**B**ook
Echo Doppler	**E**chographie **D**opplers
GIF	**G**raphical **I**nterchange **F**ormat
GZIP	**G**NU **Zip**
IBM	**I**nternational **B**usiness **M**achines
IDEA	**I**nternational **D**ata **E**ncryption **A**lgorithm
Img	**Image**
IRM	**I**magerie par **R**ésonance **M**agnétique
JPEG	**J**oint **P**hotographic **E**xperts **G**roup
JPEG 2000	**J**oint **P**hotographic **E**xperts **G**roup **2000**
Ko	**k**ilo-**o**ctet
LUT	**L**ook **U**p **T**able
LZ	**L**empel-**Z**iv
LZ77	**L**empel-**Z**iv 19**77**
LZ78	**L**empel-**Z**iv 19**78**
LZW	**L**empel-**Z**iv-**W**elch
Ms	**M**illi**s**econde
NBS	**N**ational **B**ureau of **S**tandards
NSA	**N**ational **S**ecurity **A**gency
Oct	**Oct**ets

P-Box	**P**ermutation **B**ox
PC	**P**ersonal **C**omputer
PFE	**P**rojet de **F**in d'**E**tude
PI	**P**ermutation **I**nitiale
PNG	**P**ortable **N**etwork **G**raphics
PPP	**P**oints **P**ar **P**ouce
PSD	**P**hoto**S**hop **D**ocument
RC	**R**on's **C**ode ou **R**ivest's **C**ipher
RC2	**R**on's **C**ode ou **R**ivest's **C**ipher **2**
RC4	**R**on's **C**ode ou **R**ivest **C**ipher **4**
RC5	**R**on's **C**ode ou **R**ivest's **C**ipher **5**
RC6	**R**on's **C**ode ou **R**ivest's **C**ipher **6**
RLE	**R**un-**L**ength **E**ncoding
RSA	**R**ivest, **S**hamir et **A**dleman
RVQ	**R**ecursive**V**ector **Q**uantization
S-Box	**S**ubstitution **B**ox
Scanner X	**Scanner** à rayons **X**
TEP	**T**omographie par **E**mission de **P**ositons
TIFF ou TIF	**T**agged **I**mage **F**ile **F**ormat
USA	**U**nited **S**tate of **A**merica
W3C	**W**orld **W**ide **W**eb **C**onsortium
XOR	**O**U exclusif

H&R Block : Est une préparation de déclaration de la société aux Etats-Unis fondée par les frères Henry W.Bloch et Richard Bloch

PICA : unité de mesure typographique utilisée dans les pays anglo-saxons

INTRODUCTION GENERALE

Introduction générale

C'est avec l'apparition des ordinateurs, et surtout avec Internet, que les images sont devenues omniprésentes et prépondérantes. En effet, quoi de plus précis qu'une image ? Il faudrait plusieurs pages de texte pour décrire une simple photographie, ou un schéma…

Cependant, si dans la rapidité d'information l'image détrône facilement le texte, celle-ci devient vite lourde de données, donc excessivement longue à transmettre et à traiter. Voilà pourquoi depuis quelques années, les centres de recherche en informatique dépensent de nombreuses heures sur des algorithmes de compression. Afin de limiter la taille, ou le poids, d'une image, nous devons la compresser, c'est-à-dire éliminer les informations inintéressantes ou redondantes. Il existe de nos jours plus d'une vingtaine de formats de compression, spécifiquement dans la compression d'image (.gif, .jpeg, .bmp…), ayant chacun leur propre méthode de codage, ou cumulant plusieurs algorithmes, mais tous sont complémentaires.

La cryptologie est étymologiquement la science du secret. Elle regroupe en fait de nombreuses notions. Historiquement, la cryptologie a servi à sécuriser certaines transmissions militaires, c'est-à-dire à chiffrer des messages. Plus récemment, et notamment depuis l'avènement de l'ère numérique, la cryptologie a connu d'autre usages, comme celui de la signature numérique ou de l'identification des personnes.

Nous sommes intéressé dans notre projet de fin d'étude à la compression des images par l'algorithme de huffman qui est utilisé dans les différents standards de compression des données.

Les images compressées par huffman seront automatiquement chiffrés et cryptés par les algorithmes de chiffrement à clé secrète comme DES, IDEA et RC5.

La cryptographie et la compression sont deux disciplines importantes pour augmenter le niveau de sécurité et la vitesse de transmission des données sur internet.

Notre mémoire est structuré comme suite :

- Dans le **premier chapitre** nous parlerons sur les caractéristiques des images d'une manière générale.

- Dans le **deuxième chapitre** nous présentons les méthodes de compression des images à savoir la compression sans perte et la compression avec perte. Est nous décrivons la structure des algorithmes de compression des images par huffman et RLE.

- Dans le **troisième chapitre** nous présentons les notions de base de la cryptographie classique et à clé secrète.

- Dans le **quatrième chapitre** nous décrivons la structure des algorithmes de cryptographie à clé secrète comme DES, IDEA et RC5.

- Le **Cinquième chapitre** sera consacré aux résultats obtenus de l'opération de compression des images par l'algorithme huffman comparé avec l'algorithme de compression RLE. Les images compressées par huffman seront cryptée par les algorithmes à clé secrète comme DES, IDEA et RC5.

CHPITRE I

traitement des images

1. Introduction

On désigne par "technique de traitement d'images "toutes les techniques ayant pour but la modification des caractéristiques chromatiques des pixels des images bitmap. Le traitement d'images c'est une discipline de l'informatique et des mathématiques appliquées qui étudie les images numériques et leurs transformations, dans le but d'améliorer leur qualité ou d'en extraire de l'information.

Traitement d'images est souvent synonyme d'amélioration des images avec pour but l'obtention d'une plus grande lisibilité. Il n'y a pas création d'informations, mais mise en évidence de l'information pertinente déjà présente. Ce chapitre sera consacré sur une introduction générale sur le traitement d'image.

2. Définition d'image

Une image est la représentation d'une scène acquise à l'aide de systèmes de production d'images (appareils photographique, caméra, radiographies, scanner, sonar,....). Sa forme peut être analogique (ex: négatif, photographie, vidéo..) ou numérique (images numérisées suivant divers formats (images compressées ou non...) ou obtenues par des capteurs fournissant des images numérisées) et dans ce cas un traitement par ordinateur est possible. [s1]

3. Structure générale d'un fichier image

Un fichier image contient toujours (ou presque) un **en-tête** ou **header** contenant un certain nombre d'informations indispensables à la relecture du fichier et au décodage des données image. Parmi ces informations (appelées *tag*, en Anglais), la première est généralement un code (On parle souvent de *magic number*) caractérisant le format du fichier. C'est ce code et non l'extension associée au nom du fichier, qui permet d'identifier le format et de décoder correctement la suite des données.

On trouve, ensuite, à une place et dans un ordre variable d'un format à l'autre, la taille de l'image (hauteur, largeur, en nombre des pixels) et la profondeur (la troisième dimension de l'image), précisant la taille d'un pixel, c'est-à-dire la valeur maximale

que peut prendre l'intensité du pixel. Cette valeur est parfois remplacée par le nombre de bits sur lequel est codé un pixel. Les valeurs de profondeur les plus courantes sont :

- 1 bit pour les images dites binaire (noir et blanc) ;
- 8 bits pour les images monochromes en tons continus (256 niveaux de gris) ou en couleur indexées (256 teintes).
- 24 bits (en réalité 3 x 8 bits) pour les images en vraies couleurs, ce qui correspond à 16777 216 = $256^3 = 2^{24}$ teintes.

La plupart des formats contiennent également des informations relatives à la colorimétrie (noir et blanc, niveaux de gris, couleurs indexées ou vraies couleurs, transparence), à l'organisation des données.

Pour les formats plus élaborés qui proposent plusieurs options de codage, des informations complémentaires doivent figurer dans l'en-tête. Cela concerne essentiellement le mode de compression, le mode de transmission.

Si les données ont été compressées, tous les paramètres et données supplémentaires utilisés par le codeur doivent être fournis au décodeur et doivent donc figurer dans l'en-tête. Ces données, qui ne sont généralement pas compressés peuvent être plus volumineux que le fichier contenant les données brutes de l'image.

Enfin, dans la plupart des formats, des données supplémentaires facultatives peuvent être stockées dans l'en-tête. Il s'agit alors de commentaires qui seront ignorés par le décodeur.

Après l'en-tête viennent les données image proprement dites. Dans les formats bruts (non compressés), ces données correspondent directement aux intensités lumineuses associées aux pixels.

4. Image analogique

Avec la parole, l'image constitue l'un des moyens les plus importants qu'utilise l'homme pour communiquer avec autrui. C'est un moyen de communication universel dont la richesse du contenu permet aux êtres humains de tout âge et de toute culture de se comprendre.

Elle est définie comme étant la reproduction exacte ou la reproduction analogique d'une scène réelle.

Elle peut être décrite sous la forme d'une fonction F(x, y) de brillance analogique continue, définie dans un domaine borné [noir, blanc], tel que x et y sont les coordonnées spatiales d'un point de l'image, et F est une fonction d'intensité lumineuse et de couleur. Sous cette forme, l'image est inexploitable par l'ordinateur qui ne connaît que le langage binaire, ce qui nécessite sa numérisation. [s1]

5. La numérisation

La numérisation d'une image analogique est la conversion de celle-ci de son état analogique (distribution continue d'intensités lumineuses), en une image numérique représentée par une matrice bidimensionnelle de valeurs numériques F(x, y) où :

- x, y : Coordonnées cartésiennes d'un point de l'image.
- F(x, y): niveau de gris en ce point.

L'opération de numérisation comprend trois phases (figure I.1) :

5.1. Échantillonnage

L'échantillonnage commence par découper l'image en surfaces carrées élémentaires d'une matrice carrée ou rectangulaire ; chacun des carrés éléments d'image est appelé pixel (Picture Cell) et repéré par ses coordonnées x et y.

5.2. Quantification

Les lignes sont étudiées les unes après les autres et sur chacune, la valeur de chaque pixel est mesurée; parfois ce pixel à une structure hétérogène; la valeur retenue est alors une moyenne, des détails seront donc perdus.

5.3. Codage

Représentation informatique (binaire) des valeurs représentant les pixels. [s2]

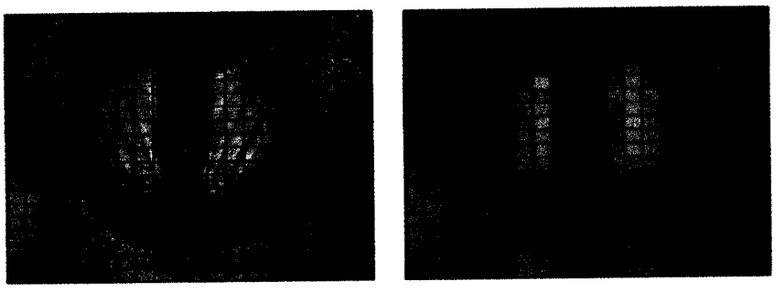


Figure I. 1 : La numérisation d'une image analogique. [s3]

6. Pixel «Picture Element»

Une image numérique est constituée d'un ensemble de points appelés pixels (pixel est une abréviation de (PICture ELement)) Le pixel représente ainsi le plus petit élément constitutif d'une image numérique. L'ensemble de ces pixels est contenu dans un tableau à deux dimensions constituant l'image [s4]. La lettre A, par exemple, peut être affichée comme un groupe de pixels dans la figure ci-dessous:

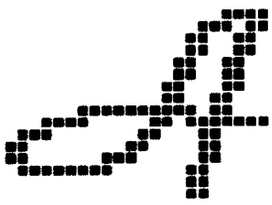


Figure I. 2: L'image comme un groupe de pixels. [s5]

Dans le cas d'une image monochrome, chaque pixel est codé sur 8 bits, et la taille mémoire nécessaire pour afficher une telle image est directement liée à la taille de l'image. Dans une image couleur (R.V.B.), un pixel peut être représenté sur trois octets: un octet pour chacune des couleurs: rouge (R), vert (V) et bleu (B). [s5]

7. Image numérique

Contrairement aux images obtenues à l'aide d'un appareil photo, ou dessinées sur du papier, les images manipulées par un ordinateur sont numériques (représentées par une série de bits). L'image numérique est l'image dont la surface est divisée en éléments de tailles fixes appelés cellules ou pixels, ayant chacun comme caractéristique un niveau de

gris ou de couleurs prélevé à l'emplacement correspondant dans l'image réelle, ou calculé à partir d'une description interne de la scène à représenter. [s2]

8. Types d'image

On distingue deux types d'images à la composition et au com portement différent: les images matricielles et les images vectorielles.

8.1. Images matricielles (ou images bitmap)

Elle est composée comme son nom l'indique d'une matrice (tableau) de points à plusieurs dimensions, chaque dimension représentant une dimension spatiale (hauteur, largeur, profondeur), temporelle (durée) ou autre (par exemple, un niveau de résolution).

8.1.1. Images 2D

Dans le cas des images à deux dimensions (le plus courant), les points sont appelés pixels. D'un point de vue mathématique, on considère l'image comme une fonction de $R \times R$ dans R où le couplet d'entrée est considéré comme une position spatiale.

8.1.2. Images 2D + temps (vidéo), images 3D, images multi résolution

Lorsqu'une image possède une composante temporelle, on parle d'animation.

Dans le cas des images à trois dimensions les points sont appelés des voxels. Ils représentent un volume.

Ces cas sont une généralisation du cas 2D, la dimension supplémentaire représentant respectivement le temps, une dimension spatiale ou une échelle de résolution. D'un point de vue mathématique, il s'agit d'une fonction de $R \times R \times R$ dans R. **[2]**

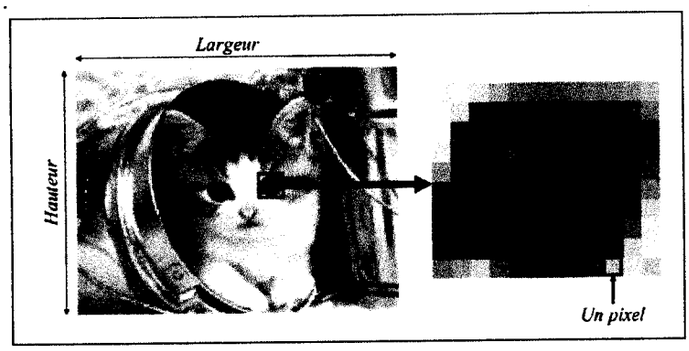


Figure I. 3: Image matricielle [s22].

➢ **La structure d'un fichier bitmap est la suivante**

- En-tête du fichier (en anglais file header)
- En-tête du bitmap (en anglais bitmap information header, appelé aussi information Header)
- Palette (optionnellement)
- Corps de l'image.

Entête du fichier

L'entête du fichier fournit des informations sur type du fichier, sa taille et indique où commencent les informations concernant l'image à proprement parler.

L'entête se compose de quatre champs:

- La signature (2 octets)
- La taille totale du fichier (4 octets)
- Un champ réservé (4 octets)
- L'offset de l'image (4 octets)

Entête de l'image

L'entête de l'image fournit des informations de l'image, notamment ses dimensions et ses couleurs, elle se compose des champs suivants:

- La taille totale de l'image en octets (sur 4 octets)
- La résolution horizontale (sur 4 octets)
- La résolution verticale (sur 4 octets)
- Le nombre de couleurs de la palette (sur 4 octets)
- Le nombre de couleurs importantes de la palette (sur 4 octets). Ce champ peut être égal à 0 lorsque chaque couleur a son importance.

Palette de l'image

La palette est optionnelle, lorsqu'elle est définie, elle contient successivement 4 octets pour chacune de ses entrées représentant:

- La composante bleue (sur un octet)
- La composante verte (sur un octet)
- La composante rouge (sur un octet)
- Un champ réservé (sur un octet)

Corps de l'image

Le codage de l'image se fait en écrivant successivement les bits correspondant à chaque pixel, ligne par ligne en commençant par le pixel en bas à gauche, comme illustré dans la figure. **[s4]**

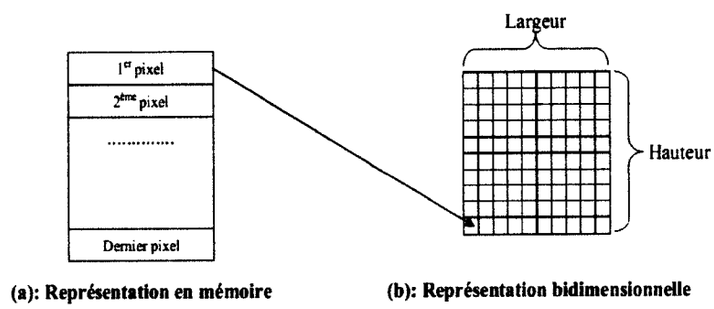


Figure I. 4: Représentations d'une image BMP. [s4]

➢ **Usage des images matricielles**

Les images bitmaps sont utilisées pour la plupart des illustrations graphiques ayant un nombre de couleurs important, comme celles présentent sur les pages Web, les photographies numérisées, etc. **[s6]**

8.2. Images vectorielles

Le principe est de représenter les données de l'image par des formules géométriques qui vont pouvoir être décrites d'un point de vue mathématique. Cela signifie qu'au lieu de mémoriser une mosaïque de points élémentaires, on stocke la succession d'opérations conduisant au tracé. Par exemple, un dessin peut être mémorisé par l'ordinateur comme « une droite tracée entre les points (x1, y1) et (x2, y2) », puis « un cercle tracé de centre (x3, y3) et de rayon 30 de couleur rouge ».

L'avantage de ce type d'image est la possibilité de l'agrandir indéfiniment sans perdre la qualité initiale, ainsi qu'un faible encombrement. L'usage de prédilection de ce type d'images concerne les schémas qu'il est possible de générer avec certains logiciels de DAO (Dessin Assisté par Ordinateur) comme AUTO CAD. Ce type d'images est aussi utilisé pour les animations Flash, utilisées sur Internet. Étant donné que les moyens de visualisation d'images actuels comme les moniteurs d'ordinateur reposent essentiellement sur des images matricielles, les descriptions vectorielles. Doivent préalablement être converties en descriptions matricielles avant d'être affichées comme images. **[s7]**

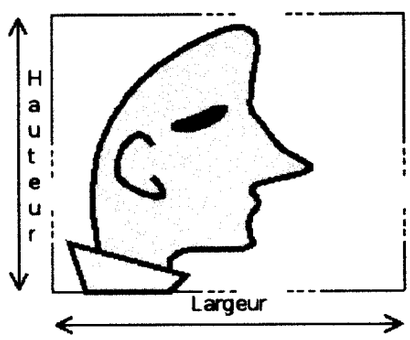


Figure I. 5: image vectorielle [s7]

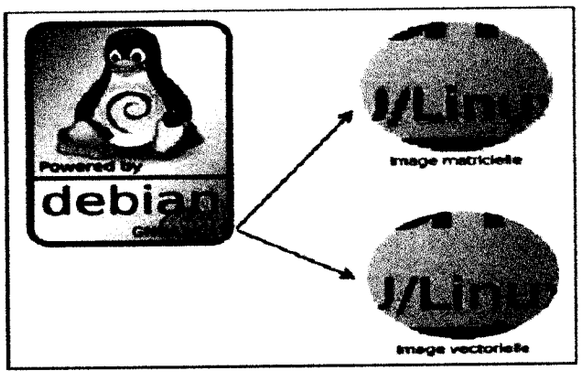


Figure I. 6: Effet de l'agrandissement d'une image vectorielle par rapport à une image matricielle. [s2]

9. Caractéristiques d'une image numérique

L'image est un ensemble structuré d'informations caractérisé par les paramètres suivants:

9.1. La définition et la résolution d'une image

On appelle la définition d'une image, le nombre de points (pixel) constituant l'image, c'est-à-dire sa «dimension informatique» (le nombre de colonnes de l'image que multiplie son nombre de lignes). Une image possédant 640 pixels en largeur et 480 en hauteur aura une définition de 640 pixels par 480, notée 640x480.

La résolution, terme souvent confondu avec la "définition", détermine par contre le nombre de points par unité de surface, exprimé en points par pouce (PPP, en anglais DPI pour Dots Per Inch); un pouce représentant 2.54 cm. La résolution permet ainsi d'établir le rapport entre le nombre de pixels d'une image et la taille réelle de sa représentation sur un support physique. Une résolution de 300 dpi signifie donc 300 colonnes et 300 rangées de pixels sur un pouce carré ce qui donne donc 90000 pixels sur un pouce carrer. La résolution de référence de 72 dpi nous donne un pixel de 1"/72 (un pouce divisé par 72) soit 0.353mm, correspondant à un point pica (unité typographique anglo saxonne). [s8]

9.2. Histogramme

Un histogramme est un graphique statistique permettant de représenter la distribution des intensités des pixels d'une image, c'est-à-dire le nombre de pixels pour chaque intensité lumineuse. Par convention un histogramme représente le niveau d'intensité en abscisse en allant du plus foncé (à gauche) au plus clair (à droite).

Ainsi, l'histogramme d'une image en 256 niveaux de gris sera représenté par un graphique possédant 256 valeurs en abscisses, et le nombre de pixels de l'image en ordonnées. Prenons par exemple l'image suivante composée de niveaux de gris:

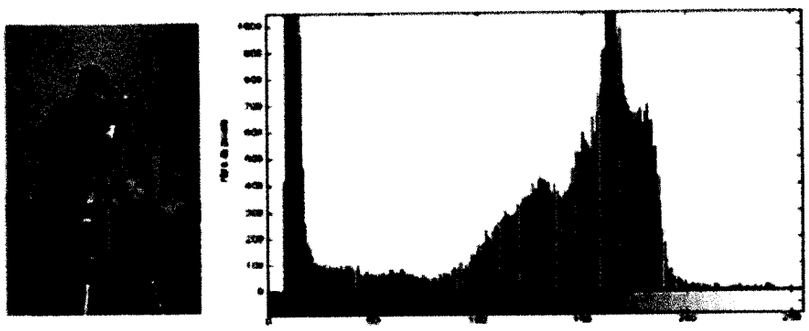


Figure I. 7: Exemple d'image en niveaux de gris. [s9]

Pour les images en couleur plusieurs histogrammes sont nécessaires.

Par exemple pour une image codée en RGB:

- un histogramme représentant la distribution de la luminance
- trois histogrammes représentant respectivement la distribution des valeurs respectives des composantes rouges, bleues et vertes. [s9]

9.3. Contours et textures

Les contours représentent la frontière entre les objets de l'image, ou la limite entre deux pixels dont les niveaux de gris représentent une différence significative. Les textures décrivent la structure de ceux-ci. L'extraction de contour consiste à identifier dans l'image les points qui séparent deux textures différentes. [s2]

9.4. Luminance

C'est le degré de luminosité des points de l'image. Elle est définie aussi comme étant le quotient de l'intensité lumineuse d'une surface par l'aire apparente de cette surface,

pour un observateur lointain, le mot luminance est substitué au mot brillance, qui correspond à l'éclat d'un objet [s1]. La moyenne ou luminance (brillance) d'une image numérique en niveau de gris est définie comme la moyenne des pixels de l'image: [s10]

$$Lum() = \frac{1}{M \times N} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)$$

Avec :

M: nombre de colonnes

N: nombre de lignes

f(x, y): la valeur de niveaux de gris dans le pointe (x, y)

9.5. Contraste

C'est l'opposition marquée entre deux régions d'une image, plus précisément entre les régions sombres et les régions claires de cette image. Le contraste est défini en fonction des luminances de deux zones d'images. Si L1 et L2 sont les degrés de luminosité respectivement de deux zones voisines A1 et A2 d'une image, le contraste C est défini par le rapport: [s2]

$$C - \frac{L1 - L2}{L1 + L2}$$

9.6. Profondeur

C'est le nombre de bits par pixel, cette valeur reflète le nombre de couleurs ou de niveaux de gris d'une image, par exemple :

- 32 bits/pixel = 1,07 milliards de couleurs
- 24 bits = 16,7 millions de couleurs
- 16 bits = 65 536 couleurs
- 8 bits = 256 couleurs

9.7. Le poids de l'image

Le poids d'une image se détermine en fonction de ces deux paramètres: dimensions, profondeur. Le poids de l'image est alors égal à sa dimension multipliée par sa profondeur. Par exemple, pour une image 640x480 en vraies couleurs (True colors) :

- Nombre de pixels (dimension) : 640 x 480 = 307200.
- Poids de chaque pixel (profondeur) : 24 bits = 3 octets.
- Le poids de l'image est ainsi égal à : 307200 x 3 = 921600 octets.

9.8. Bruit

Un bruit (parasite) dans une image est considéré comme un phénomène de brusque variation de l'intensité d'un pixel par rapport à ses voisins, il provient de l'éclairage des dispositifs optiques et électroniques du capteur. [s5]

10. Représentation des couleurs

II existe différentes catégories d'image selon le nombre de bit sur lequel est codée la valeur de chaque pixel.

10.1. L'image monochrome (binaire)

C'est le plus simple type d'image où chaque pixel peut prendre uniquement la valeur noire ou blanche (chaque pixel est codé sur un seul bit). C'est typiquement le type d'image que l'on utilise pour scanner du texte quand celui ci est composé d'une seule couleur.

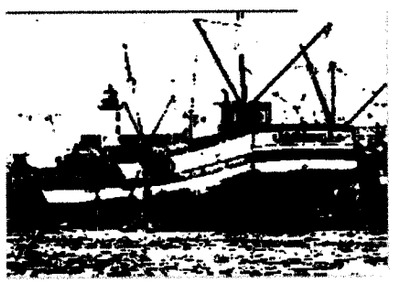


Figure I. 8: Une image binaire [s23].

10.2. L'image en niveaux de gris

Le niveau de gris est la valeur de l'intensité lumineuse à un point. La couleur du pixel peut prendre des valeurs allant du noir au blanc en passant par un nombre fini de niveaux intermédiaires. Donc pour représenter les images à niveaux de gris (Figure I.9), on peut attribuer à chaque pixel de l'image une valeur correspondante à la quantité de lumière renvoyée. Cette valeur peut être comprise par exemple entre 0 et 255. Chaque pixel n'est donc plus représenté par un bit, mais par un octet.

Figure I. 9: Image en niveaux de gris [s23].

10.3. L'image en couleurs

Même s'il est parfois utile de pouvoir représenter des images en noir et blanc ou en niveau de gris, les applications multimédias utilisent le plus souvent des images en couleurs (Figure I.10).

La représentation des couleurs s'effectue de la même manière que les images en niveaux de gris avec cependant quelques particularités. En effet, il faut tout d'abord choisir un modèle de représentation.

Figure I. 10: Image en couleurs [s23].

Il existe plusieurs modes de représentation de la couleur, le plus utilisé pour les images numériques est le mode RGB. Ce mode est basé sur la synthèse additive des couleurs, c'est à dire que le mélange des trois composantes (R, G, B) donne une couleur.

Il existe différents types d'images couleurs en fonction du nombre de bits utilisés pour le stockage de l'information couleur.

10.3.1. Image en "vraies couleurs" (True colors) (ou 24 bits)

Il s'agit d'une appellation trompeuse car on est dans un monde numérique (discret, fini) qui ne peut pas rendre compte de la réalité (infinie).

Chaque pixels peut prendre une valeur dans le RGB comprise entre 0 et 255 (soit 256x256x256 possibilités= plus de 16 millions de possibilités).

L'information couleur de chaque pixel est donc codée par 3 octets, par conséquent les images en vraies couleurs sont très "lourdes" en termes de l'espace de stockage.

10.3.2. Image indexée

Afin de diminuer la charge de travail nécessaire pour manipuler des images en 24 bits (et/ou gagner l'espace de stockage), on peut utiliser le mode de représentation en couleurs indexée.

Dans ce cas on attache une palette de 256 couleurs (usuellement) à l'image.

Ces 256 couleurs sont choisies parmi les 16 millions de couleurs de la palette RGB. Pour chaque image le programme recherche les 256 couleurs les plus pertinentes.

Chaque pixel va non plus véhiculer le code couleur RGB qui lui est affecté, mais simplement un chiffre compris entre 0 et 255. A chacun de ces chiffres va correspondre une couleur, définie par son code RGB et stockée dans une palette (Look Up Table (LUT)) avec les données de l'image

Lors de la visualisation de l'image, la correspondance se fait entre le numéro de la couleur affecté à chaque pixel (compris entre 0 et 255) et le code couleur RGB correspondant issu de la LUT. [s10]

Figure I. 11: Exemple d'une image en 256 couleurs avec sa palette associée [s23].

11. Les formats d'image

➢ **Définition**

Un format d'image est une représentation informatique de l'image, associée à des informations sur la façon dont l'image est codée et fournissant éventuellement des indications sur la manière de la décoder et de la manipuler **[s7]**.

Nom du format	Créateur	Points forts	Points faibles	Note
JPEG et JPEG2000	Joint Photographic Experts Group	Excellente compression	Compression destructrice	Spécialement conçu pour les photographies, il est cependant à utiliser avec délicatesse tant sa compression peut brouiller l'image. Le format JPEG2000, évolution du format original, peut être réglé pour compresser sans pertes.
GIF (Graphical Interchange Format)	H&R Block et Compuserve	Possibilité d'animation et de transparence, compression efficace	Limité à 256 couleurs	Très répandu sur le Web malgré ses faiblesses et un problème de droit sur son format de compression. À déconseiller pour les photos.
PNG (Portable	W3C	Excellente compression	Pas très efficace pour	Format destiné à remplacer le

Network Graphics)		sans perte. Possibilité de transparence. Standard donc pérenne.	les larges photographies	format GIF et ses limitations, mais ayant encore du mail à s'implanter dans les habitudes des développeurs. Peut remplacer les JPEG comme les GIF (sauf en ce qui concerne l'animation).
TIFF (Tagged Image File Format)	Aldus	Compression sans perte efficace. Couche de transparence.	Lourdeur des fichiers non compressés. Format propriétaire.	Format de stockage très utilisé, à éviter pour le Web
BMP (Bitmap)	Microsoft	Format par défaut de Windows	Disponible uniquement sur la plateforme de Microsoft	Généralement non compressé - le format RLE est possible mais peu reconnu
PSD (Photoshop Document)	Adobe	Très complet, conserve toutes les couches de l'image.	Taille prohibitive	Format standard de l'outil Photoshop, très populaire mais à limiter au stockage de retouche d'image du fait de la taille des fichiers

Tableau I. 1: Les différents formats d'image bitmap. [s11]

12. Système de traitement d'images

Le système de traitement est illustré dans la figure suivante:

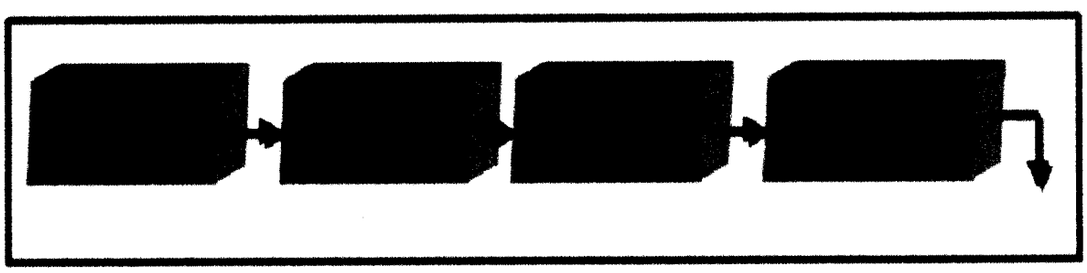


Figure I.12: Schéma d'un système de traitement d'images

L'image parvient en mode réel, une procédure de numérisation (transformation du mode réel en mode numérique) est effectuée grâce au système d'acquisition.

Ces images acquises, qui ne peuvent échapper aux effets de dégradation dus essentiellement aux phénomènes physiques tels que le flou dû au mouvement de l'image durant son acquisition, sont soumises à une étape de prétraitement.

12.1. Acquisition

La première étape d'une chaîne de traitement et d'analyse des images numériques est celle de l'acquisition d'une scène. Les objets de cette scène peuvent émettre de l'énergie, on parle alors d'imagerie en émission, comme l'imagerie infrarouge. Mais le plus souvent ils reçoivent une énergie lumineuse qu'ils vont en partie absorber et en partie réfléchir. L'analyse de l'énergie absorbée est le fait de l'imagerie en absorption, comme dans le cas de l'imagerie ultrasonore (échographie par exemple). L'analyse de l'énergie réfléchie concerne la majorité des applications de traitement d'images, où un capteur se substitue à l'œil d'un observateur. Un objet n'est alors visible que s'il est éclairé. Plus un objet réfléchit l'énergie lumineuse qu'il reçoit, plus il sera clair. Ce sont par ailleurs les différences d'absorption et donc de réflexion, des différentes longueurs d'ondes d'un spectre lumineux qui vont donner la couleur d'un objet [s1].

12.2. Prétraitement des images

Une image brute est toujours entachée de dégradations d'origine diverses, dont il va falloir essayer de minimiser l'influence afin d'obtenir l'image la plus nette possible, c'est le rôle de la fonction prétraitements a qui incombe l'opération de clarification de l'image. Les prétraitements représentent des fois une étape incontournable.

12.2.1. Amélioration d'une image:

Elle vise d'obtenir une image de meilleure qualité visuelle.

Cependant, l'œil humain étant essentiellement sensible aux forts contrastes, les méthodes d'améliorations tentent le plus souvent d'augmenter ceux-ci, dans le but d'accroître la séparabilité des régions qui composent l'image. Il existe plusieurs méthodes d'amélioration qu'on peut regrouper dans deux grandes classes:

Les méthodes ponctuelles

Ces méthodes traitent les pixels de l'image de façon indépendante de leur voisinage. La seule information prise en compte est sa valeur de niveau de gris (ou sa couleur) quelle que soit sa position dans l'image. Ces méthodes sont basées sur des modifications et spécifications d'histogrammes.

Les méthodes globales

Ces méthodes utilisent généralement des transformées (comme la transformée de Fourier par exemple) qui permettent d'obtenir une nouvelle représentation de l'image, dans un nouvel espace, dans lequel l'opération d'amélioration proprement dite est réalisée. La transformation inverse permet alors de retrouver l'image améliorée, représentée dans son espace d'origine.

12.2.2. La restauration

La restauration d'images a pour objectif principal d'atténuer, voire de supprimer dans les cas les plus favorables, les dégradations que subit une image. Le plus souvent, les dégradations considérées sont celles qui apparaissent au moment de l'acquisition de l'image, c'est-à-dire au moment de sa création.

Les méthodes utilisées en restauration d'images sont nombreuses et directement liées aux types de dégradations qu'elles tentent de corriger. Cependant, ces méthodes ont en commun de traiter le problème par l'intermédiaire de représentations fréquentielles. Les dégradations que subissent les images ont des effets sur ces images qu'il est plus facile d'expliquer et de modéliser sur une représentation fréquentielle que sur l'image elle-même. [s1]

12.3. La segmentation

Le cœur d'un système d'analyse automatique d'images est l'étape de la segmentation ou d'extraction des caractéristiques. La segmentation des images consiste à regrouper les pixels de ces images qui partagent une même propriété pour former des régions connexes. Il existe deux familles d'approches que l'on peut faire coopérer: les approches « contours » et les approches « régions ».

Dans le premier cas, les régions sont délimitées par les contours des objets qu'elles représentent (séparation). Dans le second cas, les régions sont déterminées en fonction de leurs propriétés intrinsèques (agrégation de pixels fonction d'un critère

d'homogénéité). Les deux approches sont duales. Une image est donc une mosaïque de régions homogènes séparées par des zones de transitions. [s12]

12.4. Interprétation

L'objectif de la segmentation est de permettre l'exploitation du contenu de l'image à des fins d'interprétation (aide au diagnostic en imagerie aérienne, satellitaire, médicale par exemple), de localisation et de reconnaissance (vidéosurveillance, contrôle robotique) ou de mesure des évolutions (contrôle qualité, suivi thérapeutique, etc.).[s1]

13. Domaine d'application

Le traitement d'images possède l'aspect multidisciplinaire. On trouve ses applications dans des domaines très variés tels que: Médecine (Radiographies, tomographies,...), Biologie, Météorologie, Astronomie, Géologie, Physique (spectroscopie, physique des plasmas,...), Applications militaires, Applications industrielles (Robotique, surveillance de qualité,...), Photographies, Publicité, etc. [s2]

14. Conclusion

Dans ce chapitre, nous avons présenté les caractéristiques fondamentales d'une image. Nous sommes intéressés dans notre mémoire aux images BMP. Ce sont des images non compressée. L'image BMP est Caractérisée par leur simplicité et leur format en pixel qui est facile a lire et la récupérée dans le but de le compressé par HUFFMAN et de le crypté. Le chapitre 2 sera consacré pour traiter les différentes notions de base de compression des images.

CHPITRE II

Compression des images

1. Introduction

La qualité visuelle des images numériques augmente continuellement avec le développement de nouvelles techniques d'affichage (multi-résolution, transmission progressive...) et de nouvelles technologies d'acquisition (haute définition, nouveau hardware...). La taille de ces images augmente proportionnellement à leur qualité. Leurs stockages ainsi que leurs transmissions, constituent ainsi des enjeux principaux dans le monde numérique. La compression de ces grands volumes d'informations s'impose alors comme une étape incontournable pour optimiser la taille mémoire nécessaire à leurs archivages ainsi que le temps nécessaire à leurs transmissions à distance via les réseaux informatiques.

Dans ce chapitre, nous donnons quelques notions essentielles sur la compression des images fixes. La structure générale des algorithmes utilisés ainsi que les paramètres permettant d'évaluer leurs performances.

2. L'objectif de la compression des images

L'objectif principal de la compression d'image est de réduire la quantité d'informations nécessaire à sa représentation. Cette réduction est obtenue en minimisant la redondance de l'information présente dans une image. Cette redondance peut être d'origine statistique, spatiale ou encore fréquentielle. Les méthodes de compression d'images se différencient par la technique utilisée pour minimiser cette redondance tout en sauvegardant les informations pertinentes présentes dans ces images [13].

3. Structure générale d'un schéma de compression d'images fixes

Le processus de compression d'une image fixe nécessite trois étapes principales, à savoir, l'étape de transformation ou de décorrélation, l'étape de quantification et l'étape de codage entropique. La structure générale d'un schéma de compression d'images fixes est donnée par la figure II.1.

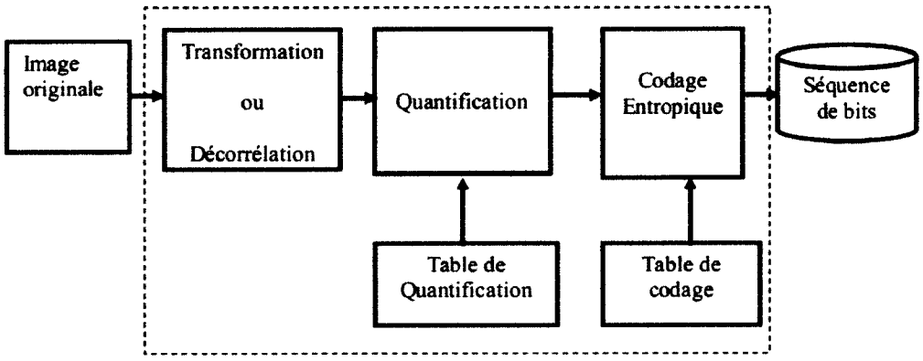


Figure II. 1: Structure général d'un schéma de compression d'images. [s13]

3.1. Etape de transformation ou de décorrélation :

Cette étape a pour objectif de réduire la redondance d'informations contenues dans l'image, afin d'obtenir une représentation de l'image qui se prête à la quantification et au codage. De plus, elle nous offre la possibilité d'ajuster les erreurs de quantification selon la sensibilité du système visuel humain. Cette transformation est obtenue en exploitant les fortes corrélations spatiales entre les pixels de l'image. Elle permet de mettre en évidence sous une forme plus compacte l'information utile contenue dans l'image.

3.2. Etape de quantification

Le but de cette étape est de réduire le nombre de bits nécessaires à la représentation des coefficients issus de l'étape de transformation. Pour ce faire, les valeurs de ces coefficients sont approximées par un ensemble fini d'éléments (scalaire ou vecteur) qui forme le dictionnaire de quantification. Ainsi, un élément quelconque d'entrée est remplacé par l'élément du dictionnaire le plus approprié

3.3. Etape de codage entropique

C'est la phase finale du processus de compression qui permet de générer le flot binaire à stocker ou à transmettre. Notons que le codage entropique est un codage sans pertes. [s13]

4. Les types des méthodes de compressions des images

4.1. Compression physique et compression logique

On considère généralement la compression comme un algorithme capable de comprimer énormément de données dans un minimum de place (**compression physique**), mais on peut également adopter une autre approche et considérer qu'en premier lieu un algorithme de compression a pour but de recoder les données dans une représentation différente plus compacte contenant la même information (**compression logique**).

La distinction entre compression physique et logique est faite sur la base de comment les données sont compressées ou plus précisément comment est-ce que les données sont réarrangées dans une forme plus compacte.

La compression **physique** Est exécutée exclusivement sur les informations contenues dans les données. Cette méthode produit typiquement des résultats incompréhensibles qui apparemment n'ont aucun sens. Le résultat d'un bloc de données compressées est plus petit que l'original car l'algorithme de compression physique a retiré la redondance qui existait entre les données elles-mêmes. Toutes les méthodes de compression dont nous allons traiter sont des méthodes physiques.

La compression **logique** Est accomplie à travers le processus de substitution logique qui consiste à remplacer un symbole alphabétique, numérique ou binaire en un autre. Changer "United State of America" en "USA" est un bon exemple de substitution logique car "USA" est dérivé directement de l'information contenue dans la chaîne "United State of America" et garde la même signification. La substitution logique ne fonctionne qu'au niveau du caractère ou plus haut et est basée exclusivement sur l'information contenue à l'intérieur même des données. [s13]

4.2. Compression symétrique et asymétrique

Dans le cas de la compression **symétrique**, la même méthode est utilisée pour compresser et décompresser l'information, il faut donc la même quantité de travail pour chacune de ces opérations. C'est ce type de compression qui est généralement utilisée dans les transmissions de données.

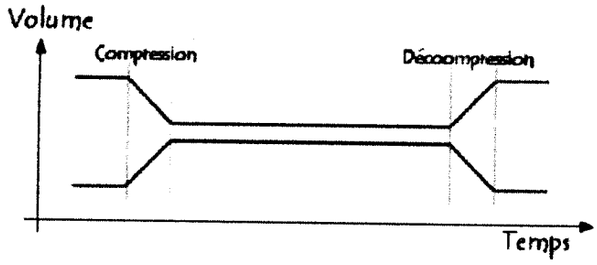


Figure II. 2: compression de type symétrique. [s14]

La compression **asymétrique** demande plus de travail pour l'une des deux opérations, on recherche souvent des algorithmes pour lesquels la compression est plus lente que la décompression. Des algorithmes plus rapides en compression qu'en décompression peuvent être nécessaires lorsque l'on archive des données auxquelles on n'accède peu souvent (pour des raisons de sécurité par exemple), car cela crée des fichiers compacts. [s14]

4.3. Compression adaptatif, semi-adaptatif et non-adaptatif

Certains codeurs statiques comme le codage de Huffman sont conçus pour compresser seulement des types spécifiques de données. Ces codages ***non-adaptatifs*** contiennent un dictionnaire statique de chaînes de caractères prédéfinies qui sont connues comme apparaissant à de grandes fréquences dans les données à encoder. Par exemple, un codeur non-adaptatif conçu spécifiquement pour compresser la langue française contiendra un dictionnaire avec des chaînes de caractères telles que "et", "mais", "de", "le", car ces chaînes apparaissent très fréquemment dans les textes en français.

Un codeur **adaptatif**, à l'inverse n'intégrera pas de données relatives à la fréquence d'apparitions des données à compresser. Des compresseurs adaptatifs comme LZW ou Huffman dynamique déterminent la dépendance des données en construisant leur dictionnaire à la volée. Ils n'ont pas de listes prédéfinies de chaînes de caractères par exemples mais les construisent dynamiquement à l'encodage.

La compression **adaptative** est capable de s'adapter à n'importe quelles données d'entrées et de retourner une sortie avec le taux de compression le meilleur possible. C'est une des principales différences avec les compressions ***non-adaptatives*** qui sont

capable d'avoir des codages efficaces uniquement avec un type de données d'entrées très restreint pour lequel ils ont étés conçus.

Un mélange de ces deux méthodes d'encodage à l'aide de dictionnaires est la méthode d'encodage ***semi*-adaptative**. Un encodage **semi-*adaptatif*** fait un premier passage sur les données pour construire le dictionnaire et un second passage pour effectuer l'encodage. En utilisant cette méthode, un dictionnaire optimal est construit avant qu'un quelconque encodage soit effectué. [4]

5. Pourquoi compresser

Les images doivent être compressée (comprimées, compactées) pour des raisons de place (économiser de l'espace disque) et parfois de vitesse de transfert. Différents formats et algorithmes ont été élaborés à cet effet par des équipes spécialisées. Notez que la qualité de l'image, la taille du fichier (en relation directe avec la vitesse de transfert) et la vitesse de compression/décompression varient également selon le programme (l'algorithme) choisi. En règle générale il faut choisir sa méthode et ses paramètres de compression selon les critères suivants qu'il faut optimiser selon les besoins:

- La rapidité de compression ;
- La rapidité de décompression ;
- Le taux de compression, ce qui influence directement la place occupée sur disque dur et la vitesse de transfert des fichiers;
- La qualité de l'image en relation avec la perte d'information (couplé parfois au taux de compression ou encore à la vitesse de compression) ;
- La portabilité, disponibilité de logiciels, standard pour les archives publiques, etc. [s15]

6. Compression sans perte et compression avec perte d'information

Il existe deux grandes catégories de compression de fichier: les compressions sans perte dites entropiques et les compressions avec perte. Les compressions sans perte conservent toutes les données d'origine pendant la compression et la décompression, elles sont souvent facilement généralisables à tous les types de données. Elles sont

indispensables pour le stockage de données texte ou de données numériques, par exemple pour des feuilles de calcul.

Les compressions avec perte permettent de compresser beaucoup plus les fichiers, et peuvent donc constituer une solution intéressante lorsque l'on a besoin de n'occuper qu'un espace disque restreint. Ces techniques entraînent la perte d'une partie des données d'origine, mais, jusqu'à un certain taux de compression, cette perte peut ne pas avoir d'incidence visible sur l'image finale, c'est à dire que les informations abandonnées pendant la compression n'affectent pas de façon notable la qualité de l'image.

Les compressions destructrices ne sont pas généralisables à tous types de données mais doivent être adaptées aux données afin de ne perdre que les informations de moindres importances. Les images permettent bien ce genre de compression grâce à des particularités qui leur sont propres.

Les algorithmes de compression exploitent deux caractéristiques:

- **Une caractéristique intrinsèque à l'image**

La redondance d'information. La compression diminue, voire supprime cette redondance d'information avec pour corollaire une diminution de la taille des fichiers et une augmentation du taux de transfert sur les réseaux ;

- **Une caractéristique extrinsèque à l'image**:

En l'occurrence du système perceptif visuel de l'Homme. En effet, ce système perceptif:

- est plus sensible à la luminance qu'à la chrominance, donc la Définition spatiale est plus importante que la Définition en chrominance ;
- est plus sensible à certaines couleurs qu'à d'autres ;
- ne perçoit qu'un nombre restreint de couleur ;

Ainsi, un algorithme sachant tirer parti de ces caractéristiques peut offrir d'intéressant taux de compressions sans que l'image paraisse altérée (alors qu'elle l'est).

La réduction totale ou partielle de la redondance conduit à des compressions sans perte d'information (pour autant que l'algorithme soit bien conçu) alors que l'exploitation des caractéristiques perceptives visuelles conduit à une dégradation de l'information non

perceptible ou peu perceptible jusqu'à un certain taux de compression. Au de là l'image la dégradation est de plus en plus visible au fur et à mesure qu'augmente le taux de compression et il convient de procéder avec prudence.

De nombreuses expressions sont utilisées pour exprimer la distinction entre les compressions « sans perte d'information » et celles « avec perte d'information ». En voici une liste non exhaustive:

SANS PERTE D'INFORMATION	AVEC PERTE D'INFORMATION
sans dégradation	avec dégradation
sans altération	avec altération
sans destruction	avec destruction
sans déperdition	avec déperdition
avec conservation	sans conservation
Conservative	non conservative
non destructive	Destructive
non dégradé	Dégradé
non altéré	Altéré
Entropique	non entropique

Tableau II. 1: la différence entre les compressions « sans perte d'information » et « avec perte d'information ». [s15]

On parle par exemple d'algorithme non destructif, de compression conservative et d'image non dégradée, la littérature présente toutes ces combinaisons pour dire la même chose.

Les compressions de programmes, de données textuelles ou numériques sont sans perte d'information, il en est autrement lorsque l'on transmet des images.

La compression avec perte va donc éliminer certaines données afin d'approcher au maximum les meilleurs taux de compression. Certaines méthodes contiennent même des

algorithmes d'élaboration d'heuristique afin d'obtenir un taux de compression le meilleur possible, pour des dégradations d'images les plus imperceptibles possibles (exploitation des caractéristiques perceptives visuelles humaines). [s15]

7. Paramètres de performance des méthodes de compression d'images

Les principaux paramètres qui permettent d'évaluer une méthode de compression sont : le taux de compression, le débit, le temps d'exécution et les mesures de distorsions. En pratique, il s'agit de réaliser le meilleur compromis possible entre le taux de compression et la qualité de l'image décompressée tout en minimisant le temps d'exécution.

7.1. Taux de compression

Le taux de compression est une mesure de la performance d'un algorithme de compression de données. Il est donné par l'équation suivante :

Taux compression = 100 * (taille-img-origine _ taille-img-compressé)/ (taille-img-origine)

Le temps de compression / décompression

Le temps de compression/décompression nécessaire pour coder/décoder une image est fonction de la complexité de l'algorithme, de l'efficacité de son implémentation et de la puissance du processeur.

7.2. Les mesures de distorsions

Les techniques de compression avec pertes modifient l'image en y introduisant des distorsions. Pour évaluer la qualité de l'image décompressée, il faut donc estimer le niveau de ces distorsions.

8. Algorithmes sans pertes

8.1. Pixel Packing

Ce n'est pas exactement une méthode de compression de données mais plutôt une manière d'enregistrer des informations de manière contiguës en mémoire. Beaucoup de formats bitmaps utilisent le Pixel Packing pour conserver la quantité de mémoire ou de

place disque dont ils ont besoin pour stocker une image. Si vous travaillez avec des images dont les pixels contiennent 4 bits dans un byte de mémoire, on trouvera commode d'enregistrer chaque pixel dans un byte de mémoire, car un byte est typiquement la plus petite portion de mémoire adressable sur la plupart des systèmes informatiques. Toutefois, on s'apercevra vite qu'en procédant de la sorte, la moitié de chaque byte ne sera pas utilisée par les données du pixel. Une image contenant 4096 pixels contenant 4 bit par pixels prendra donc 4096 bytes de mémoire de stockage dont la moitié sera irrémédiablement perdus.

Pour sauvegarder de la mémoire on peut alors faire appel au Pixel Packing. Au lieu de sauver une seule fois 4 bits par byte, on peut sauver 2 fois 4 bits par byte. La taille alors utilisée pour contenir notre image passera de 4096 à 2048 bytes puisque cette fois-ci la totalité de la mémoire sera utilisée sans pertes. On n'a donc 2 fois moins de mémoire utilisée qu'auparavant.

Le Pixel Packing n'est pourtant pas la panacée. Ce gain de mémoire à un coût en ce qui concerne la vitesse d'exécution. En effet, il faut savoir que la plupart des systèmes organisent leurs données concernant les images en tableaux de bytes contenant chacun un pixel au moins. Si c'est le cas, il serait en effet plus rapide d'enregistrer qu'une fois 4 bits/pixels par byte et de lire directement ces données en mémoire dans l'ordre préétabli que d'enregistrer deux fois 4 bits/pixels par byte ce qui demanderait de masquer et décaler chaque byte de donnée pour extraire et écrire la valeur du pixel approprié. Soit on a un gain de vitesse en matière de lecture/écriture, soit on a un gain en matière de place utilisée, mais pas les deux à la fois. [4]

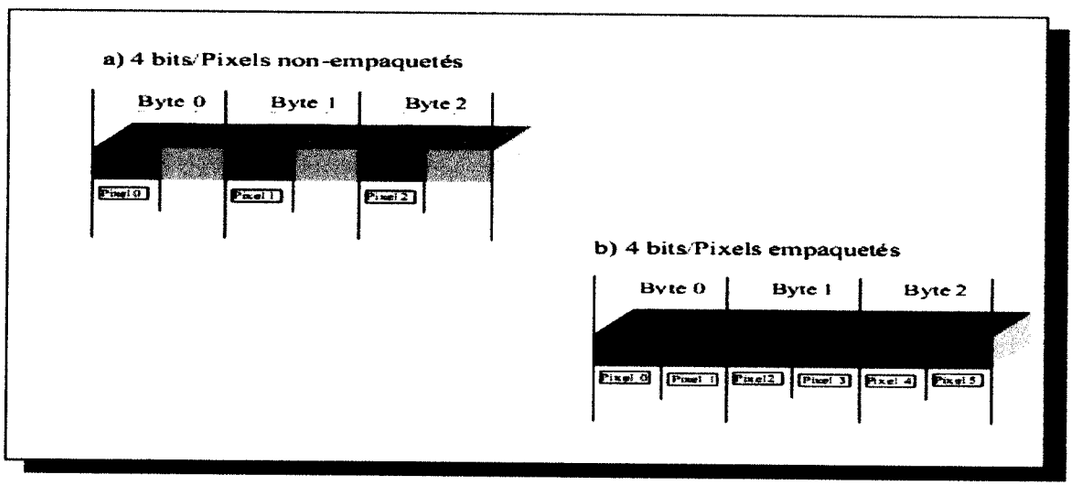


Figure II. 3: Pixel Parcking.

8.2. Run-length Encoding (RLE)

C'est une méthode utilisée par de nombreux formats d'images (BMP, PCX, TIF). Elle est basée sur la répétition de bits consécutifs. Une première valeur (codée sur un octet) donne le nombre de répétitions, une seconde la valeur à répéter (codée elle aussi sur un octet.

La phrase suivante "aaaaahhhhhhhhhhhhhh" donnerait "5a14h", elle est très utile dans ce cas là. Par contre dans "salut" cela donne "1s1a1l1u1t", elle s'avère ici très coûteuse...

Malgré tout cette méthode est peu difficile à mettre en œuvre. Il existe des variantes dans lesquelles on encodera l'image [s16] [12].

➢ **Variantes de RLE**

Les données sont normalement traitées dans un processus séquentiel qui prend des flots de données unidimensionnels plutôt que bidimensionnel. Dans un processus séquentiel, un bitmap est codé en commençant dans le coin en haut à gauche et en continuant de gauche à droite à travers chaque ligne jusqu'au coin inférieur droit de l'image (figure II.4, a). Mais des schémas alternatifs de RLE peuvent être écrits de telles manières qu'ils encodent les données de manière verticale, colonne par colonne (figure II.4, b) ou bien par flots de 4 x 4 pixels (figure II.4, c) ou encore en zigzag (figure II.4, d). De singulières variantes comme cette dernière sont utilisées dans des applications très spécialisées mais sont relativement rares.

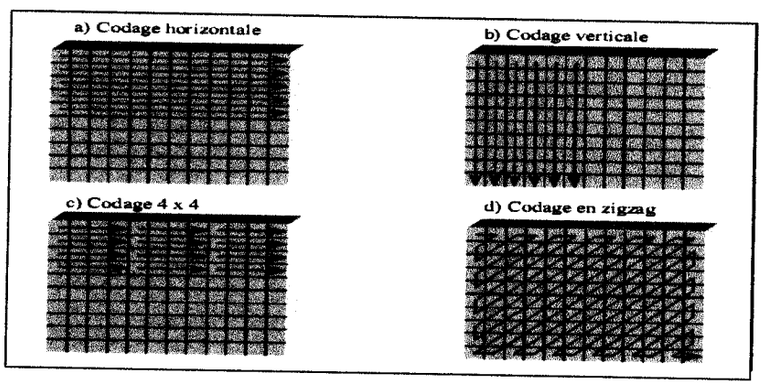


Figure II. 4: codage RLE. [s16]

8.3. La compression Lempel-Ziv-Welch (LZW)

La meilleure méthode de compression est aujourd'hui LZW. C'est la plus connue, mais aussi la moins utilisée. En effet, la méthode LZW est protégée par un brevet. Les compacteurs tels que ARJ ou Winzip, et certains formats tels que le format d'images GIF doivent donc se contenter de la méthode LZ, moins performante. La méthode LZW est employée sous UNIX, sous WINDOWS et avec la plupart des systèmes d'exploitation.

La méthode LZ a été inventée en 1978 par Lempel et Ziv, puis terminée en 1984 par Welch, ce qui fit alors la méthode LZW. Elle pose quelques problèmes dans la gestion de ses algorithmes, notamment dans la gestion de la mémoire, et du dictionnaire. C'est pour les résoudre, et améliorer les performances, que des variantes de LZW ont été inventées. On parle beaucoup de certaines variantes comme LZ77, LZ78, ou encore GZIP qui se disent plus performante, ou plus rapides. [s16]

La méthode LZW a certaines caractéristiques:

- Elle utilise un dictionnaire qu'elle construit dynamiquement, au cours de la compression, et de la décompression, qui n'est pas stocké dans le fichier compressé.
- Elle comprime en une seule lecture.
- Elle a besoin d'un apprentissage pour être efficace, et reconnaître des longues chaînes répétées. Elle est donc peu performante sur des les petits fichiers. [s17]

Exemple :

L'exemple suivant illustre l'algorithme LZW dans l'action, montrant l'état de la sortie et le ***dictionnaire*** à chaque étape, à la fois dans le codage et le décodage des données. Cet exemple a été construit pour donner une compression raisonnable sur un message très court. Dans les données de texte réel, la répétition est généralement moins prononcés, les flux d'entrée pour plus sont généralement nécessaires avant la compression s'accumule efficacité.

- Le texte en clair d'être encodé (à partir d'un alphabet en utilisant seulement les lettres majuscules) est:

TOBEORNOTTOBEORTOBEORNOT #

Le # est un marqueur utilisé pour montrer que la fin du message a été atteint. Il ya donc 26 symboles de l'alphabet clair (les 26 lettres majuscules *de A* à *Z),* plus le code d'arrêt #. Nous arbitrairement attribuer ces valeurs de 1 à 26 pour les lettres, et 0 pour «#». (La plupart des saveurs de LZW mettrait le code d'arrêt *après* l'alphabet de données, mais rien dans l'algorithme de base exige que. Le codeur et le décodeur ne doivent convenir quelle valeur il a.)

Un ordinateur rendront ces comme des chaînes de bits de . Cinq codes bits sont nécessaires pour donner des combinaisons suffisantes pour englober cet ensemble de 27 valeurs. Le dictionnaire est initialisé avec ces 27 valeurs. Comme le dictionnaire grandit, les codes aura besoin de croître en largeur pour accueillir les entrées supplémentaires. Un code à 5 bits donne $2^5 = 32$ combinaisons possibles de bits, donc quand le mot du dictionnaire 33e créée, l'algorithme devra passer à ce point à partir de 5 chaînes de bits à 6 bits cordes (pour *toutes les* valeurs de code, y compris ceux qui étaient auparavant de sortie avec seulement cinq bits). A noter que depuis le code-zéro tous les 00 000 est utilisé, et est étiqueté "0", l'entrée du dictionnaire 33e sera marqué **32.** (Sortie précédemment généré n'est pas affectée par le changement du code de largeur, mais une fois une valeur de 6 bits est générée dans le dictionnaire, il pourrait en théorie être le prochain code émis, alors la largeur de la sortie ultérieure déplace à 6 bits pour accommoder.)

Le dictionnaire initial, puis, se compose des entrées suivantes:

Symbole	Binaires	Décimal
#	00000	0
Une	00001	1
B	00010	2
C	00011	3
D	00100	4
E	00101	5
F	00110	6
G	00111	7
H	01000	8
J'ai	01001	9
J	01010	10
K	01011	11

L	01100	12
M	01101	13
N	01110	14
O	01111	15
P	10000	16
Q	10001	17
R	10010	18
S	10011	19
T	10100	20
U	10101	21
V	10110	22
W	10111	23
X	11000	24
Y	11001	25
Z	11010	26

Encodage

Caractères tampon d'entrée dans une séquence jusqu'à ω ω + caractère suivant n'est pas dans le dictionnaire. Emit le code pour ω, ω + et ajouter le caractère suivant dans le dictionnaire. Commence la bufferisation de nouveau avec le caractère suivant.

Séquence actuelle	**Suivant Char**	**Sortie**		**Dictionnaire étendue**	**Commentaires**
		Code de	**Bits**		
NULL	T				
T	O	20	10100	27: À	27 = premier code disponible après 0 à 26
O	B	15	01111	28: OB	
B	E	2	00010	29: ÊTRE	
E	O	5	00101	30: OT	
O	R	15	01111	31: OU	
R	N	18	10010	32: RN	32 nécessite 6 bits, afin d'utiliser la sortie prochaine de 6 bits
N	O	14	001110	33: NO	

O	T	15	001111	34: OT	
T	T	20	010100	35: TT	
À	B	27	011011	36: TOB	
ÊTRE	O	29	011101	37: BEO	
OU	T	31	011111	38: ORT	
TOB	E	36	100100	39: TOBE	
OT	R	30	011110	40: RAH	
RN	O	32	100000	41: RNO	
OT	#	34	100010		# Arrête l'algorithme; envoyer les suivants actu
		0	000000		et le code d'arrêt

Longueur = 25 Unencoded symboles × 5 bits / symbole = 125 bit

Length Encoded = (6 codes × 5 bits / code) + (11 × 6 bits codes / code) = 96 bits.

En utilisant LZW a sauvé 29 bits sur 125, ce qui réduit le message de près de 22%. Si le message était plus long, alors les mots du dictionnaire commenceraient à représenter des sections plus longues et plus de texte, permettant mots répétés à être envoyé très compacte.

Décodage

Pour décoder une archive compressées en LZW, on a besoin de savoir à l'avance le dictionnaire initial utilisé, mais les entrées supplémentaires peuvent être reconstruits car ils sont toujours tout simplement ***concaténations*** d'entrées précédentes.

Entrée		**Séquence de sortie**	**Nouvelle entrée dans le dictionnaire**		**Commentaires**
Bits	**Code de**		**Complet**	**Conjecture**	
10100	20	T		27: T?	
01111	15	O	27: À	28: O?	
00010	2	B	OB	B?	

			28:	29:	
00101	5	E	29: ÊTRE	30: E?	
01111	15	O	30: OT	31: O?	
10010	18	R	31: OU	32: R?	créé un code 31 (dernière pour tenir dans 5 bits)
001110	14	N	32: RN	33: N?	afin de commencer à utiliser 6 bits
001111	15	O	33: NO	34: O?	
010100	20	T	34: OT	35: T?	
011011	27	À	35: TT	36: À?	
011101	29	ÊTRE	36: TOB	37: BE?	36 = À + 1er symbole (B) de
011111	31	OU	37: BEO	38: OU?	la prochaine séquence codée reçue (BE)
100100	36	TOB	38: ORT	39: TOB?	
011110	30	OT	39: TOBE	40: EO?	
100000	32	RN	40: RAH	41: RN?	
100010	34	OT	41: RNO	42: OT?	
000000	0	#			

A chaque étape, le décodeur reçoit un code X, il ressemble X dans le tableau et les sorties de la séquence, il χ codes, et il conjectures χ +? que l'entrée de l'encodeur venez d'ajouter - car le codeur X émis par χ précisément parce que + χ? n'était pas dans le tableau, et l'encodeur va de l'avant et l'ajoute. Mais quelle est la lettre manquante? Elle est la première lettre dans la séquence codée par le *prochain* Z code que le décodeur

reçoit. Ainsi, le décodeur vérifie Z, décode la séquence dans le ω et prend la première lettre z et punaises c'est sur l'extrémité du χ que l'entrée du dictionnaire prochaine.

Cela fonctionne tant que les codes reçus sont dans le dictionnaire du décodeur, de sorte qu'ils peuvent être décodés en séquences. Que faire si le décodeur reçoit un code de Z qui n'est pas encore dans son dictionnaire? Depuis le décodeur est toujours juste un code derrière le codeur, Z peut être dans le dictionnaire de l'encodeur que si le codeur *simplement* qu'elle a engendrés, lors de l'émission X de code précédent pour χ. ? Ainsi codes Z certaine ω qui est + χ, et le décodeur peut déterminer le caractère inconnu comme suit:

1. Le décodeur voit X, puis Z.
2. Il sait les codes X du χ séquence et les codes Z certaine ω séquence inconnue.
3. Il sait l'encodeur viennent d'être ajoutées au code Z χ + quelques caractères inconnus,
4. et il sait que le caractère inconnu est la première lettre de z ω.
5. Mais la première lettre de ω (χ = +?) Doit alors être également la première lettre de χ.
6. Alors ω doit être χ + x, où **x est la *première* lettre de χ.**
7. Donc les chiffres décodeur quelles codes Z même si ce n'est pas dans le tableau,
8. et dès réception de Z, le décodeur décode que χ + x, et ajoute χ + x à la table comme la valeur de Z.

Cette situation se produit chaque fois que l'encodeur rencontres d'entrée du *CCSCA* forme, où *c* est un caractère unique, *S* est une chaîne *et* CS est déjà dans le dictionnaire, *mais* le SCC est pas. Le codeur émet le code *pour* le CS, en mettant un nouveau code *pour* le SCC dans le dictionnaire. Ensuite, il voit *le SCC* à l'entrée (à partir de la deuxième *c CCSCA)* et émet le nouveau code qu'il venez d'insérer. L'argument ci-dessus montre que chaque fois que le décodeur reçoit un code qui n'est pas dans son dictionnaire, la situation doit ressembler à ceci.

Bien que l'entrée du *CCSCA* formulaire peut sembler improbable, ce modèle est assez courante lorsque le flux d'entrée est caractérisée par la répétition significative. En

particulier, les longues chaines d'un seul caractère (qui sont communs dans les types d'images LZW est souvent utilisé pour encoder) générer plusieurs modèles de ce type.

8.4. Algorithme de Huffman

8.4.1. Présentation

Considérons un fichier texte. La compression de Huffman est un algorithme dit **statistique**. Plus un caractère est fréquent, plus il sera codé sur un nombre de bits faible. D'ordinaire, un caractère (**un unsigned char**) est sur 8 bits. Dans la langue française, la lettre *e*, très courante sera codée sur 2 ou 3 bits tandis qu'un *w*, plus rare utiliseront 8 ou 9 bits par exemple.

Une première méthode vise à utiliser une table générique de fréquence des caractères (voir **tableau II.2**) qui est réutilisée à chaque compression et ce indépendamment du fichier à compresser. Cette table résulte d'une étude statistique de la fréquence d'apparition de chaque lettre. Par exemple, le *e* est crédité de 26% (cela veut dire qu'en moyenne, 26 lettres sur 100 sont des *e* en français). La table fournit alors pour chaque lettre un nombre de bits et un code précis. L'implémentation de cette méthode est triviale puisqu'il suffit d'enregistrer la table d'occurrences dans le programme.

Cette manière de procéder est la plus rapide mais perd en efficacité si le texte est en Espagnol où la lettre *o* est beaucoup plus courante que le *e* ... On voit alors qu'une étude statistique sur le texte à compresser, visant à créer une table d'occurrences propre au fichier, se révèle meilleure que l'utilisation d'une table générique. Le revers de la médaille est l'introduction d'un temps de calcul supplémentaire et le rajout d'un en-tête dans le fichier compressé. C'est ce point de vue que nous avons choisi d'aborder pour mener à bien le développement du programme.

Il existe une troisième méthode qui consiste à construire l'arbre binaire au fur et à mesure de la lecture et de l'adapter en conséquence. Cet algorithme permet de s'affranchir de l'en-tête mais l'actualisation de l'arbre rend le processus très lent s'il est mal optimisé. La littérature accorde le terme d'algorithme de Huffman *adaptatif*.

Lettres	Occurrences	Fréquence	CODAGE
C	1	5	00000
G	1	5	00001
H	1	5	00010
L	1	5	00011
N	1	5	00100
O	1	5	00101
P	1	5	00110
U	1	5	00111
X	1	4	1000
A	2	4	1001
F	2	4	0100
M	2	4	0101
D	3	3	101
[ESPACE]	4	3	011
E	6	2	11

Tableau II. 2: Exemple de table d'occurrences.

8.4.2. Etapes suivies pour la compression/décompression.

- **Compression :**

1) Etude statistique.
2) Détermination de la table des codes et nombre de bits pour chaque caractère.
3) Ecriture de la table en en-tête du fichier compressé.
4) Encodage du fichier compressé, caractère par caractère, compte tenu de la table.

- **Décompression** :

1) Reconstruction de la table grâce à l'en-tête.
2) Décodage et reconstruction du fichier non compressé, caractère par caractère.

➢ **Quelques remarques**

- Comme la table dépend du fichier à compresser, il est essentiel de pouvoir la retrouver lors de la décompression. C'est pour cette raison que l'on place cette table en début du fichier compressé afin de pouvoir décompresser en toute sérénité. Le but est de limiter le plus possible la place prise par cet en-tête afin de ne pas trop augmenter la taille du fichier compressé.
- L'algorithme fonctionnera sur des fichiers contenant des caractères ASCII (7 bits) ou Extended ASCII (8 bits). Dans la présentation, on supposera néanmoins que l'on travaille avec des *.txt pour des raisons évidentes de clarté.

8.4.3. Création de la table : arbre de Huffman

Raisonnons sur un exemple : soit à compresser le texte :

«GoldenEye et Pascal programment ensemble»

En ignorant la casse, on obtient la table d'occurrences suivante :

Lettre	Nombre d'occurrences	Fréquence
O	2	5%
L	3	7.5%
D	1	2.5%
E	8	20%
N	3	7.5%
Y	1	2.5%
T	2	5%
P	2	5%
A	3	7.5%
S	2	5%
C	1	2.5%
R	2	5%
M	3	7.5%
G	2	5%

B	1	2.5%
[SPC]	4	10%
TOTAL	**40**	**100%**

Tableau II. 3: exemple.

Nous allons maintenant créer l'arbre de Huffman, une structure de données qui nous permettra d'accorder un code à chaque lettre, sur un certain nombre de bits, inversement proportionnel à la fréquence de la lettre. Pour cela, commençons par trier, par ordre croissant de fréquence, les caractères. Ces derniers sont considérés comme des arbres (figure II.6).

Figure II. 5: Construction de l'arbre de Huffman « Etape 1 ».

La construction de l'arbre est aisée à comprendre. On prend les deux derniers arbres (D et C) de la liste, on les fusionne dans un troisième de fréquence somme des 2 précédents et on l'insère dans la liste triée. (Voir figure II.7).

Figure II. 6: Construction de l'arbre de Huffman « Etape 2 ».

On réitère le même procédé tant que le nombre d'éléments de la liste d'arbres est strictement supérieur à 1 (voir figure II.8, II.9, II.10).

Figure II. 7: Construction de l'arbre de Huffman « Etape 3 ».

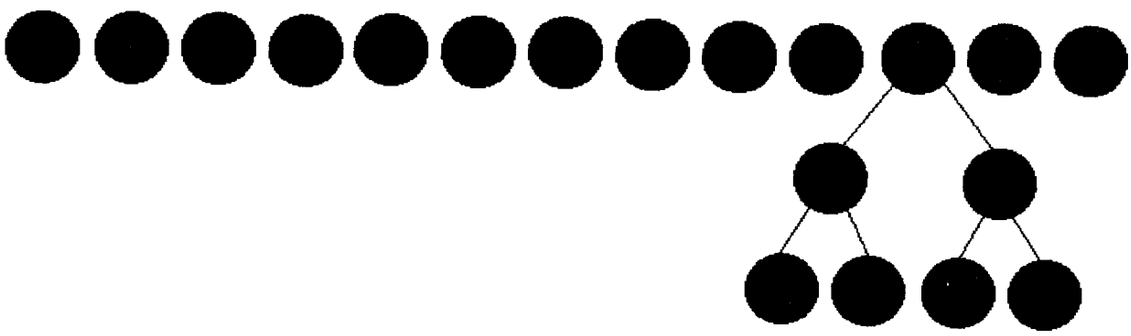


Figure II. 8: Construction de l'arbre de Huffman « Etape 4 ».

Voici l'arbre final. Le lien de gauche reçoit 0 tandis que celui de droite a la valeur 1.

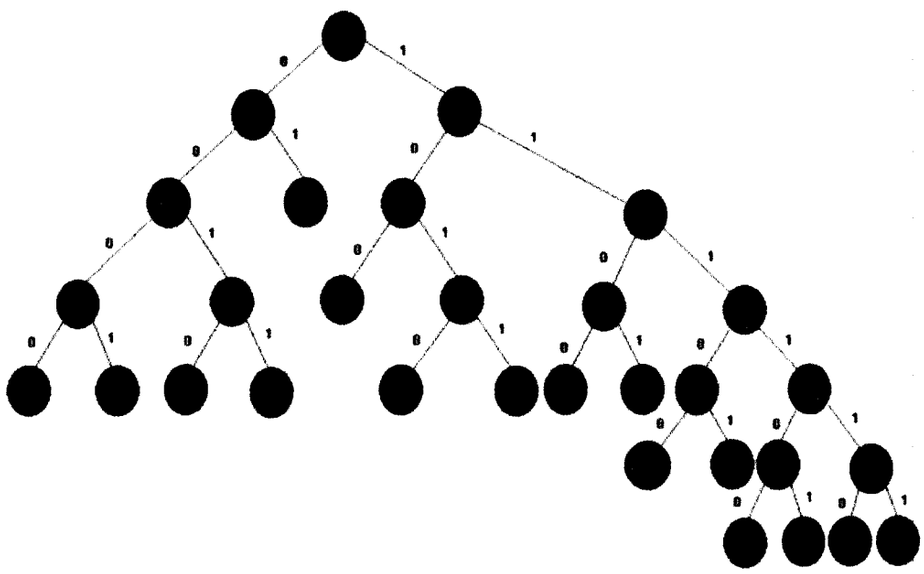


Figure II. 9: Construction de l'arbre de Huffman «Résultat final».

On remarque que plus un caractère est rare, plus il faut parcourir d'étages dans l'arbre de Huffman. Cela correspond bien à l'idée de compression statistique puisqu'un étage représente 1 bit de codage. La lettre *e*, la plus courante, apparaît au deuxième étage (on part de la racine qui fait office de rez de chaussée) d'où un codage sur 2 bits.

Résumons l'ensemble des résultats dans un tableau (tableau II.4):

Caractère	Fréquence	Nombre de bits	Code de Huffman
O	5%	4	0001
L	7.5%	4	1101
D	2.5%	6	111110
E	20%	2	01
N	7.5%	4	1100
Y	2.5%	6	111101
T	5%	4	0000
P	5%	4	0011
A	7.5%	4	1011
S	5%	4	0010
C	2.5%	6	111111
R	5%	5	11101
M	7.5%	4	1010
G	5%	5	11100
B	2.5%	6	111100
[SPC]	10%	3	100

Tableau II. 4: Table de codage de Huffman.

8.4.4. Résultat de la compression (sans en-tête)

Le texte original comprenait 40 caractères, des *char* codés sur un octet (8 bits). Par conséquent la taille du fichier était de 40 octets (320 bits). Compte tenu de la table ci-dessus, on obtient le résultat suivant (tableau II.5):

Caractère	**G**	**O**	**L**	**D**	**E**	**N**	**E**	**Y**	**E**		**E**	**T**		**P**	**A**
Nb de bits	5	4	4	6	2	4	2	6	2	3	2	4	3	4	4
Cumulation	5	9	13	19	21	25	27	33	35	38	40	44	47	51	55

S	**C**	**A**	**L**		**P**	**R**	**O**	**G**	**R**	**A**	**M**	**M**	**E**	**N**	**T**		**E**

4	6	4	4	3	4	5	4	5	5	4	4	4	2	4	4	3	2
59	65	69	73	76	80	85	89	94	99	103	107	111	113	117	121	124	126

N	**S**	**E**	**M**	**B**	**L**	**E**
4	4	2	4	6	4	2
130	134	136	140	146	150	**152**

Tableau II. 5: Résultat de la compression.

On remarque que la taille du texte compressé est de 152 bits soit moins de la moitié du texte initial (320 bits pour mémoire) !

9. Algorithmes à pertes

9.1. JPEG, RVQ et compression fractale

Aujourd'hui, trois méthodes de compression d'images se partagent l'essentiel du marché. Elles ont pour nom: JPEG (Joint Photographic Experts Group), RVQ (RecursiveVector Quantization) et compression fractale.

9.1.1. La technologie JPEG

La technologie JPEG est souvent incorporée dans des produits destinés à l'utilisateur final (les éditeurs graphiques notamment), car la méthode de compression est simple à programmer et surtout relativement rapide. Plusieurs paramètres permettent de jouer sur la qualité d'image et par la même sur le taux de compression. Plus la qualité est élevée, moins efficace est la compression et inversement. Cette technologie présente un inconvénient : pour un degré de qualité donné, la taille d'un fichier JPEG est souvent supérieure à celle que permettent d'obtenir les procédés rivaux.

9.1.2. La méthode RVQ

A l'origine, la méthode RVQ à été mise au point par la société Knowledge Adventure spécialisée dans les logiciels éducatifs interactifs. Non seulement elle exige une puissance de calcul considérable (même en comparaison des deux autres méthodes de

compression approximative), mais encore elle nécessite une intervention humaine à chaque étape afin de choisir les meilleurs paramètres de compression. Si la compression est un processus lent, la décompression en revanche est rapide.
Cette rapidité rend RVQ utilisable pour la décompression d'informations en temps réel, comme des données audio ou vidéo.

9.1.3. La compression fractale

La compression fractale est l'apanage de la société Iterated Systems. L'idée étant qu'une image est stockée sous la forme d'un ensemble de formules plutôt que d'un ensemble de points (ou bien de transformations d'ensembles de points). Ainsi l'apparence d'une image dépend uniquement du périphérique de sortie, car la compression fractale est indépendante de la résolution. Les principes mathématiques mis en œuvre pour la compression sont extrêmement complexes, mais à la manière des procédés JPEG et RVQ, la décompression est raisonnablement rapide.
Ainsi avec ces trois méthodes, l'efficacité de la compression dépendra de la nature des images traitées : seule l'expérience décidera.

10. Conclusion

La compression des données est appelée à prendre un rôle encore plus important en raison du développement des réseaux et du multimédia. Son importance est surtout due au décalage qui existe entre les possibilités matérielles des dispositifs que nous utilisons (débits sur Internet, sur Numéris et sur les divers câbles, capacité des mémoires de masse,...) et les besoins qu'expriment les utilisateurs (visiophonie, vidéo plein écran, transfert de quantités d'information toujours plus importantes dans des délais toujours plus brefs). Quand ce décalage n'existe pas, ce qui est rare la compression permet de toute façon des économies. Les méthodes déjà utilisées couramment sont éfficaces et sophistiquées (Huffman, LZW, JPEG) et utilisent des théories assez complexes. Les méthodes du futur sauront sans doute s'adapter à la nature des données à compresser et utiliseront l'intelligence artificielle.

CHPITRE III

Techniques de Cryptage

compression approximative), mais encore elle nécessite une intervention humaine à chaque étape afin de choisir les meilleurs paramètres de compression. Si la compression est un processus lent, la décompression en revanche est rapide.
Cette rapidité rend RVQ utilisable pour la décompression d'informations en temps réel, comme des données audio ou vidéo.

9.1.3. La compression fractale

La compression fractale est l'apanage de la société Iterated Systems. L'idée étant qu'une image est stockée sous la forme d'un ensemble de formules plutôt que d'un ensemble de points (ou bien de transformations d'ensembles de points). Ainsi l'apparence d'une image dépend uniquement du périphérique de sortie, car la compression fractale est indépendante de la résolution. Les principes mathématiques mis en œuvre pour la compression sont extrêmement complexes, mais à la manière des procédés JPEG et RVQ, la décompression est raisonnablement rapide.
Ainsi avec ces trois méthodes, l'efficacité de la compression dépendra de la nature des images traitées : seule l'expérience décidera.

10. Conclusion

La compression des données est appelée à prendre un rôle encore plus important en raison du développement des réseaux et du multimédia. Son importance est surtout due au décalage qui existe entre les possibilités matérielles des dispositifs que nous utilisons (débits sur Internet, sur Numéris et sur les divers câbles, capacité des mémoires de masse,...) et les besoins qu'expriment les utilisateurs (visiophonie, vidéo plein écran, transfert de quantités d'information toujours plus importantes dans des délais toujours plus brefs). Quand ce décalage n'existe pas, ce qui est rare la compression permet de toute façon des économies. Les méthodes déjà utilisées couramment sont éfficaces et sophistiquées (Huffman, LZW, JPEG) et utilisent des théories assez complexes. Les méthodes du futur sauront sans doute s'adapter à la nature des données à compresser et utiliseront l'intelligence artificielle.

1. Introduction

L'homme a toujours ressenti le besoin de dissimuler des informations, bien avant même l'apparition des premiers ordinateurs et de machines à calculer. Depuis sa création, le réseau Internet a tellement évolué qu'il est devenu un outil essentiel de communication. Cependant cette communication met de plus en plus en jeux des problèmes d'économie des entreprises présentes sur le Web. Les transactions faites à travers le réseau peuvent être interceptées, d'autant plus que les lois, ont du mal à se mettre en place sur Internet, il faut donc garantir la sécurité de ces informations, c'est la cryptographie qui s'en charge.

2. Qu'est-ce que la cryptographie?

Le mot **cryptographie** est un terme générique désignant l'ensemble des techniques permettant de **chiffrer** des messages, c'est-à-dire permettant de les rendre inintelligibles sans une action spécifique. Le verbe **crypter** est parfois utilisé mais on lui préfèrera le verbe *chiffrer*. La cryptologie est essentiellement basée sur l'arithmétique. Il s'agit dans le cas d'un texte de transformer les lettres qui composent le message en une succession de chiffres (sous forme de bits dans le cas de l'informatique car le fonctionnement des ordinateurs est basé sur le binaire), puis ensuite de faire des calculs sur ces chiffres pour:

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

Le fait de coder un message de telle façon à le rendre secret s'appelle *chiffrement*. La méthode inverse, consistant à retrouver le message original, est appelé *déchiffrement*. Le chiffrement se fait généralement à l'aide d'une *clef de chiffrement*, le déchiffrement nécessite quant à lui une *clef de déchiffrement*. On distingue généralement deux types de clefs :

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

On appelle *décryptement* (le terme de *décryptage* peut éventuellement être utilisé également) le fait d'essayer de *déchiffrer illégitimement* le message (que la clé de déchiffrement soit connue ou non de l'*attaquant*). Lorsque la clef de déchiffrement n'est pas connue de l'attaquant on parle alors de **cryptanalyse** ou **cryptoanalyse** (on entend souvent aussi le terme plus familier de *cassage*). La **cryptologie** est la science qui étudie les aspects scientifiques de ces techniques, c'est-à-dire qu'elle englobe la cryptographie et la cryptanalyse. [s18][14][15]

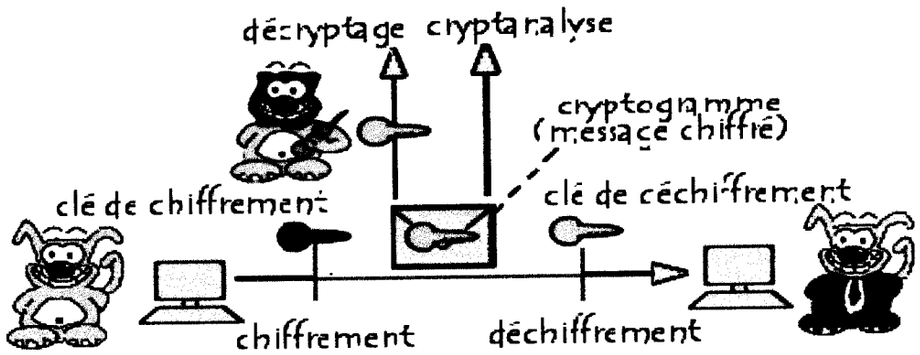


Figure III. 1 : Principes de base de la cryptologie [s18]

3. Les fonctions de la cryptographie

La cryptographie est traditionnellement utilisée pour dissimuler des messages aux yeux de certains utilisateurs. Cette utilisation a aujourd'hui un intérêt d'autant plus grand que les communications via Internet circulent dans des infrastructures dont on ne peut garantir la fiabilité et la confidentialité. Désormais, la cryptographie sert non seulement à préserver la confidentialité des données mais aussi à garantir leur **intégrité** et leur **authenticité**. [5]

3.1 Objectifs de la cryptographie

Les principaux services offerts par la cryptographie moderne sont:

Confidentialité : assurer que les données concernées ne pourront êtres dévoilées que par les personnes autorisées.

Intégrité : Assurer que les données ne seront pas altérées pendant leur transmission ou leur stockage.

Authentification/Identification : Prouver l'origine d'une donnée ou s'assurer de l'identité d'une personne.

Non répudiation : Garantir que les actions ne seront pas reniées.

En plus de ces quatre objectifs fondamentaux, on peut citer :

- **Signature :** Lier une information à une entité.
- **Autorisation :** Lever une restriction d'accès à une information.
- **Contrôle d'accès :** Restreindre l'accès qu'aux entités privilégiées.
- **Anonymat :** Préserver l'identité d'une entité de la divulgation. [s19]

4. Les Méthodes de Chiffrement (Codage)

Une méthode de chiffrement est un algorithme de cryptage défini par une fonction mathématique associée à une clé, qui est séquence de caractères.

On transforme le message initial en un message en faisant subir à chaque caractère une transformation dépendant d'une clé de cryptage.

Exemple simplicité sur des lettres (en remplaçant chaque lettre par son n° d'ordre dans l'alphabet):

MESSAGE DE BASE:	B	I	E	N	V	E	N	U	E
CLÉ DE CRYPTAGE:	H	E	L	L	O	H	E	L	L
CHIFFREMENT	2	9	5	14	22	5	14	21	5
	8	5	12	12	15	8	5	12	12
TOTAL (MODOLO 27):	10	14	17	26	10	13	19	6	17
RESULTAT:	J	N	Q	Z	J	M	S	F	Q

En réalité, cela se fera par des opérations plus complexes, sur des bits, avec des clés binaires qui ont un certain nombre de bits. Les algorithmes (méthodes) sont différents, mais pas les clés.

5. Les Méthodes Cryptographiques

Il existe deux grandes familles d'algorithmes cryptographiques à base de clefs : les algorithmes à clef secrète ou algorithmes symétriques, et les algorithmes à clef publique ou algorithmes asymétriques.

5.1 Chiffrement symétrique ou à clef secrète

Dans la cryptographie conventionnelle, les **clefs de chiffrement et de déchiffrement sont identiques** : c'est la clef secrète, qui doit être connue des tiers communiquant et d'eux seuls. Le procédé de chiffrement est dit symétrique.

Les algorithmes symétriques sont de deux types :

- les algorithmes de **chiffrement en continu ou par flot**, qui agissent sur le texte en clair un bit à la fois.
- les algorithmes de **chiffrement par blocs**, qui opèrent sur le texte en clair par groupes de bits appelés blocs. [s20]

5.1.1 Modes de chiffrement par flot

Pour comprendre le cryptage en continu, il suffit de connaître par exemple les vidéos au format RealVideo très répandues sur Internet : on visualise l'image au fur et à mesure que les données sont reçues. Le principe est le même dans le cas de nos "stream-ciphers" : le cryptage est effectué bit-à-bit sans attendre la réception complète des

données à crypter.

Une technique de chiffrement, du nom de "One-Time Pad" est utilisée pour chiffrer les flux. C'est le chiffrement inconditionnel le plus sûr. Pour cela, on a besoin d'une chaîne aléatoire de la même longueur que le message d'origine, ce qui n'est pas pratique. Le but d'un stream cipher est de générer une chaîne aléatoire à partir d'une clé de longueur courte.

Une autre technique consiste à "xorer", c'est-à-dire à appliquer un OU exclusif (XOR) au message avec un autre message prédéfini. Bien entendu, cela nécessite que le destinataire (la personne qui décrypte) connaisse le message prédéfini et donc cela rajoute de la complexité au schéma général.

Les stream-ciphers sont utilisés aujourd'hui par différentes applications. Pour chiffrer les flux, l'algorithme RC4 est très utilisé. [s20]

5.1.2 Modes de chiffrement par blocs

Les algorithmes de chiffrement par blocs peuvent être utilisés suivant différents modes, dont les deux principaux sont le mode ECB (*Electronic CodeBook*) et le mode CBC (*Cipher Block Chaining*).

Le mode ECB (Electronic CodeBook)

Le mode du "carnet de codage électronique" (*Electronic CodeBook*) est la méthode la plus évidente pour utiliser un algorithme de chiffrement par blocs : un bloc du texte en clair se chiffre, indépendamment de tout, en un bloc de texte chiffré. L'avantage de ce mode est qu'il permet le chiffrement en parallèle des différents blocs composant un message.

L'inconvénient de ce mode est qu'un même bloc de texte en clair sera toujours chiffré en un même bloc de texte chiffré. Or, dans le chiffrement sur un réseau par exemple, les données à chiffrer ont des structures régulières facilement repérables par un cryptanalyste, qui pourra donc obtenir beaucoup d'informations. D'autre part, un attaquant actif pourra facilement manipuler les messages chiffrés en retirant, répétant ou inter changeant des blocs. Un autre inconvénient, qui s'applique au chiffrement par blocs en général, est l'amplification d'erreur : si un bit du texte chiffré est modifié pendant le transfert, tout le bloc de texte en clair correspondant sera faux.

Le mode CBC (Cipher Block Chaining)

La solution aux problèmes posés par le mode ECB est d'utiliser une technique dite de chaînage, dans laquelle chaque bloc du cryptogramme dépend non seulement du bloc de texte en clair correspondant, mais aussi de tous les blocs précédents.

En mode de "chiffrement avec chaînage de blocs" (*Cipher Block Chaining*), chaque bloc de texte en clair est combiné par *ou exclusif* avec le bloc chiffré précédent avant d'être chiffré. Le premier bloc du texte en clair est, quant à lui, combiné avec un bloc appelé *vecteur d'initialisation.* L'utilisation d'un vecteur d'initialisation différent pour chaque message permet de s'assurer que deux messages identiques (ou dont les premiers blocs sont identiques) donneront des cryptogrammes totalement différents.

Le gros avantage du mode CBC est donc que la structure du texte en clair est masquée par le chaînage. Un attaquant ne peut plus manipuler le cryptogramme, excepté en retirant des blocs au début ou à la fin. Un inconvénient est qu'il n'est plus possible de paralléliser le chiffrement des différents blocs (le déchiffrement reste parallélisable).

On pourrait craindre que le chaînage de bloc n'entraîne une propagation d'erreur importante. De fait, une erreur d'un bit sur le texte en clair affectera tous les blocs chiffrés suivants. Par contre, si un bit du texte chiffré est modifié au cours du transfert, seul le bloc de texte en clair correspondant et un bit du bloc de texte en clair suivant seront endommagés : le mode CBC est dit auto-réparateur. [s20]

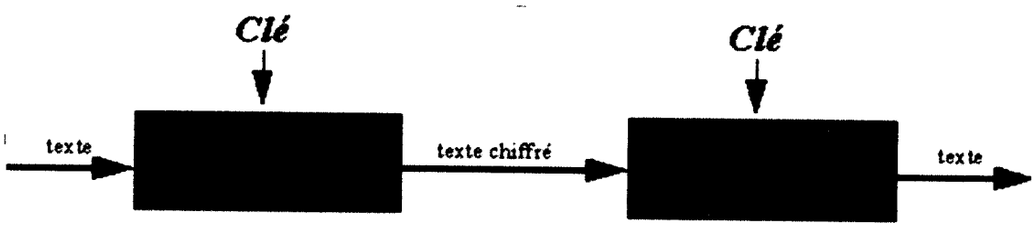


Figure III. 2 : Chiffrement symétrique. [s19]

5.2 Chiffrement asymétrique ou à clef publique

Avec les algorithmes asymétriques, les clefs de chiffrement et de déchiffrement sont distinctes et ne peuvent se déduire l'une de l'autre. On peut donc rendre l'une des deux publique tandis que l'autre reste privée. C'est pourquoi on parle de chiffrement à clef publique. Si la clef publique sert au chiffrement, tout le monde peut chiffrer un message, que seul le propriétaire de la *clef privée* pourra déchiffrer. On assure ainsi la confidentialité. Certains algorithmes permettent d'utiliser la clef privée pour chiffrer. Dans

ce cas, n'importe qui pourra déchiffrer, mais seul le possesseur de la clef privée peut chiffrer. Cela permet donc la signature de messages.

Le concept de cryptographie à clef publique fut inventé par Whitfield Diffie et Martin Hellman en 1976, dans le but de résoudre le problème de distribution des clefs posé par la cryptographie à clef secrète. De nombreux algorithmes permettant de réaliser un cryptosystème à clef publique ont été proposés depuis. Ils sont le plus souvent basés sur des problèmes mathématiques difficiles à résoudre, donc leur sécurité est conditionnée par ces problèmes, sur lesquels on a maintenant une vaste expertise. Mais, si quelqu'un trouve un jour le moyen de simplifier la résolution d'un de ces problèmes, l'algorithme correspondant s'écroulera.

Nombre des algorithmes proposés pour la cryptographie à clef publique se sont révélés rapidement non sûrs, ou non réalisables sur le plan pratique. Tous les algorithmes actuels présentent l'inconvénient d'être bien plus lents que les algorithmes à clef secrète ; de ce fait, ils sont souvent utilisés non pour chiffrer directement des données, mais pour chiffrer une *clef de session* secrète. Certains algorithmes asymétriques ne sont adaptés qu'au chiffrement, tandis que d'autres ne permettent que la signature. Seuls trois algorithmes sont utilisables à la fois pour le chiffrement et pour la signature : RSA, ElGamal et Rabin. [s20]

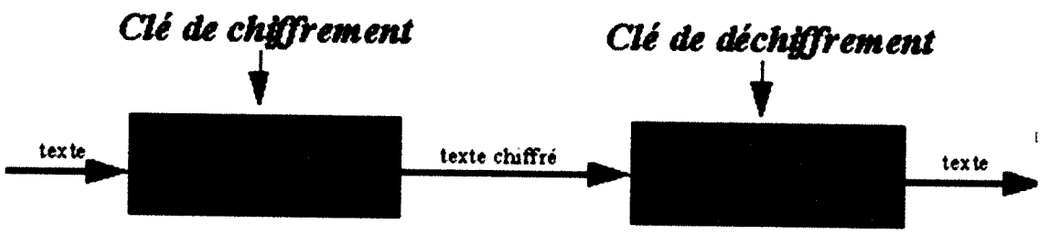


Figure III. 3: Chiffrement asymétrique. [s19]

5.3 Avantages et inconvénients des deux types de cryptographie :

Le tableau suivant résume les avantages et inconvénients des deux types de systèmes de cryptage.

Type	Avantages	Inconvénients
Clé symétrique	Rapide, Matériellement facile à implémenter	Les deux clés sont les mêmes, Difficile à distribuer,
Clé asymétrique	Utilise des clés différentes, Clés faciles à distribuer, Assure l'intégrité et l'accusé de réception	Lent et gourmand en ressource

Tableau III. 1: Avantages et inconvénients des deux types de cryptographie.

6. La distribution des clés

Un des problèmes principaux en matière de cryptographie est la distribution des clés, c'est à dire comment faire pour qu'une clé ne soit connue que par les personnes concernées. Il faut transmettre ces clés en toute sécurité, même lorsque les individus sont éloignés.

La divulgation de la clé, pour la cryptographie à clé secrète, doit se faire par des moyens traditionnels comme la poste ou le téléphone.

Par contre, en ce qui concerne la cryptographie à clé publique, les clés privées doivent être obtenues de manière tout à fait sûre, et les clés publiques doivent être disponibles. Le gestionnaire de ces clés publiques doit prouver qu'il est bien le gestionnaire des clés.

Ce problème a donné naissance aux certificats numériques. Un certificat est un document numérique attestant de la propriété d'une clé par une personne. Il doit être infalsifiable. Il doit pouvoir être obtenu en toute sécurité, et de telle façon que seul son destinataire légitime puisse l'utiliser.

Un certificat comprend en général :

- La clé publique,
- Le nom du propriétaire,

• La date d'expiration de la clé,

• Le nom du responsable du certificat,

• Le numéro de série du certificat.

7. Conclusion

Dans ce chapitre nous avons présenté une introduction générale sur la cryptographie, en a distingué deux classe importante des méthodes de chiffrement, c'est le cryptage symétrique a clé secrète et le cryptage asymétrique a clé publique, nous somme intéressé dans notre mémoire par la première méthodes de chiffrement à clé secrète qui sera appliqué sur des images compressées.

CHPITRE IV

Les diagrammes de la physiologie

1. Introduction

Les algorithmes de chiffrement définissent les transformations de données qui ne peuvent pas être facilement inversées par les utilisateurs non autorisés. Dans ce chapitre, nous présentons les différents algorithmes et techniques de chiffrement.

2. Algorithme de chiffrement à clef secrète

La cryptographie symétrique est très utilisée et se caractérise par une grande rapidité (opérations simples, chiffrement à la volée) et par des implémentations aussi bien software que hardware ce qui accélère nettement les débits et autorise son utilisation massive.

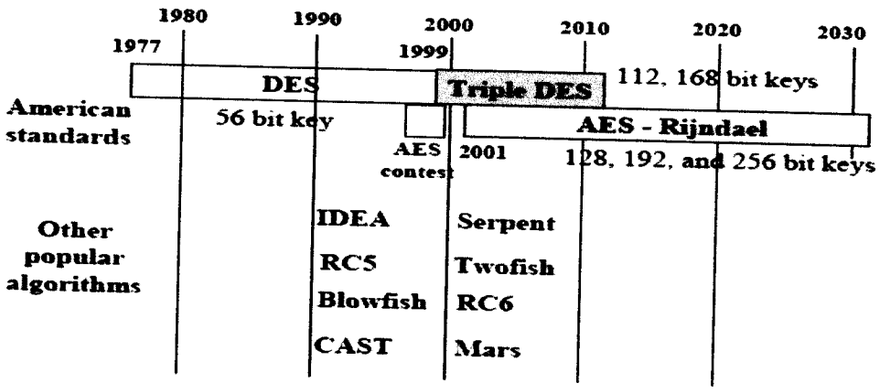


Figure IV. 1: Historique de chiffrements symétriques

L'idée générale du chiffrement par blocs est la suivante :

1) Remplacer les caractères par un code binaire.
2) Découper une chaîne en blocs de longueur donnée.
3) Chiffrer un bloc en l'"additionnant" bit par bit à une clef.
4) Déplacer certains bits du bloc.
5) Recommencer éventuellement un certain nombre de fois l'opération 3.
6) Passer au bloc suivant et retourner au point 3 jusqu'à ce que tout le message soit chiffré.

On distingue trois catégories de chiffrement par bloc :

- **Chiffrement par substitution** : Les substitutions consistent à remplacer des symboles ou des groupes de symboles par d'autres symboles ou groupes de symboles dans le but de créer de la confusion.
- **Chiffrement par transposition**: Les transpositions consistent à mélanger les symboles ou les groupes de symboles d'un message clair suivant des règles prédéfinies pour créer de la diffusion. Ces règles sont déterminées par la clé de chiffrement. Une suite de transpositions forme une permutation.
- **Chiffrement par produit** : C'est la combinaison des deux. Le chiffrement par substitution ou par transposition ne fournit pas un haut niveau de sécurité, mais en combinant ces deux transformations, on peut obtenir un chiffrement plus robuste. La plupart des algorithmes à clés symétriques utilisent le chiffrement transformations ont été faites une fois (substitution et transposition)[16].

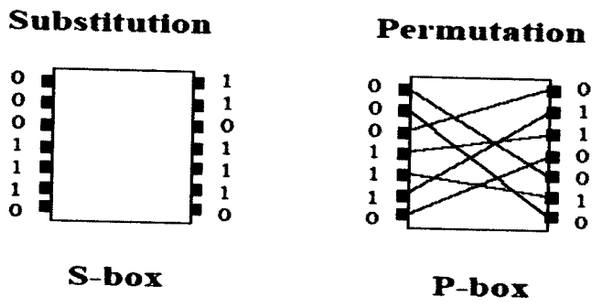


Figure IV. 2 : Substitution et permutation.

- **Effet d'avalanche**

C'est une propriété des chiffrements par blocs composés de couches (layers) ou "rounds" caractérisés par un petit changement à l'entrée. Le changement d'un simple bit d'entrée produit généralement de multiples changements de bits après un round, plusieurs autres changements de bits après un autre round jusqu'au changement éventuel de la moitié du bloc. [6]

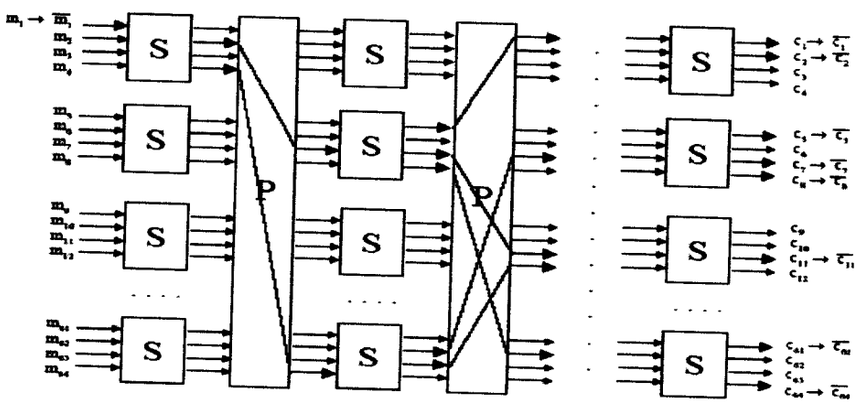


Figure IV. 3 : Effet d'avalanche.

2.1. Le OU exclusif

Le OU exclusif ou proprement dit le XOR est un algorithme de cryptographie le plus simple à implémenter sur ordinateur, c'est un algorithme symétrique puisque l'algorithme de cryptage est similaire a celui de décryptage, il fonctionne par une clé, qui doit être choisi selon le message à crypter, c'est l'étape la plus importante.

Il est relativement facile de crypter une image avec des techniques informatiques simples l'idée est d'additionner deux images, l'originale et une image clef.

Une image digitalisée peut être décrite comme une suite de pixels, chaque pixel ayant une couleur. Supposons pour simplifier que l'image est en niveau de gris. Le niveau de gris sera représenté par exemple par un nombre entier entre 0 est 255 (0 : noir ,255 : blanc).

Ces nombres sont codés dans l'ordinateur en binaire sur 8bits .on peut additionner deux bits à l'aide de la fonction XOR donnée par le tableau suivant :

- **Le tableau de XOR**

XOR	0	1
0	0	1
1	1	0

Tableau IV. 1: Tableau de XOR

Ainsi, si le premier pixel de l'image originale est 01110011 et le premier pixel de l'image clef 10100101, l'addition des deux sera :

1er pixel de l'image original	01110011
1er pixel de l'image clef	10100101
1er pixel de l'image brouillée	11010110

Figure IV. 4 : L'addition du pixel originale et la clef

Le grand intérêt de cette méthode réside dans le fait que si l'on additionne l'image brouillée et l'image clef, on retrouve l'image originale. Si l'on additionne le premier pixel de l'image brouillée au premier pixel de l'image clef on aura :

1er pixel de l'image	11010110
1er pixel de l'image clef	10100101
1er pixel de l'image original	01110011

Figure IV. 5 : L'addition du pixel brouillée et la clef

2.2. Le DES (Data Encryption System)

2.2.1. Introduction

L'algorithme DES, Data Encryption System, est un standard de chiffrement aux Etats-Unis et dans une grande partie du monde depuis la fin des années 1970.

Jusque dans les années 1970, seuls les militaires possédaient des algorithmes à clé secrète fiables. Devant l'émergence de besoins civils, le NBS (National Bureau of Standards) lança le 15 mai 1973 un appel d'offres dans le Federal Register (l'équivalent du Journal Officiel américain) pour la création d'un système cryptographique répondant aux critères suivants :

- posséder un haut niveau de sécurité lié à une clé de petite taille servant au chiffrement et au déchiffrement
- être compréhensible
- ne pas dépendre de la confidentialité de l'algorithme.
- être adaptable et économique.
- être efficace et exportable.

Fin 1974, IBM propose « Lucifer », qui, grâce à la NSA (National Security Agency), est modifié le 23 novembre 1976 pour donner le DES (Data Encryption Standard). Le DES a finalement été approuvé en 1978 par le NBS. Le DES fut normalisé par l'ANSI (American National Standard Institute) sous le nom de ANSI X3.92, plus connu sous la dénomination DEA (Data Encryption Algorithm)[16][17].

2.2.2. Principe de fonctionnement

Le DES transforme des blocs de 64 bits avec une clef secrète de 56 bits, La clé du DES est une chaîne de 64 bits (succession de 0 et de 1), mais en fait seuls 56 bits servent réellement à définir la clé. Les bits 8, 16, 24, 32, 40, 48, 56,64 sont des bits de parité (=bits de détection d'erreur). Le 8ème bit est fait en sorte que sur les 8 premiers bits, il y ait un nombre impair de 1. Par exemple, si les 7 premiers bits sont 1010001, le 8ème bit est 0. Ceci permet d'éviter les erreurs de transmission.

Il y a donc pour le DES 2^{56} clés possibles, soit environ ... 72 millions de milliards possibilités.

Son principe est le suivant :

- Fractionnement du texte en blocs de 64 bits (8 octets).
- Permutation initiale des blocs.
- Découpage des blocs en deux parties: gauche et droite, nommées G et D.
- Etapes de permutation et de substitution répétées 16 fois (appelées rondes).
- Recollement des parties gauche et droite puis permutation initiale inverse.

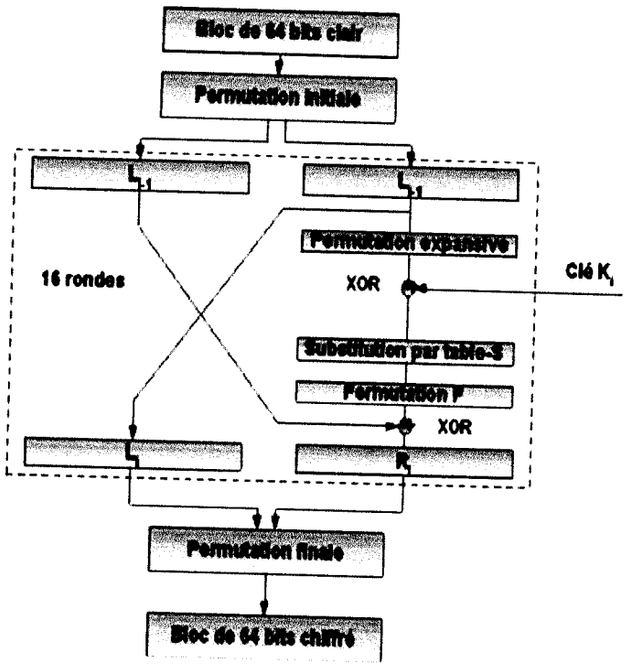


Figure IV. 6 : Principe de DES

1) Fractionnement de texte

Le texte en clair est scindé en un nombre de bloc comportant chacun 64bits.µ

2) Permutation initiale

Dans un premier temps, chaque bit d'un bloc est soumis à la permutation initiale, pouvant être représentée par la matrice de permutation initiale (notée PI) suivante :

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

Tableau IV. 2 : Permutation initiale

Cette matrice de permutation indique, en parcourant la matrice de gauche à droite puis de haut en bas, que le 58ème bit du bloc de texte de 64 bits se retrouve en première position, le 50ème en seconde position et ainsi de suite.

3) Scindement en blocs de 32 bits

Une fois la permutation initiale réalisée, le bloc de 64 bits est scindé en deux blocs de 32 bits, notés respectivement G et D (pour gauche et droite, la notation anglo-saxonne étant L et R pour Left and Right). On note G_0 et D_0 l'état initial de ces deux blocs :

G_0	58 50 42 34 26 18 10 2
	60 52 44 36 28 20 12 4
	62 54 46 38 30 22 14 6
	64 56 48 40 32 24 16 8
D_0	57 49 41 33 25 17 9 1
	59 51 43 35 27 19 11 3
	61 53 45 37 29 21 13 5
	63 55 47 39 31 23 15 7

Tableau IV. 3 : Scindement en blocs de 32 bits

Il est intéressant de remarquer que G_0 contient tous les bits possédant une position paire dans le message initial, tandis que D_0 contient les bits de position impaire.

4) Ronde

Les blocs G_0 et D_0 sont soumis à un ensemble de transformation itérative appelée rondes, explicitées dans ce schéma, et dont les détails sont donnés plus bas :

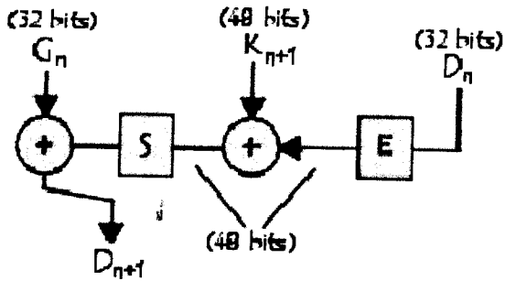


Figure IV. 7 : Ronde

5) Fonction d'expansion

Les 32 bits du bloc D_0 sont étendus à 48 bits grâce à une table (matrice) appelé table d'expansion (notée E), dans laquelle les 48 bits sont mélangés et 16 d'entre eux sont dupliqués :

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

Tableau IV. 4 : Table d'expansion

Ainsi, le dernier bit de D_0 (c'est-à-dire le $7^{ème}$ bit du bloc d'origine) devient le premier, le premier devient le second, ...

De plus, les bits 1,4,5,8,9,12,13,16,17,20,21,24,25,28 et 29 de D_0 (respectivement 57, 33, 25, l, 59, 35, 27, 3, 6l, 37, 29, 5, 63, 39, 31 et 7 du bloc d'origine) sont dupliqués et disséminés dans la matrice.

6) OU exclusif avec la clé

La matrice résultante de 48 bits est appelée D_0 ou bien E [D_0]. L'algorithme DES possède ensuite à un OU exclusif entre la première clé K1 est E [D_0]. Le résultat de ce OU exclusif est une matrice de 48 bits que nous appellerons D_0 par commodité (il ne s'agit pas du D_0 de départ).

7) Fonction de substitution

D_0 est ensuite scindé en 8 blocs de 6 bits, noté D_{0i}. Chacun de ces blocs passe par des fonctions de sélection (appelées parfois boîtes de substitution ou fonctions de compression), notées généralement S_i.

Les premiers et derniers bits de chaque D_{0i} détermine (en binaire) la ligne de la fonction de sélection, les autres bits (respectivement 2, 3, 4 et 5) déterminent la colonne. La sélection de la ligne se faisant sur deux bits, il y a 4 possibilités (0, 1, 2, 3). La sélection de la colonne se faisant sur 4 bits, il y a 16 possibilités (0 à 15). Grâce à cette information, la fonction de sélection "sélectionne" une valeur codée sur 4 bits.

Voici la première fonction de substitution, représentée par une matrice de 4 par 16 :

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

Tableau IV. 5 : première fonction de substitution

Soit D_{01} égal à 101110. Les premiers et derniers bits donnent 10, c'est-à-dire 2 en binaire. Les bits 2, 3, 4 et 5 donnent 0111, soit 7 en binaire. Le résultat de la fonction de sélection est donc la valeur situé à la ligne n°2, dans la colonne n°7. Il s'agit de la valeur 11, soit en binaire 111.

Chacun des 8 blocs de 6 bits est passé dans la fonction de sélection correspondante, ce qui donne en sortie 8 valeurs de 4 bits chacune. Voici les autres fonctions de sélection :

S2

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

S3

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

S4

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

S5

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

S6

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

S7

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

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

Tableau IV. 6 : fonctions de sélection

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

8) Permutation

Le bloc de 32 bits obtenu est enfin soumis à une permutation P dont voici la table :

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

Tableau IV. 7 : Tableau de permutation

9) OU Exclusif

L'ensemble de ces résultats en sortie de P est soumis à un OU Exclusif avec le G_0 de départ (comme indiqué sur le premier schéma) pour donner D1, tandis que le D_0 initial donne G_1.

L'ensemble des étapes précédentes (rondes) est réitéré 16 fois.

10) Permutation initiale inverse

A la fin des itérations, les deux blocs G_{16} et D_{16} sont "recollés, puis soumis à la permutation initiale inverse :

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

Tableau IV. 8 : Permutation initiale inverse

Le résultat en sortie est un texte codé de 64 bits.

11) Génération des clés

Etant donné que l'algorithme du DES présenté ci-dessus est public, toute la sécurité repose sur la complexité des clés de chiffrement.

L'algorithme ci-dessous montre comment obtenir à partir d'une clé de 64 bits (composé de 64 caractères alphanumériques quelconques) 8 clés diversifiées de 48 bits chacune servant dans l'algorithme du DES :

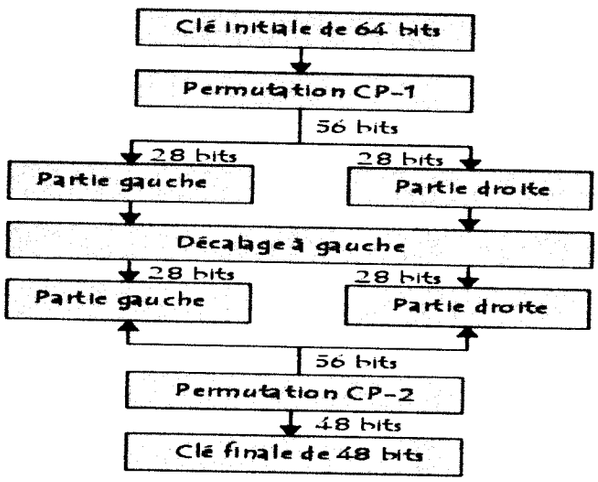


Figure IV. 8 : Algorithme de génération de clés

Dans un premier temps les bits de parité de la clé sont éliminés afin d'obtenir une clé d'une longueur utile de 56 bits.

La première étape consiste en une permutation notée CP-1 dont la matrice est présentée ci-dessous :

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

Tableau IV. 9 : Première matrice de permutation

Cette matrice peut en fait s'écrire sous la forme de deux matrice G_i et D_i (pour gauche et droite) composées chacune de 28 bits :

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

Tableau IV. 10 : Matrices composantes

On note G_0 et D_0 le résultat de cette première permutation.

Ces deux blocs subissent ensuite une rotation à gauche, de telles façons que les bits en seconde position prennent la première position, ceux en troisième position la seconde, ... Les bits en première position passent en dernière position.

Les 2 blocs de 28 bits sont ensuite regroupés en un bloc de 56 bits. Celui-ci passe par une permutation, notée CP-2, fournissant en sortie un bloc de 48 bits, représentant la clé K_i.

CP-2	14	17	11	24	1	5	3	28	15	6	21	10
	23	19	12	4	26	8	16	7	27	20	13	2
	41	52	31	37	47	55	30	40	51	45	33	48
	44	49	39	56	34	53	46	42	50	36	29	32

Tableau IV. 11 : Deuxième matrice de permutation

Des itérations de l'algorithme permettent de donner les 16 clés K_1 à K_{16} utilisées dans l'algorithme du DES. [7]

LS	1	2	4	6	8	10	12	14	15	17	19	21	23	25	27	28

Tableau IV. 12 : Les 16 clé de l'algorithme

2.2.3. Sécurité du DES

Depuis son invention, la sécurité du DES a fait l'objet d'études attentives. Des techniques spéciales telles que la cryptanalyse différentielles , ou linéaire , on été inventées pour attaquer le DES , mais les attaques les plus efficaces proviennent d'une exploration exhaustive de l'espace des clés .Avec des matériels spécifique ou des grands réseaux de stations de travail , il est maintenant possible de déchiffrer les cryptogrammes venant du DES en quelques jours , voire même quelque heures .Au train ou la puissance des ordinateurs augmente , on s'attend à ce que le DES puisse bientôt être cassé par un simple PC. [8][17]

2.3. L'IDEA (International Data Encryption Algorithm)

2.3.1. Introduction :

IDEA est un algorithme de chiffrement symétrique conçu par Xuejia Lai et James Massey.

IDEA est un algorithme de chiffrement par blocs utilisé pour chiffrer et déchiffrer des données. Il manipule des blocs de texte en clair de 64 bits. Une clé de chiffrement longue de 128 bits , est utilisée pour le chiffrement des données.

Les opérations mathématiques utilisées dans cet algorithme sont :

- XOR.
- L'addition modulo 2^{16} (65 536).
- Multiplication modulo $2^{16}+1$ (65 537, qui est un nombre premier).

2.3.2. Principe de fonctionnement

- **Détermination des sous-clés**

Les 52 sous-clés générées à partir de la clé de 128 bits sont produites comme suit :

1) La clé de 128 bits est divisée en huit blocs. Ces huit blocs sont en fait les huit premières sous-clés utilisées dans le chiffrement.
2) La clé de 128 bits est ensuite cycliquement décalée de 25 positions et divisée en huit blocs de 16 bits. Ces huit blocs sont les huit sous-clés suivantes utilisées dans le chiffrement.
3) Le cycle est répété jusqu'à ce que les 52 sous-clés soient toutes générées.

Utilisation des sous-clés de chiffrement :

Round 1	K[1] , K[2] , K[3] , K[4] , K[5] , K[6]
Round 2	K[7] , K[8] , K[9] , K[10] , K[11] , K[12]
Round 3	K[13] , K[14] , K[15] , K[16] , K[17] , K[18]
Round 4	K[19] , K[20] , K[21] , K[22] , K[23] , K[24]
Round 5	K[25] , K[26] , K[27] , K[28] , K[29] , K[30]
Round 6	K[31] , K[32] , K[33] , K[34] , K[35] , K[36]
Round 7	K[37] , K[38] , K[39] , K[40] , K[41] , K[42]
Round 8	K[43] , K[44] , K[45] , K[46] , K[47] , K[48]
Transformation finale	K [49], K [50], K [51], K [52]

Dans le premier round, les quatre premières sous-clés sont combinées avec tout d'abord deux blocs de 16 bits du texte clair selon l'addition modulo 2^{16}, et avec deux autres blocs de texte clair en utilisant la multiplication modulo $2^{16} + 1$. Les résultats sont ensuite traités plus loin, où deux autres sous-clés sont employées et l'opération du troisième groupe algébrique, le OU exclusif bit-à-bit, est utilisée. À la fin du premier round de chiffrement, quatre valeurs de 16 bits sont produites et elles sont utilisées comme valeurs d'entrée au deuxième round de chiffrement, dans un ordre inter changé.

Le même processus s'effectue dans les sept rounds de chiffrement suivants, avec des sous-clés différentes pour chaque combinaison. Les quatre blocs de 16 bits produits par le dernier round, le huitième, sont combinés avec les quatre dernières sous-clés selon l'addition modulo 2^{16} et la multiplication modulo 2^{16}. Le résultat final forme quatre blocs chiffrés de 16 bits, donc 64 bits de texte chiffré.

Il est à noter que dans la multiplication de deux blocs de 16 bits modulo $2^{16} + 1$, un bloc de 16 bits ayant tout ses bits à 0 n'est pas interprété comme ayant une valeur totale de 0 mais plutôt une valeur de $2^{16.}$

Soient A, B, C et D quatre blocs de 16 bits et 52 sous-clé K[1] à K[52]. ((·) Est une multiplication et (+) est une addition).

- **Avant d'effectuer le premier round**

A = A · K[1]

B = B + K[2]

C = C + K[3]

D = D · K[4]

- **Round 1 :**

E = A XOR C

F = B XOR D

E = E · K [5]

F = F + E

F = F · K [6]

E = E + F

A = F XOR A

C = F XOR C

B = E XOR B

D = E XOR D

Tampon=B

B=C

C = Tampon

Répéter sept autres fois en utilisant les sous-clés suivantes : K[7] à K[12] pour le deuxième round, ..., K[43] à K[48] pour le huitième round.

- **Après le huitième round**

A = A · K [49]

B = B + K [50]

C = C + K [51]

D = D. K [52]

➢ **Déchiffrement**

Le déchiffrement s'effectue essentiellement de la même manière que le chiffrement. La seule différence est que les 52 sous-clés sont générées de façon inverse au chiffrement. Aussi les blocs de texte chiffré doivent être traités dans l'ordre inverse du chiffrement pour inverser parfaitement le processus de chiffrement.

Utilisation des sous-clés de déchiffrement, selon les sous-clés de chiffrement (Ex.: $K_{déchiffrement}[1] = 1 / K_{chiffrement}[49]$) :

Transformation initiale	1 / K[49], - K[50], - K[51], 1 / K[52]
Round 1	K[47], K[48], 1 / K[43], - K[45], - K[44], 1 / K[46]
Round 2	K[41], K[42], 1 / K[37], - K[39], - K[38], 1 / K[40]
Round 3	K[35], K[36], 1 / K[31], - K[33], - K[32], 1 / K[34]
Round 4	K[29], K[30], 1 / K[25], - K[27], - K[26], 1 / K[28]
Round 5	K[23], K[24], 1 / K[19] , - K[21] , - K[20] , 1 / K[22]
Round 6	K[17], K[18], 1 / K[13] , - K[15] , - K[14] , 1 / K[16]
Round 7	K[11], K[12], 1 / K[7], - K[9], - K[8], 1 / K[10]
Round 8	K[5], K[6], 1 / K[1], - K[3], - K[2], 1 / K[4]

La méthode de génération des sous-clés de l'IDEA est toujours régulière et donc pourrait être une faiblesse à l'algorithme. Cependant, il est considéré comme étant hautement sécuritaire. En effet, pour résoudre IDEA avec l'attaque en force, il faudrait effectuer 2^{128}, donc 10^{38} opérations.

IDEA est protégé par les lois internationales de "copyright" et a été breveté dans plusieurs pays. Néanmoins, son utilisation non commerciale est gratuite.

Le bloc de texte en clair IDEA est divisé en quatre blocs, chacun 16 bits de long, la clé de Chiffrement est découpée en 8 sous blocs de 16 bits.

Tous les blocs de la clef ne servent pas pendant une ronde, seuls les 6 sous blocs de clef sont nécessaires pour une ronde et 4 sont nécessaires pour la transformation finale.

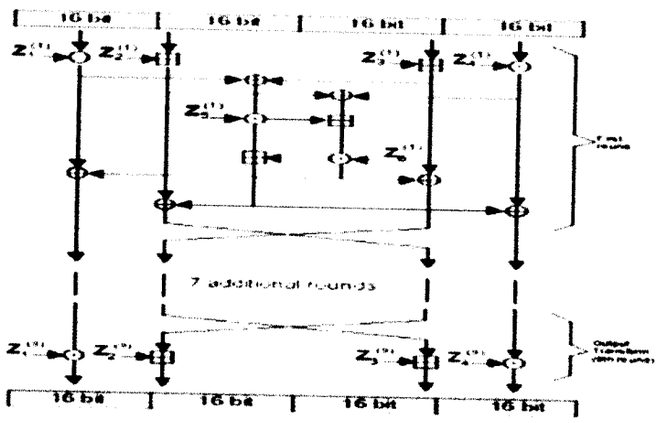
- **Calculs des sous clefs**

L'algorithme utilise 6 clefs par ronde, donc 48 totaux, plus les 4 nécessaires à la transformation finale, soit 52 clefs.

La clef initiale de 128 bits est découpée en 8 sous clefs de 16 bits. Les 6 premières sous – clefs sont utilisées pour la première ronde et les deux restante pour la deuxième ronde

Ensuite, on décale la clef de 25 bits sur la gauche et on la redécoupe en 8 sous clefs dont les 4 premières sont utilisées pour la deuxième ronde et les 4 restante pour la troisième ronde et ainsi de suite jusqu'à la fin des rondes et de la transformation finale. [4]

Texte clair (64 bits)



Texte chiffré (64 bits)

⊕ OU-Exclusif sur 2 blocs de 16 bits

⊞ Addition modulo 2^{16} sur 2 blocs de 16 bits

⊙ Multiplication modulo $2^{16} + 1$ sur 2 blocs de 16 bits

Figure IV. 9. Principe de IDEA.

2.4. Le RC5

Le RC5 à été conçu en 1995 par Ronald Rivest pour la RSA Security. L'acronyme "RC" signifie "Ron's Code" ou "Rivest's Cipher".

2.4.1. Introduction

Le RC5 à l'avantage d'avoir une longueur de bloc de données variable, un nombre de rounds variable et une clé de longueur variable. Ainsi, l'utilisateur a le contrôle sur le rapport entre la vitesse d'exécution et la sécurité de son chiffrement. En général, une longue clé et un nombre élevé de rounds assurent une plus grande sécurité. La taille des blocs de données pour sa part accommode différentes architectures de systèmes.

La simplicité de l'algorithme du RC5 rend son implémentation facile et, le plus important, rend son analyse plus aisée. De plus, la forte utilisation des décalages de bits (appelés rotations) dans le chiffrement prévient l'usage de la cryptanalyse linéaire et différentielle.

En plus du mode EBC (Electronic Code Book), le RC5 est aussi utilisé avec le mode CBC (Cipher-Block Chaining).

2.4.2. Principe de fonctionnement

Comme pour le RC2, il y a deux parties dans l'algorithme, soit une procédure d'expansion de la clé et une procédure de chiffrement. Les opérations utilisées sont l'addition modulo $2^{(\text{nombre de bits des blocs})}$ (+), le OU-Exclusif (XOR) et le décalage de bits vers la gauche (<<<).

Soient K [0], K [1], ..., K [n] les sous-clés dérivées de la procédure d'expansion de la clé et A et B les deux parties d'un bloc de texte clair à chiffrer.

A=A+K[0]
B=B+K[1]
Pour i allant de 1 jusqu'au nombre du rounds(r)
A=((A XOR B)<<<B)+K[2i])

B=((B XOR A)<<<A)+K[2i+1])

Fin pour

Le déchiffrement est tout aussi facile. Partager le texte en deux mots A et B et effecteur :
Pour i décroissant r à 1 :

B= ((B – K [2i+1])>>>A) XOR A)
A= ((A-K [2i] >>B) XOR B)
B=B-K[1] A=A-K[0]

La création du tableau de clefs est plus compliqué mais aussi, nous devons commencer recopier les bits de la clef dans un tableau L de c mots de 32 bits en remplissant si nécessaire les derniers bits par des zéros. Initialisant d'abord le tableau en utilisant un générateur congruente modulo 2^{32}:

K [0] = P
Pour i variant de 1 à 2(r+1) -1, effectuant :
K[i] = (K[i] +Q) modulo 2^{32}
P = 0* B7E151663 et P = 0*9E3779B9 sont des constante basées sur e :
Pour finir, mélangent Let K :
i = j = 0
A = B =0
Effectuant n fois (ou n est le maximum de 2 (r+1) et c)
A = K[i] = (K[i] + A+B) <<< 3
B = L[i] = (L[i] + A + B) <<< (A + B)
i = (i+1) modulo 2 (r+1)
j = (j+1) modulo c

- **Génération des sous clefs**

La longueur de clef est variable dans RC5 (varie de 0 à 2048 bits). La création du tableau des sous clefs est plus compliquée. Commencez par copier les bits de la clef dans le tableau L. Initialiser alors le tableau en utilisant un générateur congruente modulo 2^{32}. [s21] [9] [10] [11]

3. Conclusion

L'algorithme de chiffrement joue un rôle important aussi bien au niveau de la sécurité que des performances. Dans le cas de configurations réseaux importantes, le choix d'un algorithme devient judicieux.

Pour conclure, les algorithmes DES, RC5 et IDEA conviennent parfaitement, ils sont sûrs et offrent de bonnes performances dans l'opération de chiffrement des images.

CHPITRE V

Implémentation des résultats

1. Introduction

Nous présentons dans ce chapitre, une étude comparative entre les deux algorithmes de compression huffman (que nous avons implémenté) et la méthode RLE (qui était implémenté l'année passée dans un projet PFE [12]), puis nous analysons les résultats obtenue de l'opération de chiffrement appliquer sur des images compressées par l'algorithme de huffman.

2. La comparaison entre huffman et RLE

2.1. Paramètre de comparaison

Pour faire la comparaison, on va se baser sur quelques points qui sont :

- ✓ Le taux de compression
- ✓ La vitesse et le temps de l'opération de compression
- ✓ La qualité de l'opération de chiffrement des images compressées

2.2. Le langage utilise

Le langage choisi pour réalisation de notre application est le BORLAND C++ BUILDER6. Ce choix repose sur le fait que Borland possède tout la puissance du langage C++ oriente objet comme il offre la possibilité de développer rapidement des applications sous Windows grâce a ses différentes bibliothèques. Il permet la création instantanée des interfaces utilisateurs car il offre une gestion de l'interface.

2.3. Implémentation

Description de l'application (interface et composantes) : la fenêtre d'interface de notre logiciel est présentée ci-dessus par la figure :

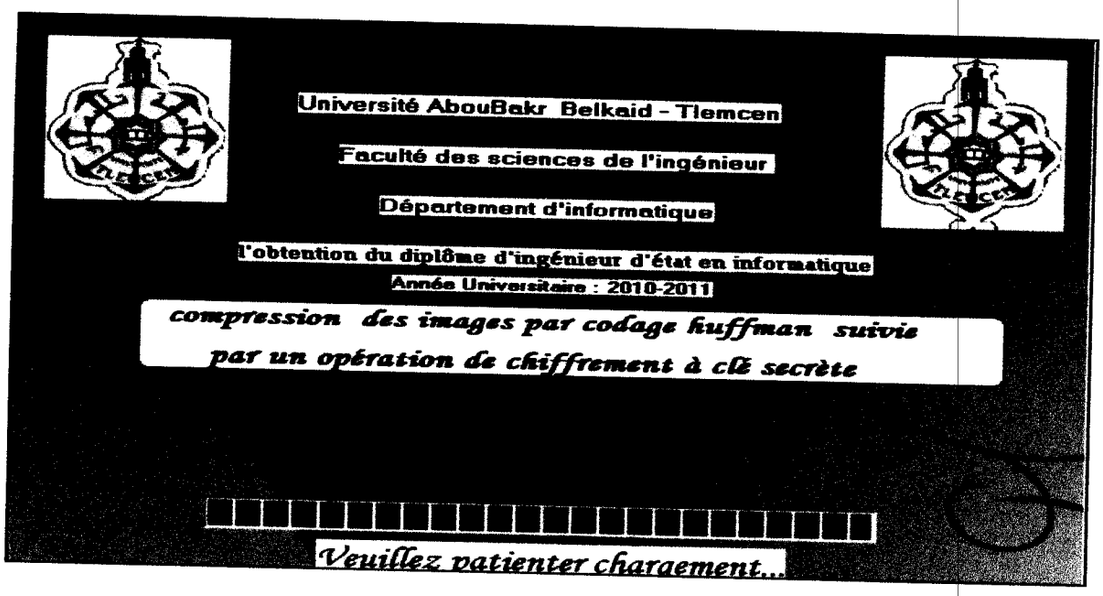


Figure V. 1: interface A propos.

L'interface générale de l'application est représentée par la figure ci-dessous

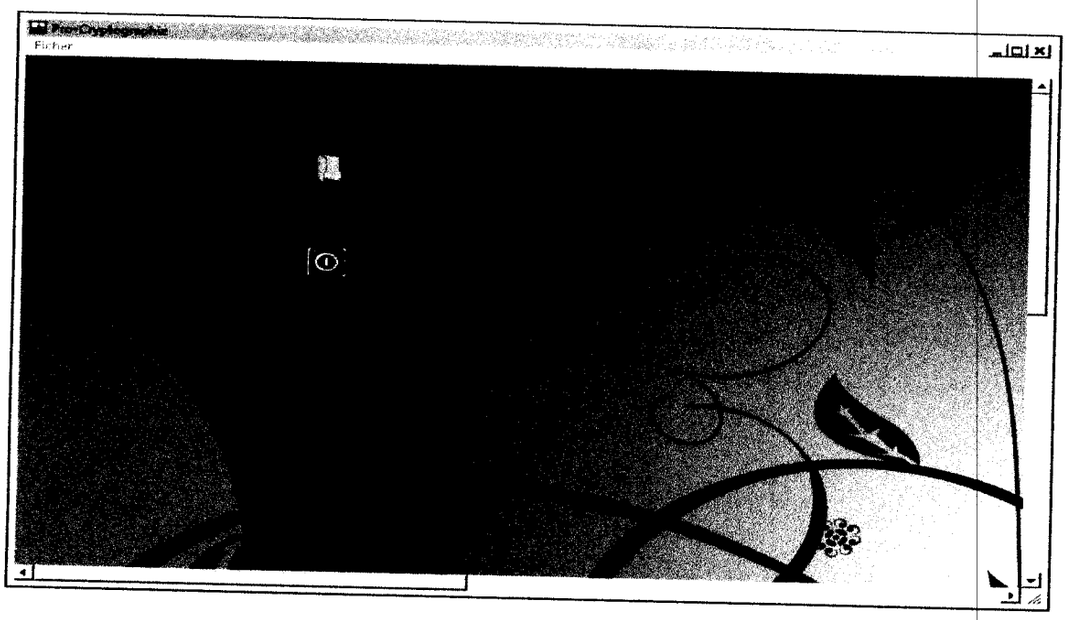


Figure V. 2: interface générale de l'application

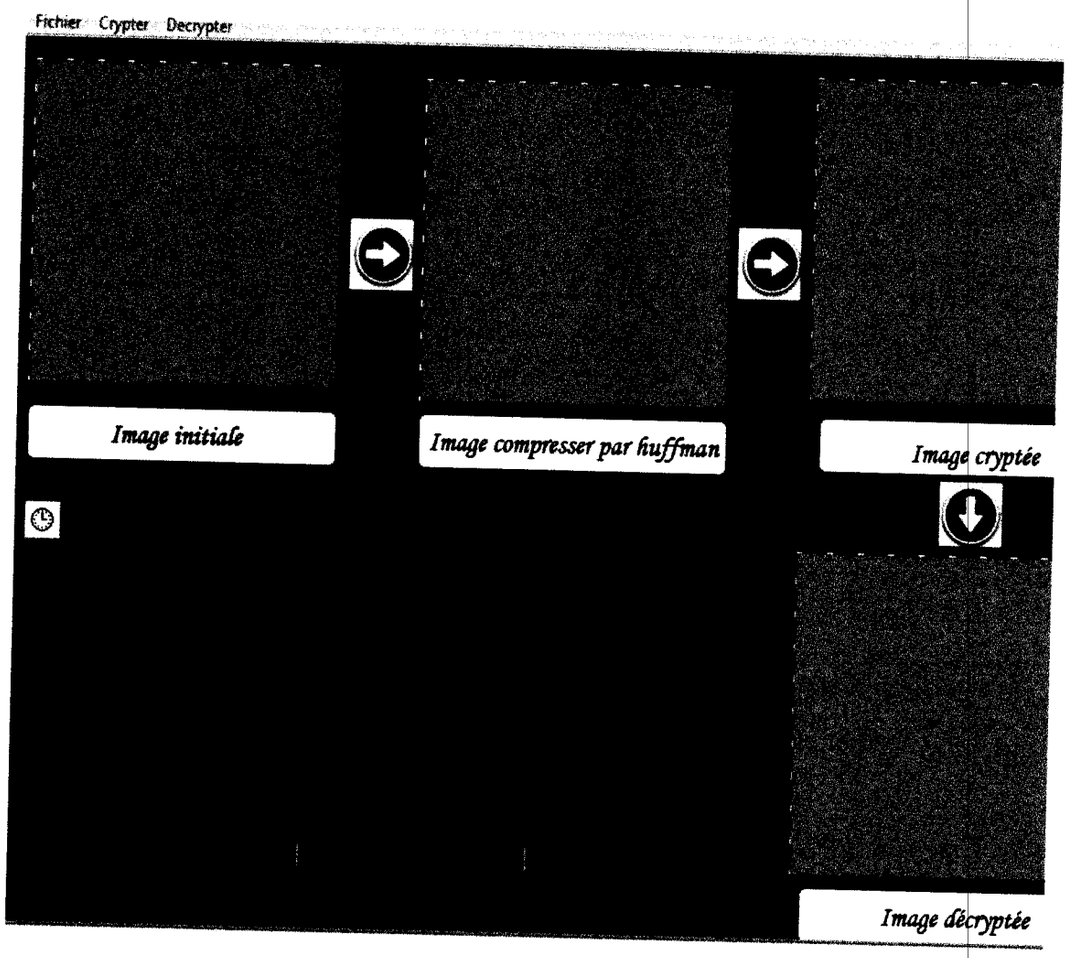


Figure V. 3: interface de compression par huffman-chiffrement par clé secrète

La fiche menu comporte les commandes suivantes :

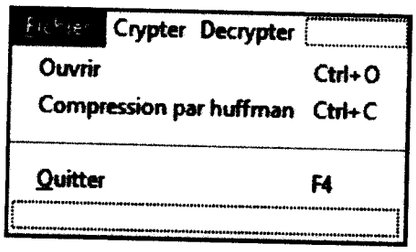
Fichier :

Ouvrir : permet d'ouvrir une image BMP

Compression-chiffrement : permet d'ouvrir l'interface de compression des images par huffman suivie par une opération de chiffrement par clé secrète.

Quitter : permet de quitter l'application.

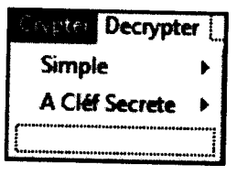
La figure ci-dessous nous permet d'afficher l'interface de compression et de chiffrement

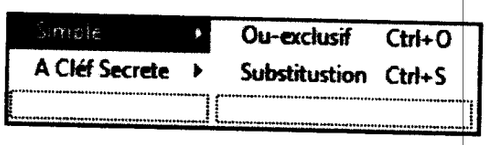


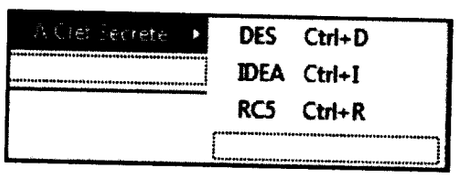
Crypter : contient le chiffrement simple et chiffrement à clé secrète

Simple : permet de choisir un algorithme de cryptage simple (ou-exclusif ou substitution)

A clé secrète : permet de choisir un algorithme de cryptage à clé secrète **(DES, IDEA ou RC5)**







2.4. Résultats obtenus et l'étude comparative

La figure ci-dessous nous permet d'afficher l'image initiale, l'image après l'opération de compression, ainsi que la taille de l'image avant et après compression mais aussi le taux de la compression et le temps de compression.

Figure V. 4: interface représentant l'image avant et après la compression

La figure ci-dessous nous permet d'ouvrir une image BMP et de compresser par huffman et de la crypter par l'algorithme de chiffrement choisi ainsi que d'afficher l'image décrypter et d'afficher le temps de cryptage et de décryptage sans oublier de passer par un mot de passe bien sur.

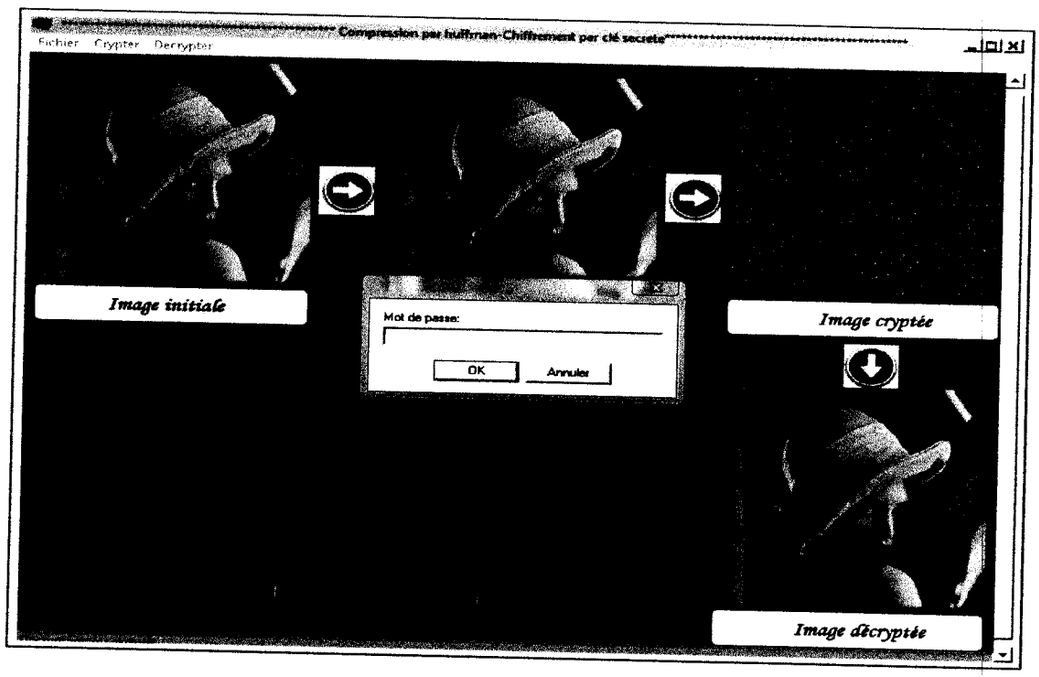


Figure V. 5: interface représentant l'image compressée avant et après cryptage

2.4.1. Résultat de l'opération de compression d'une image par l'algorithme huffman et RLE

On a choisi 4 images différentes :

Image 1

Image 2

Image 3

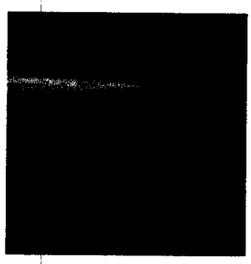
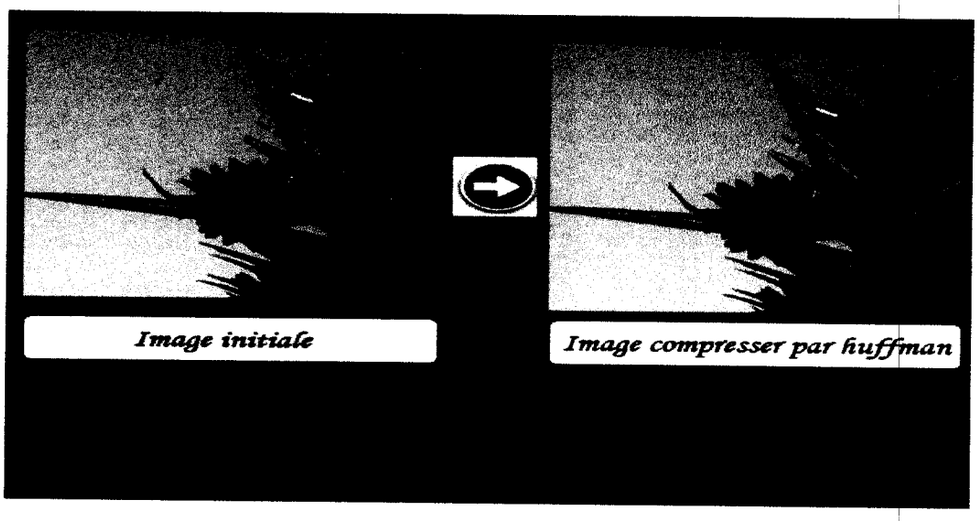


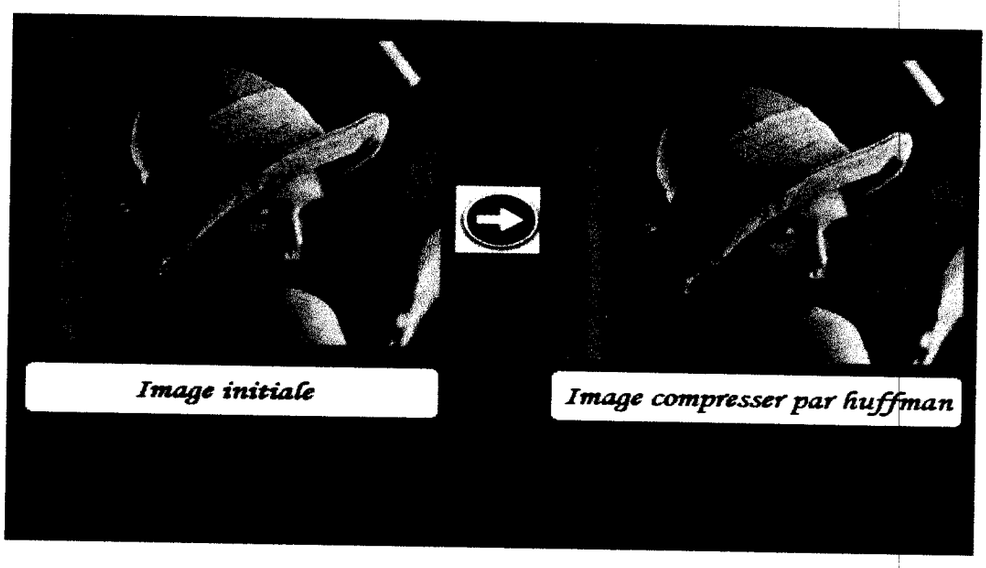
Image 4

2.4.1.1. Compression par huffman

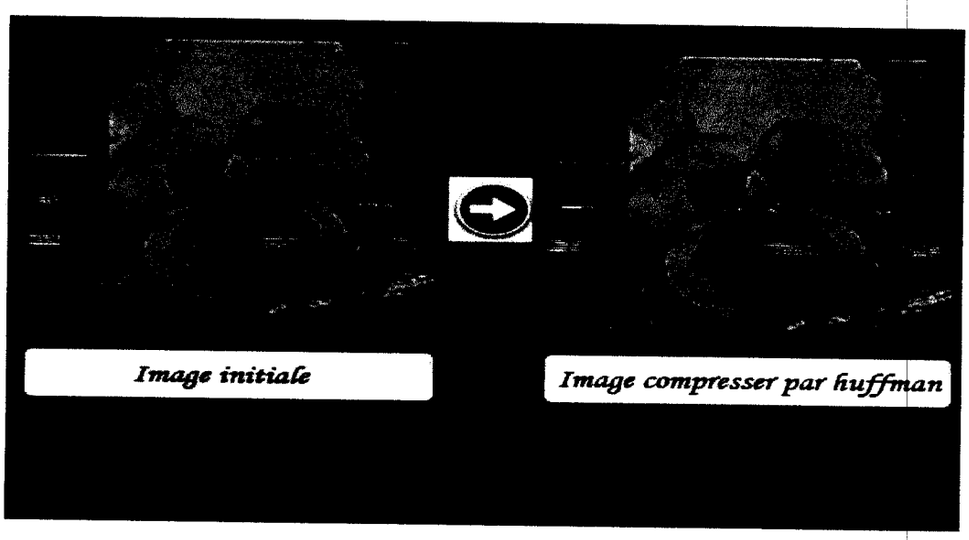
- ✓ **Compression de l'image 1**



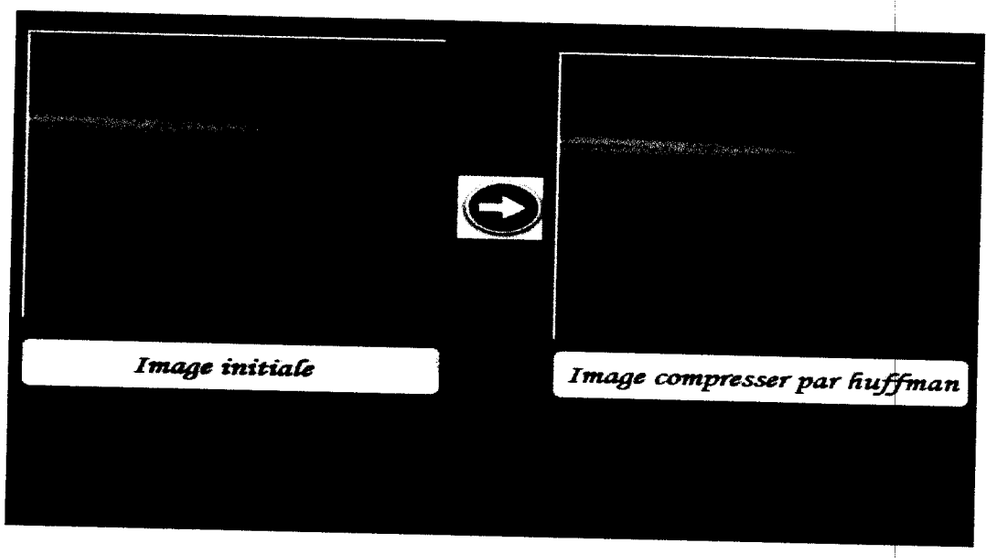
✓ **Compression de l'image 2**



✓ **Compression de l'image 3**

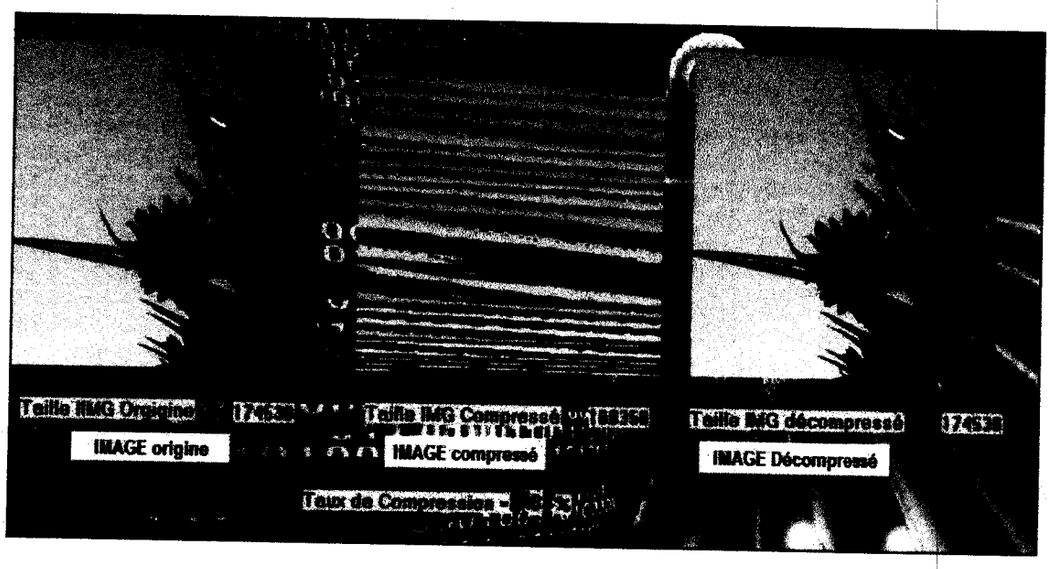


✓ **Compression de l'image 4**

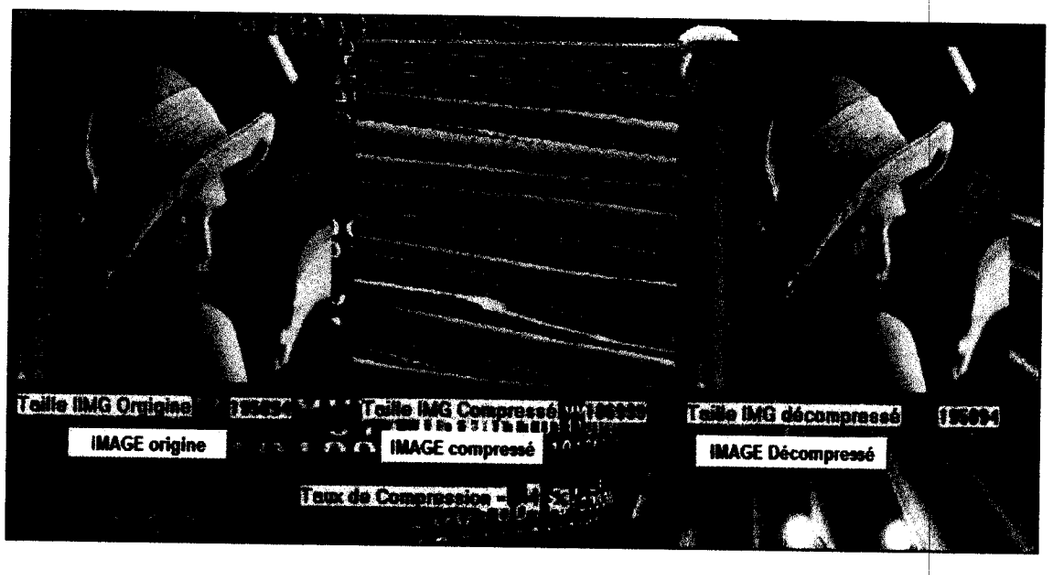


2.4.1.2. Compression par RLE

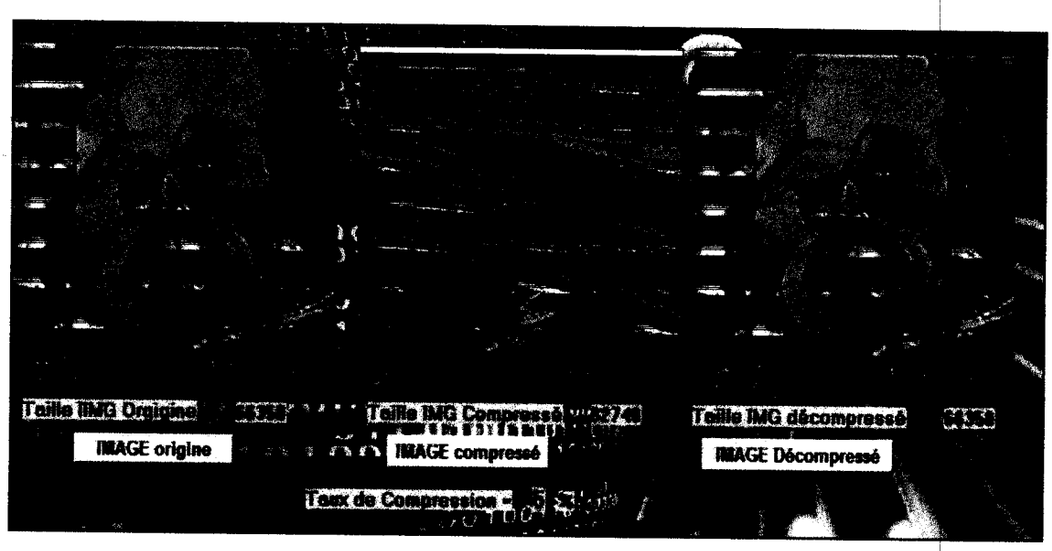
✓ **Compression de l'image 1**



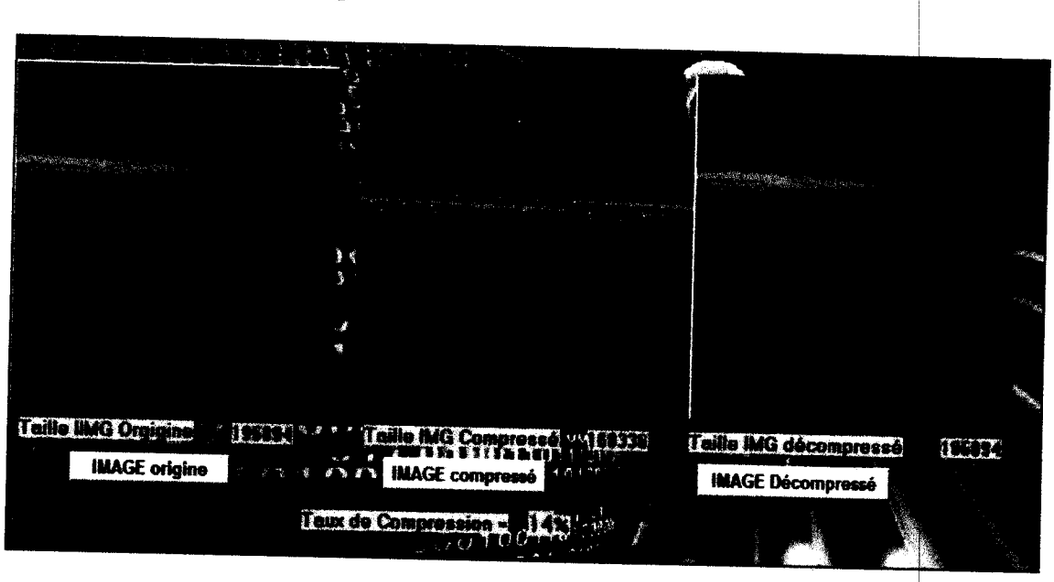
✓ **Compression de l'image 2**



✓ **Compression de l'image 3**



✓ **Compression de l'image 4**



Etude la comparative

Taille image	Algo Huffman			Algo RLE		
	Taille image compressée	**Taux de compression**	**Temps de compression**	**Taille image compressée**	**Taux de compression**	**Temps de compression**
Image 1 (195 ko)	7268 oct	95 %	0,54 ms	160350 oct	8 %	1, 23 ms
Image 2 (195 ko)	8436 oct	95 %	0,58 ms	18639 oct	4 %	1,13 ms
Image 3 (66 ko)	7753 oct	88 %	0, 33 ms	62748 oct	5 %	0,89 ms
Image 4 (195 ko)	5767 oct	97 %	0,57 ms	168336 oct	14 %	1,15 ms

Tableau V. 1: résultat de compression par huffman et RLE

Remarque : d'après les résultats obtenue nous observons dans le tableau v.1 que la compression par l'algorithme huffman est plus optimale (efficace) par rapport a l'algorithme RLE, plus la vitesse ou le temps de compression de huffman est plus accéléré que l'algorithme RLE.

2.4.2. La comparaison entre les différents algorithmes de chiffrement sur l'image compressé par huffman

- **Paramètre de comparaison**

Pour faire la comparaison en va baser sur quelques points qui sont :

- ✓ le temps de cryptage et le temps de décryptage des images compressées par huffman.
- ✓ La visibilité et la performance de l'opération de cryptage appliqué sur chaque image

- **Les algorithmes de chiffrement**

Pour les systèmes symétriques :

- Les algorithmes simples comme le ou exclusif ou la transposition utilisent n'importe quelle clé.

- Pour les algorithmes à clé secrète :

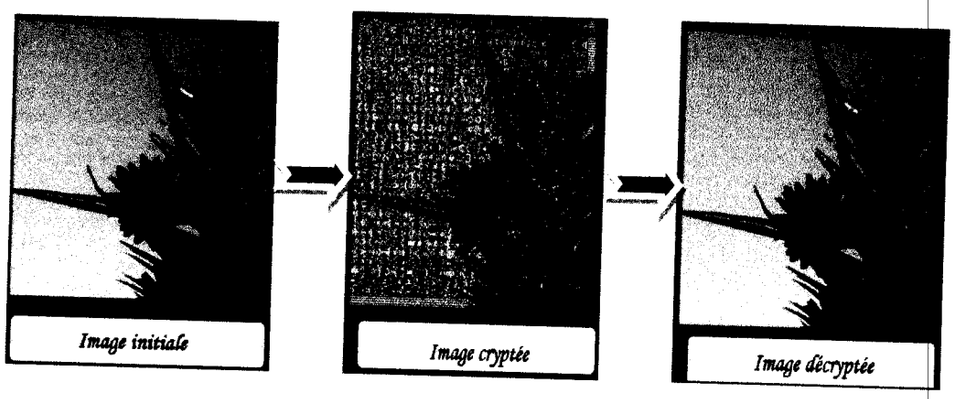
- DES (clef de 56 bits)
- IDEA (Clef de 128 bits).
- RC5 (Clef de 128 à 256 bits).

2.4.2.1. La qualité de l'opération de cryptage des images compressées par les différents algorithmes de chiffrement

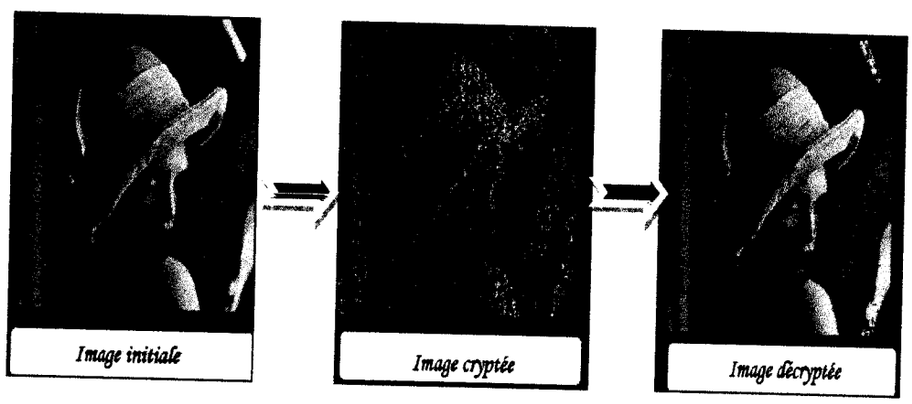
➢ **Les algorithmes simples**

✓ **Cryptage des images par ou exclusif**

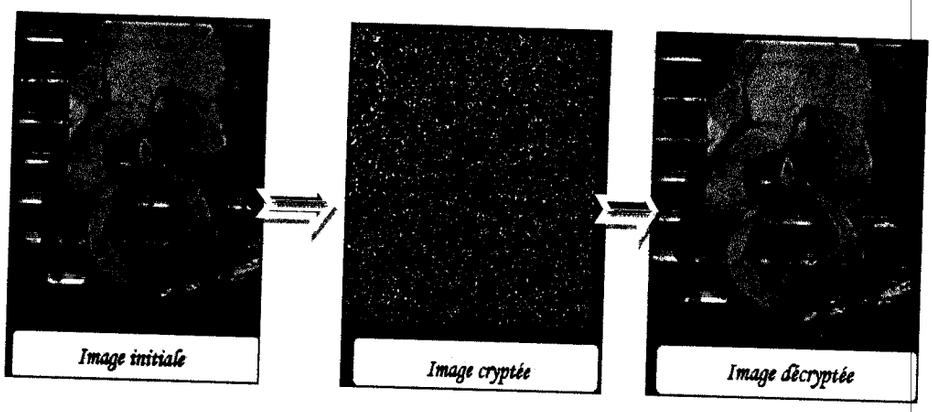
- **Image 1**



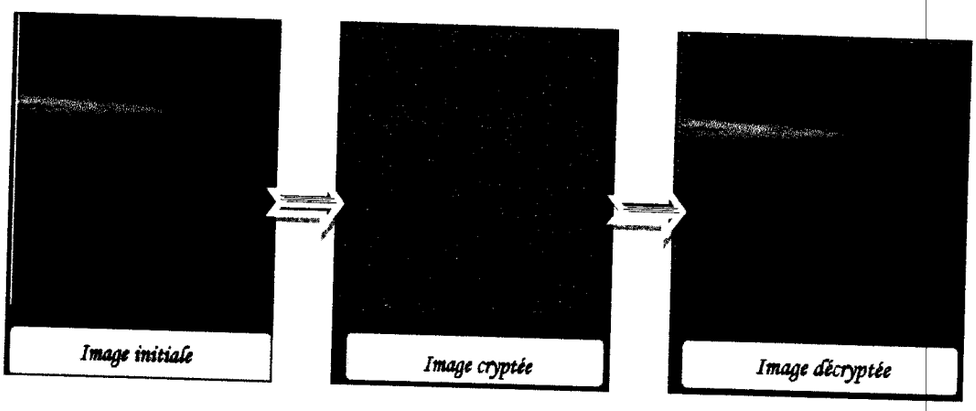
- **Image 2**



- **Image 3**



- **Image 4**

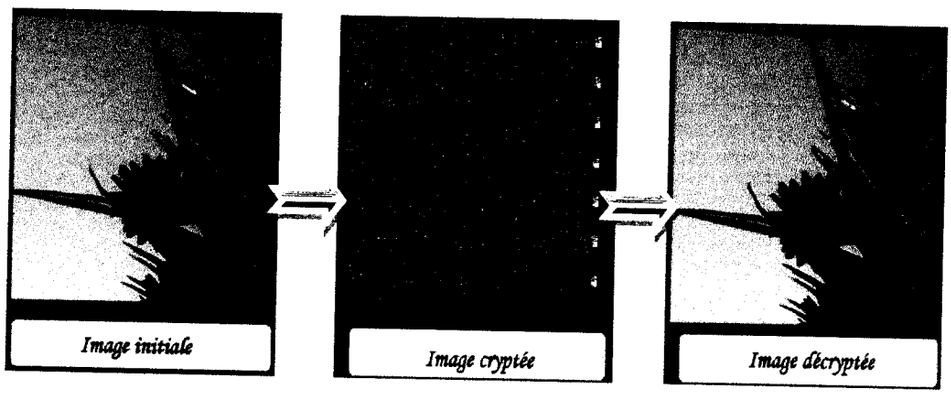


Images	**Image 1**	**Image 2**	**Image 3**	**Image 4**
Qualité	Moycn	Moycn	Moycn	Moycn
le temps de cryptage	3.354 (Sec)	4.431 (Sec)	1.607 (Sec)	1.482 (Sec)
le temps de décryptage	2.402 (Sec)	3.104 (Sec)	0.905 (Sec)	1.061 (Sec)

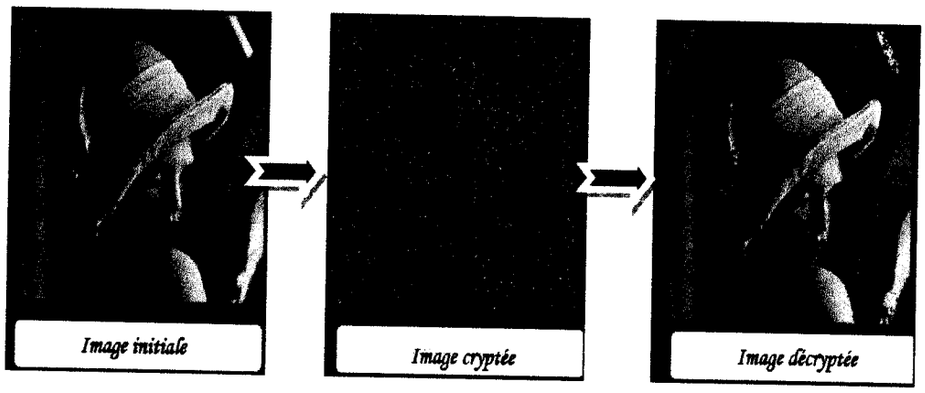
Tableau V. 2: résultat du cryptage par ou exclusif

✓ **Cryptage des images par substitution**

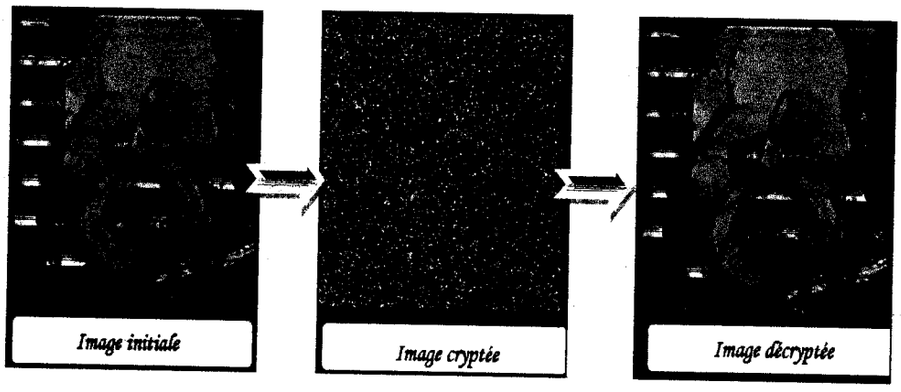
- **Image 1**



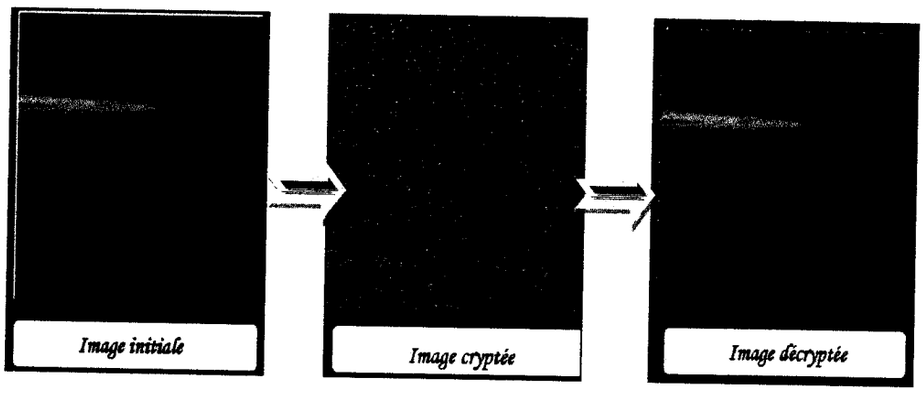
- **Image 2**



- **Image 3**



- **Image 4**



Images	**Image 1**	**Image 2**	**Image 3**	**Image 4**
Qualité	Bonne	Bonne	Bonne	Bonne
le temps de cryptage	1.545	1.375	1.482	2.465
le temps de décryptage	0.873	2.122	1.576	2.933

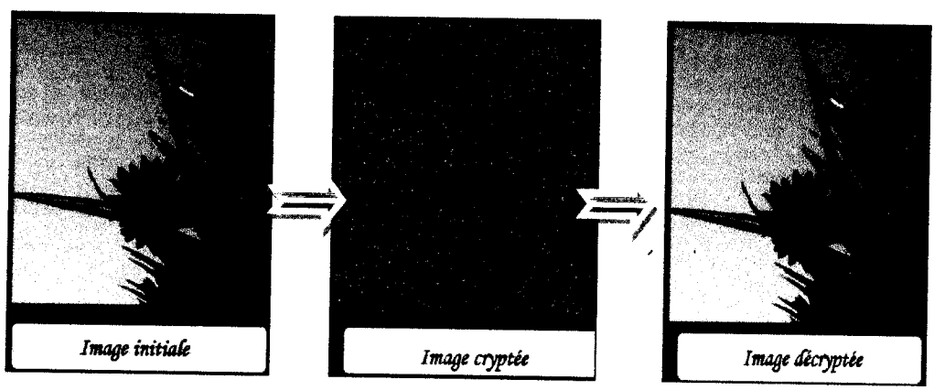
Tableau V. 3: résultat du cryptage par substitution

Remarque : nous observons que les algorithmes classique de chiffrement par ou exclusif et substitution donnent une mauvaise qualité de cryptage des images. Il y a toujours des traces sur l'image originale.

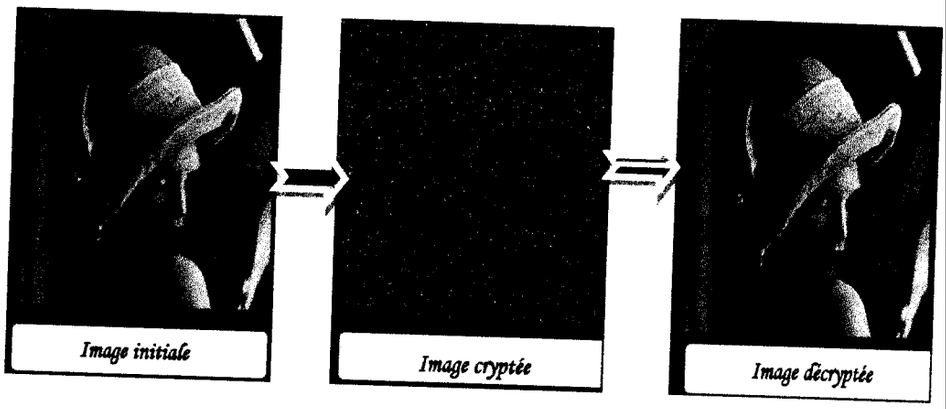
- **Les algorithmes à clé secrète**

✓ **Cryptage des images par DES**

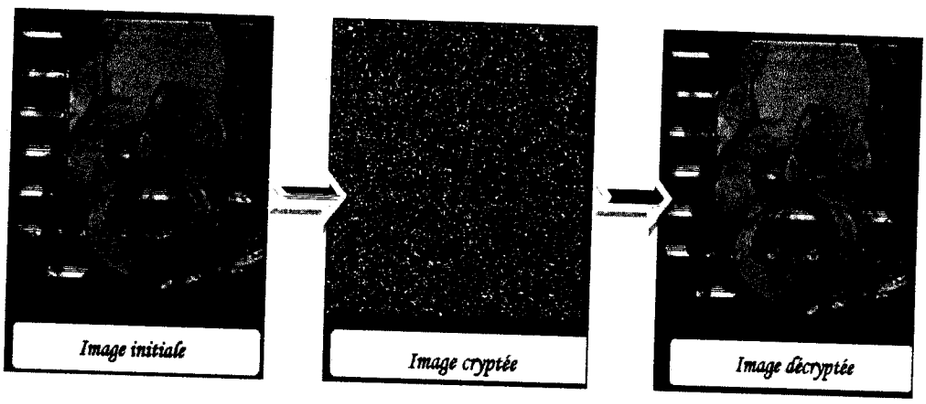
- **Image 1**



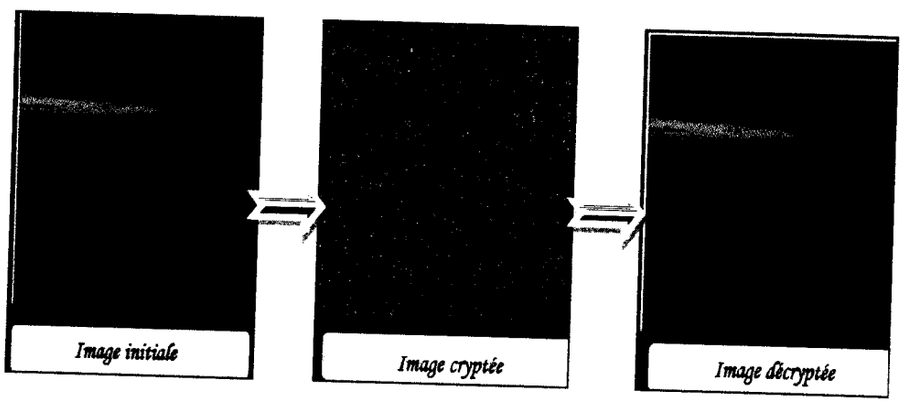
- **Image 2**



- **Image 3**



- **Image 4**

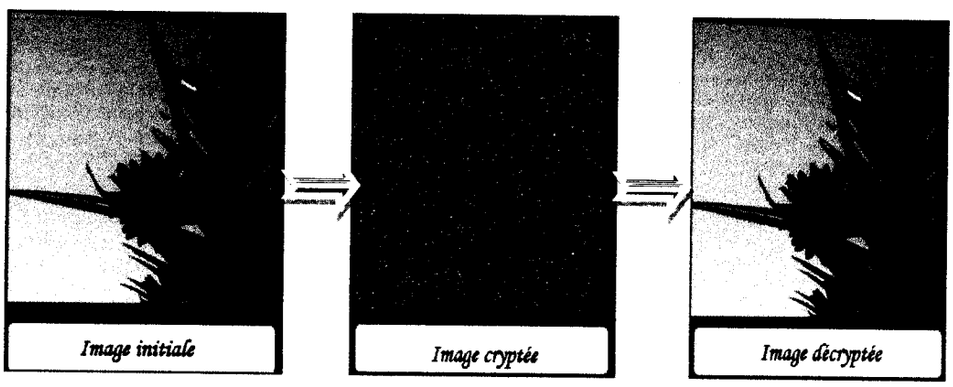


Images	Image 1	Image 2	Image 3	Image 4
Qualité	Bonne	Bonne	Bonne	Bonne
le temps de cryptage	1.607	3.837	2.168	1.872
le temps de décryptage	3.308	2.434	3.541	3.182

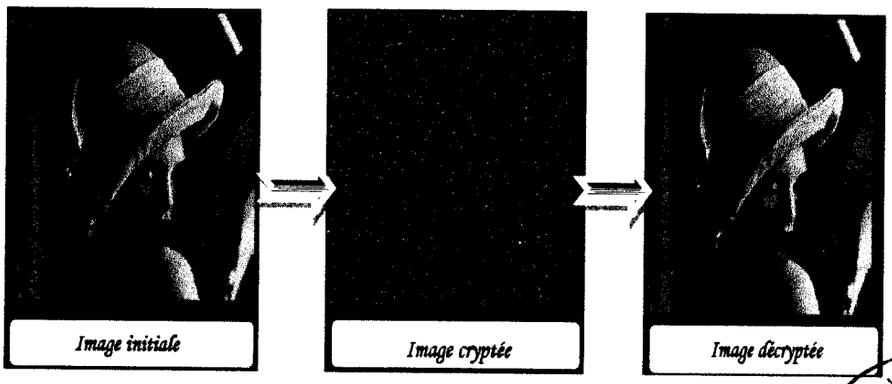
Tableau V. 4: résultat du cryptage par DES

✓ **Cryptage des images par IDEA**

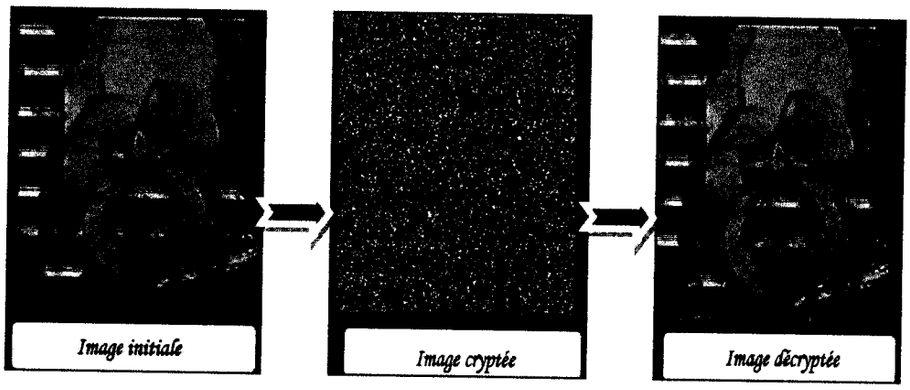
- **Image 1**



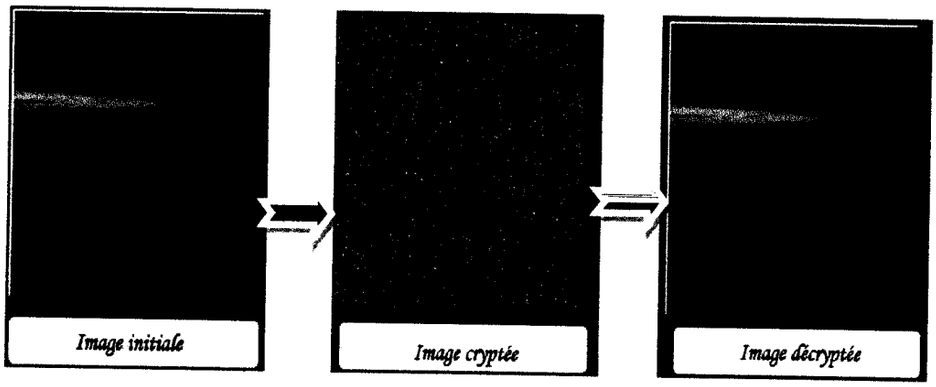
- **Image 2**



- **Image 3**



- **Image 4**

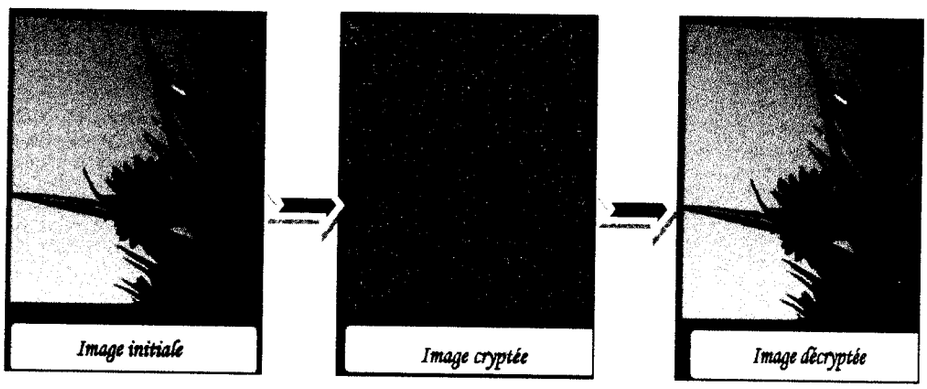


Images	**Image 1**	**Image 2**	**Image 3**	**Image 4**
Qualité	Bonne	Bonne	Bonne	Bonne
le temps de cryptage	2 .59	2.995	1.749	2 .044
le temps de décryptage	2.215	2.699	2.792	3.152

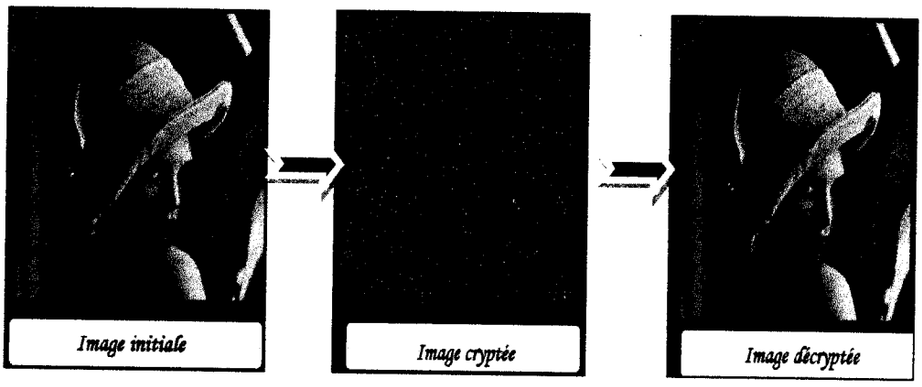
Tableau V. 5: résultat du cryptage par IDEA

✓ **Cryptage des images par RC5**

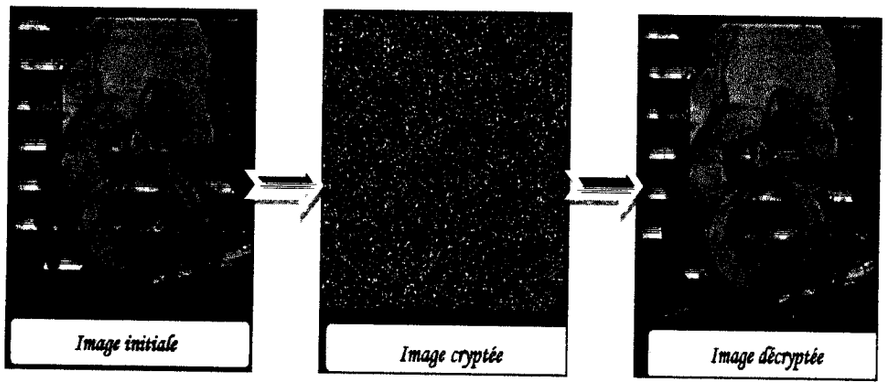
- **Image 1**



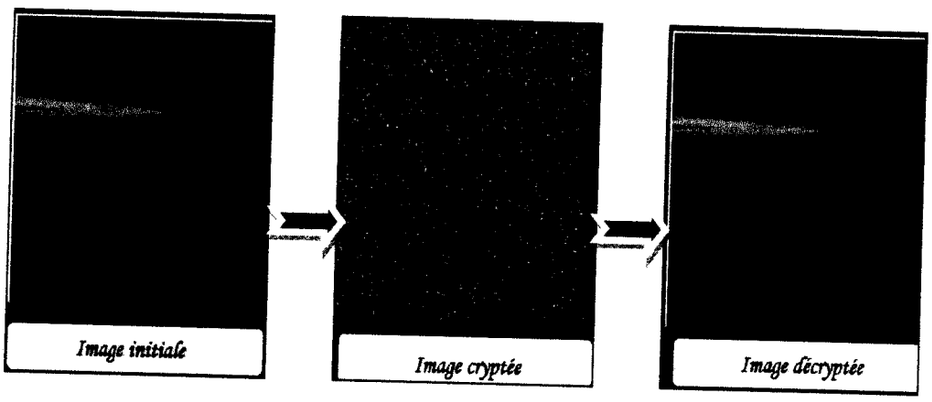
- **Image 2**



- **Image 3**



- **Image 4**



Images	**Image 1**	**Image 2**	**Image 3**	**Image 4**
Qualité	Bonne	Bonne	Bonne	Bonne
le temps de cryptage	3.697	2.449	2.106	3.104
le temps de décryptage	2.683	3.276	2.855	2.621

Tableau V. 6: résultat du cryptage par RC5

Remarque : Les algorithmes de chiffrement à clé secrète donnent une meilleur qualité de cryptage par rapport aux algorithmes classique comme ou exclusif et substitution.

Même nous observons que la vitesse de chiffrement et déchiffrement par l'algorithme IDEA est plus rapide que les deux algorithmes DES et RC5.

3. Conclusion

Nous remarquons que le taux et la vitesse de compression par huffman est plus élevée par rapport à la compression par RLE ce qui signifie que la compression par huffman est la meilleurs, et que les algorithmes à clé secrète conduit a une meilleur qualité de cryptage par rapport aux algorithmes simples. La compression d'une image après l'opération chiffrement augmenté le niveau de sécurité est donne une meilleure qualité de cryptage des images.

CONCLUSION GENERALE

Conclusion générale

Tout au long de la préparation de notre projet de fin d'études, nous avons essayé de mettre en pratique les connaissances acquises durant nos études universitaires et cela dans le but de réaliser une application de compression et de cryptage des images.

Au cours de cette mémoire, nous avons étudié et implémenté l'algorithme de compression des images huffman et de le comparer par l'algorithme de compression RLE qui étudiées l'année passée ainsi les différents algorithmes de chiffrement des images comme les algorithmes classiques : ou exclusif, et substitution et les algorithmes à clé secrète le DES, IDEA, et RC5.

Les critères de comparaisons pris en considération sont :

- ✓ Le taux de compression par l'algorithme huffman et RLE des différentes images utilisées.
- ✓ La vitesse de compression des images par les deux algorithmes huffman et RLE.
- ✓ La visibilité et la performance de l'opération de cryptage appliqué sur chaque image compressée.
- ✓ le temps de cryptage et le temps de décryptage des images compressées par huffman.

D'après l'étude comparative que nous avons réalisée, nous remarquons que le résultat issu de la compression des images par l'algorithme huffman est meilleur par rapport au résultat issu de la compression par RLE en termes de vitesse et taux de compression.

Nous avons observé aussi que les algorithmes classiques et notamment le ou exclusif offre une mauvaise qualité de chiffrement par rapport aux algorithmes de chiffrement standards comme DES, IDEA, et RC5 qui ont nettement plus efficace et donnent une meilleur qualité de chiffrement des images compressées par l'algorithme huffman.

Comme perspective de notre travaille nous souhaitons pour les prochaines projets de fin d'étude d'élargir notre travail sur la compression et le cryptage des séquences images représentés par des fichiers vidéo comme avi et mpeg.

Référence

[1]: R.C.Gonzalez, P.Wintz, Digital Image Processing, ADDISON-WESLEY, second edition, 1987.

[2] : Mr. Zitouni Athmane, Compression des images animées par le codage EZW 3D, Mémoire de fin d'étude, faculte des sciences et sciences de l'ingénier, université Mohamed khider- Biskra, 2008

[3] : Anis Benyelloul et Said-eddine Bensalem, Approches de segmentation d'images par méthodes biomimétiques, Mémoire d'ingénieur, INI 2007.

[4] : DANIEL GIEZENDANNER, TRAITEMENT NUMERIQUE DES IMAGES, INFOGRAPHIE, 14 MAI 2000

[5] : Félix Van Laethem ,les Infrastructures de clés publiques, Ecole d'Ergologie de Belgique EPFC Campus ULB de la Plaine 1050 Bruxelles, Année Académique 2006-2007

[6] : Renaud Dumont, Introduction à la Cryptographie et à la Sécurité informatique.

[7] :Sebiane tawfiq et Mejdoub fadila , mémoire d'ingénieur d'état en télécommunication
« Étude et comparaison des cryptages DES et AES »

[8] : Buchman Johann, Introduction a la cryptographie

[9] : S. Bruce, Cryptographie appliquée- Algorithmes, protocoles, 1997

[10] : G. Zennor, Cours de cryptographie, 2000

[11] : B. Beckett, introduction aux méthodes de la cryptologie, 1990

[12] : soulimane tazila, " Compression des images par RLE suivie d'une operation de cryptage par les algorithmes de chiffrement suivants : AES, BLOWFISH et 3-WAY " mémoire de projet de fin d'étude, Unuversité tlemcen, 2010.

[13] : J.P.GUILLOIS, techniques de compression des images.

[14] : M.RIGUIDEL, quelques rappels sur les techniques cryptographiques (confidentialité et intégrité mais pas disponibilité)

[15] : Emonet jean-Bruno, algorithmes de chiffrement

[16] : Implémentation des algorithmes de cryptographie pour la protection des images de types BMP et JPEG – Mr BELKEBIR Rachid , Mr CHAFAI Mohamed Radouane – 2007

[17] : Etude comparative entre les algorithmes cryptographiques appliqués sur des fichiers textes et des fichiers images (BMP) basé sur une simulation d'une attaque exhaustive – melle BENHAMOU Fatima, Melle DERRAR Ghizlane Ibtissam – 2006

Référence site web

[s1] : http://sharc.csi.dz/indcx.php

[s2] : www.kaddour.com/chap1/chap1.html

[s3] : http://plate-forme-ast.mshparisnord.org/

[s4]: www.mysic-assoc.net/spip/IMG/pdf/te-traitement-d-image-2.pdf

[s5]: http://bu.umc.edu.dz/theses/informatique/TOR4276.pdf

[s6]: http://S2-e-monsite.com/2010/03/16/85124207image-numérique.pdf.pdf

[s7]: http://fr.wikipedia.org/wiki/

[s8]: http://djoume.taket.org/infoc/TP5.pdf

[s9]: www.lagis.univ-lille1.fr/formations/sii/06-07/projets/cmucam-rapport.pdf

[s10]: http://www.unise.fr/perso/favier/vision/traitements-ponctuels.pdf

[s11]: http://www.journaldunet.com/developpeur/tutoriel/gra/

[s12]: www.scribd.com/doc/50374437/traitImage

[s13]: http://www.lirmm.fr/~ajm/Cours/04-05/DESS_TNI/Rapports/compression.pdf

[s14]: http://laure.gigou.pagesperso-orange.fr/_private/initiation/00_sommaire.htm

[s15]: http://membres.multimania.fr/frdupoux/compress/index.htm

[s16]:http://autoformation.freehostia.com/base/Imagerie-video/Compression-images.htm

[s17]: http://membres.multimania.fr/frdupoux/compress/index.htm

[s18]: http://www.commentcamarche.net/contents/crypto/crypto.php3.

[s19]: http://math.univ-lyon1.fr/~roblot/masterpro.html

[s20]: http://www.labouret.net/crypto/#1

[s21]: http://www.uqtr.ca/~delisle/Crypto/prives/blocs_rc.php

[s22]: http://www.iut-arles.univ-provence.fr

[s23]: thcscs.univ-batna.dz/indcx.php? Option=com_docman&task...4