République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université AMAR TELIDJI - Laghouat

Faculté des Sciences et de l'ingénierie

Mémoire de fin d'études

Pour l'obtention du diplôme d'ingénieur d'état en informatique

Option : Systèmes Parallèles et Distribués (SPD)

Thème

Étude et Implémentation des algorithmes de stéganographie pour le texte arabe

Réalisé par :

BEDERINA Hamza

AZZOUZ Saad

Suivi par:

Mr. BENSAAD Mohamed Lahcen

N° d'ordre :……/2009-PFE/DGI

Promotion: 2008/2009

Dédicaces 1

A mes chers parents qui m'ont soutenu durant ma vie e et ma scolarité,

A mes frères et mes sœurs,

A toute la famille SAAD.

A tous mes amis et en particulier Mounir, Naama, Hamza, Abedesslam, Ameur ,Hakim,Mamoun et toute ma promotion.

J'exprime mes sentiments les plus sincères et les plus profonds en leur dédiant ce modeste travail.

SAAD AZZOUZ

Dédicaces 2

A mes chers parents qui m'ont soutenu durant ma vie e et ma scolarité,

A mes frères et mes sœurs, A ma grande mère,

A toute la famille HAMZA.

A tous mes amis et en particulier Lakhdar.B, Saad.A, Habib.B et toute ma promotion.

J'exprime mes sentiments les plus sincères et les plus profonds en leur dédiant ce modeste travail.

HAMZA BEDERINA

Remerciements

Ce travail n'aurait pu se réaliser sans l'aide et l'amitié d'un très grand nombre de personnes.
Ces remerciements leur sont dédiés, en espérant n'avoir oublié personne dans cet exercice périlleux.

Nous aimerons tout d'abord remercier notre encadreur, Monsieur Ben Saad Lahcen pour son aide et soutient scientifique, et pour nous avoir guidé dans l'accomplissement de ce travail et pour sa patience et ses encouragements.

Nous remercions nos professeurs pour leurs enseignements qui nous ont été très bénéfiques.
On tient à remercier le département technique.

Et surtout, nous tenons à remercier nos amis proches qui nous ont apporté un soutient et nous ont souvent aidé à décompresser.

Il y a aussi d'anciens collègues dont nous espérons croiser les chemins à l'avenir, leur amitié, ainsi que leur intérêt pour notre travail, Nous ont été précieux.

Nos parents et nos familles, nous ont constamment encouragés et accompagnés dans le chemin que nous avons choisi. On leur doit beaucoup d'en être arrivé là.

Table des matières

Résumé 1

Chapitre 1 : ***Introduction à la stéganographie*** 2

1. Introduction 3
2. Définitions et Terminologies 3
3. Objectifs de la stéganographie 6
4. Conditions requises 7
5. Domaine d'utilisation 8
6. Supports & techniques de la stéganographie 8
6.1. Texte 8
6.1.1. Méthodes des "espaces" (en fin de lignes) 9
6.1.2. Méthodes des "espaces" (entre les mots) 9
6.2. Son 10
6.3. Image 10
6.4. HTML 12
6.5. Programmes 12
6.6. Disques dur ou CD 12
7. Analyse et sécurité 13
8. Conclusion 14

Chapitre 2 : ***Les méthodes de stéganographie du texte arabe*** 15

1. Introduction 16
2. Les méthodes de stéganographie et tatouage pour le texte arabe 16
1.1. Les techniques de signes diacritiques 16
2.1.1 Introduction 16
2.1.2 L'utilisation de signes diacritiques (Aabed et autres, 2007) 17
2.1.3 L'utilisation de signes diacritiques multiples (L'approche textuelle) 18
1.2. L'utilisation des points dans la stéganographie (M.H.Shirali-Shahreza et M.Shirali-Shahreza, 2006) 20
1.3. L'utilisation des extensions dans la stéganographie 21
2.3.1 L'utilisation des lettres avec points et avec extensions (Gutub et Fattani, 2007) 21
2.3.2 L'utilisation des extensions (méthode optimisée) 22
2.4 Les Caractères PSEUDO-SPACE et PSEUDO-CONNECTION (2008) 25

Chapitre 3 : ***Implémentation de quelques algorithmes*** 27

1. Introduction 28
2. L'environnement de programmation 28
3. L'implémentation des algorithmes pour chaque méthode 28
3.1. L'algorithme de la technique de signes diacritiques 29
3.2. L'algorithme de la technique de Kashida (Apres les lettres) 31
3.3. L'algorithme de La technique de ZNJ_ZWNJ 32

4. conclusion 34

Chapitre 4 : ***Comparaison entre les algorithmes implémentés*** 35

1. Introduction 36
2. Comparaison entre les algorithmes implémentés 36
2.1 Comparaison du point de vue capacité 36
2.2 La comparaison de coté imperceptibilité 37
2.3 La comparaison de coté robustesse 38
3. Conclusion 38

Conclusion Générale 39

La liste des figures

Figure.1.1. Stéganographie et concepts liés	03
Figure 1.2. Principe d'une dissimulation d'information.	05
Figure 2.1. La dissimulation de « E7 » (en hexadécimal) utilisant l'approche de signes diacritiques	18
Figure 2.2. *Les signes diacritiques du texte chiffré : (a) avant et (b) après en cercle : La découverte du dernier signe diacritique supplémentaire.*	18
Figure 2.3. Les lettres arabes	20
Figure 2.4. Le décalage du point de la lettre "Fa'a" en haut pour dissimiler la valeur de bit égale "1".	21
Figure 2.5. L'implémentation de Watermarking en ajoutant "Kashida" après les lettres.	22
Figure 2.6. L'implémentation de Watermarking en ajoutant "Kashida" avant les lettres.	22
Figure 2.7. L'insertion des bits secret en utilisant les extensions.	24
Figure 3.1. Principe d'un système dissimulation d'information.	29

La liste des Tableaux

Tableau 2.1. Les huit signes diacritiques arabes principaux	17
Tableau 2.2. Les résultats de codage de la valeur binaire 110001 selon les trois scénarios de l'approche textuelle	19
Tableau 4.1. La Capacité de dissimulation avec les trois approches.	36

Résumé

Pour pouvoir communiquer de façon secrète, il faut d'abord pouvoir communiquer tout simplement. On attachera à des messages anodins, un message secret. Afin de décoder ce message, le correspondant doit connaître un secret et/ou la technique pour déchiffrer et extraire ce message. Il est évident que ce message caché peut être lui-même codé et/ou signé en utilisant des méthodes cryptographiques. La stéganographie étudie les techniques pour communiquer de l'information de façon cachée. L'adjectif caché ne signifie pas ici que l'information est visible mais codée, il s'agit alors de cryptographie. Ici, il signifie que la présence de l'information n'est pas perceptible parce qu'enfouie dans une autre information.

L'objectif de ce travail est d'introduire cette famille de sécurité par dissimulation et en particulier la dissimulation d'information dans un texte arabe en appliquant quelques méthodes déjà réalisés par des personnes, ensuite implémenter quelques méthodes en pratique et finalement faire une comparaison entre elles et terminer avec une conclusion.

Mots clés : Stéganographie texte, Watermarking, texte arabe, dissimulation de données.

Chapitre 1

Introduction à la stéganographie

1. Introduction

Les hommes cherchent depuis l'Antiquité des méthodes pour transmettre les informations secrètement. Celles-ci reposent sur deux grands principes :

- **Le Chiffrement** : utilisé pour rendre les informations transmises illisibles pour les personnes tentant de les intercepter, donc on voit que le chiffrement ne cherche pas à dissimuler la transmission des informations, dans ce cas on parle sur la **cryptographie**.
- **La Dissimulation d'information** : utilisé pour rendre la transmission d'information indétectable en la dissimulant à l'intérieur de données transmises en clair, il s'agit dans ce cas de la **stéganographie** [1].

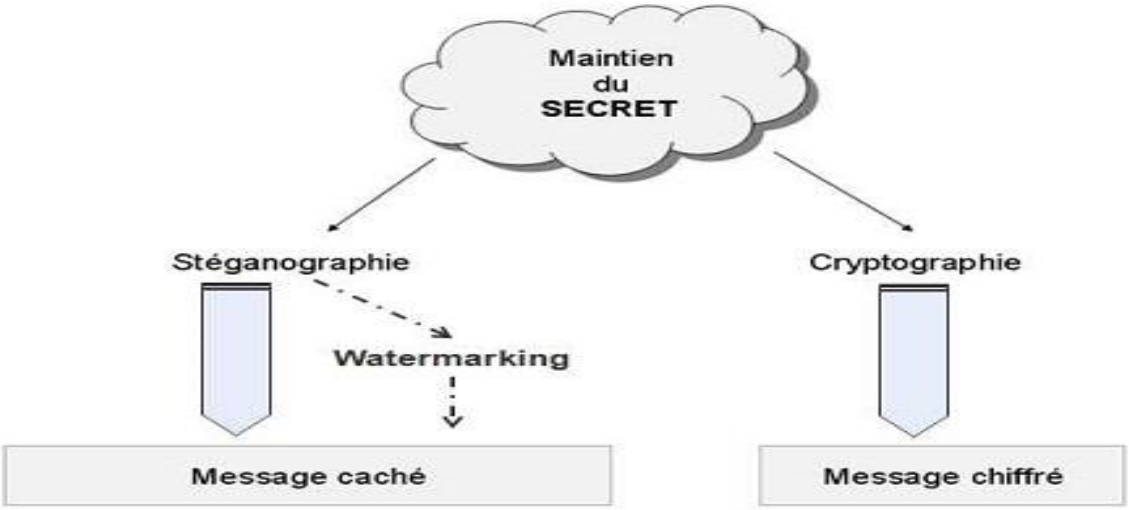


Figure.1.1 *– Stéganographie et concepts liés*

Le principal avantage de la stéganographie par rapport à la cryptographie est que le fichier renfermant le message caché reste anodin. Personne ne peut se douter qu'il cache un message. Au contraire, un message chiffré annonce clairement qu'il renferme une information, puisque, par définition, il est chiffré [2].

Ces méthodes peuvent être et sont souvent combinées mais nous allons nous concentrer sur la dissimulation d'informations qui représente La stéganographie.

La stéganographie a pour fonction de permettre la transmission sécurisée de messages dans des circonstances où la cryptographie ne peut être mise en œuvre (par exemple parce qu'elle est interdite!). Ceci a un intérêt énorme, à l'heure où la sécurité des transferts de fichiers par Internet n'est plus assurée.

2. Définitions et Terminologies

La stéganographie (en anglais: steganography ou data hiding) est encore une technique peu connue du grand public (à noter au passage qu'il ne faut pas confondre sténographie et

stéganographie). En fait le mot stéganographie tire son origine d'une étymologie grecque : **steganos** signifiant dissimulé, couvert et **graphos** signifiant écriture, dessin. Ainsi nous pouvons en déduire que la stéganographie est l'art de "cacher" des messages secrets au sein de messages plus anodins.

Stéganographie :
(*Glossaire RIFF* du NTIC (Nouvelles Technologies de l'Information et de la Communication))
C'est une Technique voisine de la cryptographie, consistant à insérer des données au sein d'autres données, de manière à transmettre un message utile au sein d'un bruit : c'est le message visible. En matière de protection de la propriété intellectuelle, la stéganographie peut être utilisée comme base d'insertion de watermarks: le stéganogramme est alors l'information d'identification du document et le bruit le document lui-même.

Sténographie : (Informatique Editoriale Larousse - Version**:** 1.0)
Procédé d'écriture formé de signes abréviatifs et conventionnels, qui sert à transcrire la parole aussi rapidement qu'elle est prononcée. (Abréviation : sténo).

En informatique, La stéganographie est l'art et la science de cacher une information privée ou secrète dans un support, apparemment anodin. Le support peut être de plusieurs types tels qu'un fichier texte, image, audio, ou vidéo, mais peut également être un système de fichier, ou un code source. La caractéristique principale est que le fichier doit sembler ne contenir aucune information sensible, c.-à-d. ne renfermer aucune information secrète.

L'insertion d'un message dans le fichier choisi implique la modification de parties de son contenu (code). Tout l'art de la stéganographie consiste à faire en sorte que ces changements soient invisibles ou inaudibles. Plus le message est réduit et le fichier volumineux, plus cette altération a des chances de passer inaperçue. D'où l'utilisation de fichiers image, son ou vidéo plutôt que de fichiers texte, de taille plus réduite. A cet aspect quantitatif s'ajoute un aspect qualitatif. Le message caché est inséré là où il sera le plus imperceptible : dans les fichiers son, par exemple, le message caché est intégré aux basses fréquences qui correspondent au bruit de fond.

La stéganographie repose sur l'idée de sécurité par l'obscurité : si personne ne sait qu'il y a un fichier caché, personne ne cherchera à le regarder ou le récupérer. Et avec tout ce qui passe sur l'Internet, et le nombre de fichiers joints que les gens s'échangent, personne ne dispose de suffisamment de ressources informatiques pour scanner tous ces transferts d'images, sons et autres fichiers.

La stéganographie est la technique consistant à insérer un fichier dans un second fichier, sans que l'aspect extérieur de ce dernier ne soit modifié (hormis sa taille).

Le terme dissimulation d'information est très général ; il désigne le fait de cacher une information dans un support. Cependant, selon les objectifs, et les contraintes qui en découlent, on distingue différentes variantes.

Tatouage :(Merriam-Webster's Collegiate Dictionary, Eleventh Edition.)
C'est une inscription sur papier résultant des différences dans l'épaisseur produite par pression de la conception de ou le modèle en métal (produisant l'inscription) quand le papier est supporté à la lumière également.

Tout d'abord, le **médium vierge** dans lequel des informations sont cachées est appelé **médium de couverture**, ou plus simplement le **médium**. Une fois que les informations sont insérées, nous utilisons alors l'expression **stégo-médium**.

D'une manière générale, nous appelons **données** l'information dissimulée dans le stégo-médium.

Le fichier "destinataire" doit être de taille suffisante pour accueillir votre fichier de données. Dans certains cas, sa taille initiale va varier, dans d'autres non. Mais ceci n'a strictement aucune importance tant que personne ne peut comparer le fichier initial et le nouveau fichier créé.

Le fichier destinataire peut être de différents types: Graphique (jpg, gif, bmp, pcx, tif, etc), Son (wav), Texte (txt), ou autres formats divers (html, pdf, etc). Par contre il n'existe pas de logiciel permettant d'utiliser tous ces types à la fois. Il vous faudra donc choisir un logiciel en fonction du type de fichier que vous désirez utiliser.

Le processus complet de dissimulation d'information repose sur deux opérations (figure 1.2) :

- la dissimulation, qui consiste à insérer l'information dans le médium ;
- l'extraction, qui récupère cette information. Le mot détection est également utilisé lorsqu'il s'agit de vérifier la présence d'une information (représentée grâce à un signal, une caractéristique particulière du médium...) dans le stégo-médium, sans pour autant vouloir l'extraire.

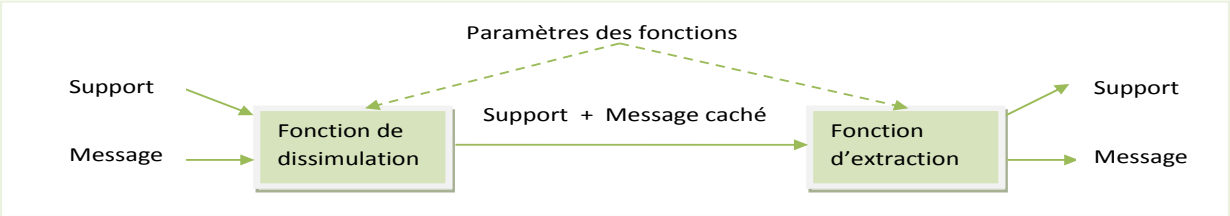


***Figure 1.2** – Principe d'une dissimulation d'information.*

Selon les objectifs poursuivis, les schémas de dissimulation d'information portent des noms différents, on en distingue trois principaux :

1) **La stéganographie** (Data Hiding) cherche à cacher un message secret, ou message plus sommairement, dans un médium de sorte que personne ne puisse distinguer un médium vierge d'un stégo-médium. La nature de l'information dissimulée ne revêt pas d'importance : il peut tout aussi bien s'agir d'un texte en clair que de sa version chiffrée. Ce message n'a a priori aucun lien avec le stégo-médium qui le transporte.

2) **Le tatouage** cherche à répondre au problème de la protection des droits d'auteur. Un client essaye d'abord de détecter la possible présence d'une marque dans un médium, puis dans l'affirmative, de vérifier si l'utilisateur a bien acheté une licence. Il s'agit bien de dissimulation d'information puisque, pour y parvenir, on insère un tatouage (ou marque, ou filigrane) dans le médium spécifique au propriétaire. Comme celui-ci souhaite protéger son médium et non une version trop déformée, l'insertion doit minimiser les modifications subies par le médium afin d'être imperceptible. Ensuite, chaque copie du stégo-médium contient la même marque, celle du propriétaire légal. Ici, la dissimulation ne signifie pas la même chose qu'en stéganographie : un attaquant sait qu'un tatouage est présent dans le stégo-médium, mais cette connaissance ne doit cependant pas lui permettre de le retirer.

3) Enfin, **le fingerprinting** cherche à permettre la détection des copies illégales d'un stégo-médium. Chaque utilisateur authentifié reçoit sa propre copie du médium qui contient une empreinte l'identifiant. Ainsi, lorsqu'une copie illégale est découverte, la lecture de l'empreinte indique la source de la fuite. A la différence du tatouage où l'origine du médium importe, le fingerprinting se préoccupe plutôt de l'utilisateur final. Chaque copie du médium contient une information différente, relative à son utilisateur, rendant alors chaque stégo-médium différent.
Lorsqu'un attaquant tente uniquement de détecter si un message transite dans un médium sur le canal de communication, on dit de lui qu'il est passif. La plupart des solutions de stéganographie ne considèrent que ce type d'attaquant, au contraire des deux domaines suivants où il est actif : l'attaquant sait alors que le stégo-médium contient une information et il tente de la modifier ou de la retirer.

3. Objectifs de la stéganographie

Malgré leurs objectifs distincts, ces trois variantes n'en requièrent pas moins des paramètres communs :

- chaque approche nécessite des données, que ce soit un message, un tatouage ou une empreinte.
- ces données sont dissimulées dans un support, le médium, qui possède plus ou moins d'importance selon le schéma : aucune pour la stéganographie, capitale pour les deux autres ;
- il est indispensable de pouvoir distinguer des personnes différentes, utilisant des données identiques dans un même médium : chacune doit donc posséder sa propre stégo-clé (ou plus simplement la clé) afin que l'insertion de ces données identiques permettent quand même de différencier les protagonistes.

Toutefois si le but de la stéganographie est de dissimuler un message sans éveiller l'attention humaine, avec la stéganographie informatique il faut également veiller à ne pas éveiller l'attention des logiciels d'analyse. Car si une image est soupçonnée de contenir un message stéganographié, on pourra toujours la soumettre à un logiciel chargé de traquer tout bruit de fond trop organisé et statistiquement non aléatoire : on peut alors facilement repérer un message stéganographié et effectuer une stéganalyse (tentative de récupération du message en clair caché).

Il faut donc que le message à cacher soit en tout point comparable à une suite de bits aléatoires : pour cela une seule solution: il faut préalablement crypter le message.

4. Conditions requises

Les objectifs de la dissimulation d'information peuvent changer de manière subtile. Classiquement, les applications sont triées en fonction de trois critères [1] :

- **l'imperceptibilité** : les données ne doivent pas être « perceptibles » dans le stégo-médium. Pour le tatouage ou le fingerprinting, l'objectif est de ne pas détériorer le stégo-médium protégé. Cependant, la contrainte est plus forte en stéganographie où il s'agit plutôt d'une indétectabilité statistique afin qu'une personne surveillant le canal ne remarque pas la présence du message ;
- **La capacité** : il s'agit du nombre de bits significatifs dissimulés dans le support.
- **La robustesse** : elle représente la manière avec laquelle le message dissimulé résiste aux modifications apportées au stégo-médium.

En stéganographie, une propriété essentielle est l'indétectabilité statistique puisqu'une personne surveillant le canal de communication ne doit pas pouvoir différencier un médium d'un stégo-médium. De plus, comme le message constitue l'information principale, la capacité doit aussi être assez élevée. Quant à la robustesse, elle constitue une défense contre les modifications subies par le stégo-médium. Néanmoins, la meilleure défense reste l'incapacité de l'adversaire à détecter le message. Ainsi, la plupart du temps, le canal ne modifie pas le stégo-médium et les besoins en robustesse sont minimes. En revanche, des mesures doivent être prises lorsque que l'adversaire est actif, soit en terme de robustesse, soit pour contrôler l'intégrité du message afin de détecter un éventuel changement dans celui-ci.

En tatouage, les contraintes diffèrent largement. Tous les utilisateurs savent, ou soupçonnent très fortement, qu'une marque est dissimulée dans le stégo-médium, et il n'est donc nul besoin de chercher à en détecter la présence. Le stégo-médium doit toutefois rester aussi proche, au sens d'une mesure de similitude sur l'espace des média, que possible de l'original afin de ne pas être dénaturé. La capacité dépend étroitement de l'application. Si le tatouage est suffisamment discriminatoire, un bit d'information suffit à répondre à la question : cette marque est-elle présente dans ce stégo-médium ? Même lorsque les données sont extraites et une mesure de confiance calculée, l'utilisation d'un seuil pour valider ou non la présence de la marque ne fournit toujours qu'un bit d'information. Au contraire, lorsque les données extraites servent ensuite à diriger une action, on considère alors que le tatouage transporte de l'information.

Enfin, les besoins du fingerprinting sont à peu près identiques à ceux du tatouage pour l'imperceptibilité et la robustesse (pour ce dernier critère, les raisons diffèrent : on ne souhaite pas voir un utilisateur distribuer sa propre copie... avec l'empreinte de quelqu'un d'autre insérée). En revanche, la capacité est importante car un médium doit contenir une empreinte spécifique à un utilisateur. Dans ces conditions, il n'est pas réaliste de se contenter, comme en tatouage, d'une réponse binaire sur la présence d'une empreinte dans un stégo-médium car il faudrait alors tester

toutes les empreintes pour un stégo-médium. Ainsi, tout comme en stéganographie, l'extraction de l'empreinte est indispensable.

5. Domaine d'utilisation

De nombreux usages peuvent exister dans des domaines très variés [2] mais souvent sensibles comme :

- **Communiquer en toute liberté même dans des conditions de censure et de surveillance :**
  Dans certain pays la cryptographie est interdite et quiconque est surpris en train de l'utiliser risque des peines importantes. En effet en utilisant une stéganographie robuste, il est impossible de suspecter la moindre trace d'un message crypté.

- **Publier des informations ouvertement mais à l'insu de tous des informations qui pourront ensuite être révélées et dont l'antériorité sera incontestable et vérifiable par tous :**
  Les attestations officielles (de diplômes, par exemple), faites sur du papier spécial, comportant éventuellement un filigrane, des dessins, une signature manuscrite, des tampons, etc. Pensons aussi aux nouvelles cartes d'identité plastifiées, aux procédés nombreux employés pour sécuriser les billets de banque, de telle sorte que les destinataires soient sûrs qu'ils proviennent bien de l'établissement habilité à les émettre et non de quelque faux-monnayeur.

6. Supports & techniques de la stéganographie

Nous passer en revue les différents supports numériques utilisés pour dissimuler des données: le texte, l'image et le son, page HTML, programme, disque dur ou CD.

6.1. Texte

La stéganographie sur un support texte est quelque chose qui ne supporte pas la fantaisie. En effet chaque retouche est directement visible par le lecteur. D'autre part un texte signé par les méthodes décrites est relativement facilement identifiable.

La dissimulation de données dans du texte est une chose bien particulière, et comme nous allons le voir elle n'a pas grande chose à voir avec l'image ou le son. Et ceci pour plusieurs raisons: on ne travaille pas avec le texte dans "l'à peu près". C'est à dire, dans une image on peut considérer qu'"endommager" celle ci avec un filtre passe-haut suffisamment "léger" ne change quasiment rien à la perception que nous allons avoir de celle-ci. Par contre avec un texte, soit le texte est comme l'original soit il ne l'est pas.

Celui ci ne permet quasiment aucune modification. Une des exigences de la dissimulation de données est d'endommager le moins possible le texte original. Pour cela nous allons utiliser la méthode dite des "espaces".

6.1.1. Méthodes des "espaces" (en fin de lignes)

Il y a 2 méthodes différentes quoi que reposant sur le même principe. La première méthode consiste à mettre des espaces en fin de ligne. On se définit un code à suivre et l'on commence:

0 espaces en fin de ligne correspondent à 0, 1 espace en fin de ligne correspond à 1.

Exemple: Ici la fin de ligne est détectée par un retour à la ligne, et les espaces dans cet exemple sont remplacés par des "_" pour plus de lisibilité.

Bonjour ceci est un message caché.	→	0 espace
A vous de le lire. _	→	1 espace
Je pense que vous commencez_	→	1 espace
à comprendre le principe.	→	0 espace
Malheureusement tout n'est pas_	→	1 espace
rose.	→	0 espace
Mais bon, nous arrivons quand même à_	→	1 espace
dissimuler un octet._	→	1 espace

Comme vous pouvez le voir dans notre exemple nous avons codé: 01101011 soit un octet de dissimulé.

Inconvénients :

- Il faut énormément de lignes pour coder peu de texte. En effet, il faut 8 lignes pour coder 1 octet. Donc imaginons que l'on veuille coder une phrase de 20 mots (chaque mot faisant environ 4 caractères) et que l'on code chaque caractère sur 7 bits (on optimise comme on peut), il va nous falloir environ 560 lignes.

- Super visible par une personne extérieure qui s'y attend un peu. Et donc facilement manipulable.

Avantages :

- Très facile et donc très simple à implémenter. Et puis ça peut marcher avec de nombreuses personnes.

- On peut rajouter des espaces pour coder plus de caractères sur moins de texte:
  En utilisant 3 espaces: 0 espace <=> 00, 1 espace <=> 01, 2 espace <=> 10, 3 espace <=> 11. Il faut alors 4 lignes pour coder 1 octet (au lieu de 8).

6.1.2. Méthodes des "espaces" (entre les mots)

Cette méthode est basée sur le même principe, mais cette fois-ci nous allons coder notre texte dans le nombre d'espaces entre chaque mot. C'est encore plus visible que la méthode précédente, mais le rapport texte codé sur texte hôte est beaucoup plus important. On se met d'abord d'accord sur une convention :

Un espace entre 2 mots suivit de deux espaces entre les 2 mots suivants <=> 0, deux espaces entre 2 mots suivit d'un espace entre les 2 mots suivants <=> 1

Pour mieux comprendre voici un exemple avec le texte suivant:

Ceci_est__essai__de_texte__caché_dans__un_texte_hôte. Vous__devez__avouer que_ce__ n'est_ pas_ très__subtil.

_ __ : 0, __ _ : 1, __ _ : 1, __ _ : 1, _ __ : 0, _ _ _ : 1, __ _ : 1, _ _ _ : 0 => 01110110 soit un octet.

Le rapport texte à coder (à cacher) sur le texte support (le Cover-médium). Il vous faut 2 mots pour un bit, donc pour cacher une phrase de 250 bits par exemple, il faut un texte comme un support de 500 mots, en comptant 10 mots par ligne en moyen on arrive à 50 lignes. On gagne un rapport d'espace libre de 80% : (250-50)*100/250 comparé à la méthode des espaces en fin de ligne.

6.2. Son

De faibles variations, imperceptible pour l'oreille, dans les basses fréquences ou ce que l'on appelle le bruit de fond peuvent contenir une grande quantité d'information. Un grésillement infime peut cacher des secrets.

Evidemment, ce bruit doit de préférence être transmis de façon numérique sans quoi les vraies pertes de transmission pourraient effacer entièrement le message caché.

6.3. Image

Une image est constituée de points (ou pixels) qui sont autant de données permettant à l'ordinateur de recréer l'image lors de la lecture du fichier. Il est possible d'insérer de nouvelles lettres et chiffres dans ces données, afin de constituer un message caché dans l'image initiale.

« Chaque pixel est constitué de trois couleurs : le rouge, le vert et le bleu. Certains de ces points peuvent être remplacés par une autre information sans que les changements apportés dans l'image soient perceptibles à l'œil humain. ».

Reste que, pour pouvoir lire ce message caché, la personne recevant l'image doit connaître la clef permettant de lire les informations contenues dans celle-ci. Sans cette clef privée, il est impossible de lire le contenu caché, et l'image garde tout son mystère.

Loin d'être complexe, les méthodes pour cacher une image numérique dans une autre image sont simples. Plusieurs techniques sont utilisées. La première est basée sur les couleurs présentes dans une photo. De façon similaire au son, le spectre des couleurs peut être traduit par des courbes.

Les logiciels de codage insèrent dans ces spectres (rouge vert bleu) de micros variations qui ne seront pas perceptibles pour les personnes qui regardent les photos. Tous les types de données peuvent être insérés de cette façon, seule la capacité de l'image à "absorber" ces données cachées, limite les transferts. Une autre méthode consiste à modifier imperceptiblement la valeur

de chaque pixel de l'image. Là aussi, la dégradation de l'image passe inaperçue et seul le logiciel adéquat permet de retrouver les données cachées.

De façon plus sophistiquée, il est aussi possible de cacher des données dans des constructions mathématiques (fractales en particulier). Ces constructions qui se répètent à l'infini selon une suite de chiffres peuvent inclure des bouts de fichiers sans en altérer la représentation graphique.

Une image peut en cacher une autre. En remplaçant les bits qui altèrent le moins (bits de poids faible) l'image de couverture par les bits les plus représentatifs (bits de poids fort) de l'image à cacher. Pour une image codée sur 32 bits, on peut en remplacer environ 4 bits sans problèmes. Voici quelque exemple :

Format JPEG : Les images les plus utilisées sur le net sont des images format JPEG.

La compression JPEG (Joint Photographic Experts Group) utilise « Discrete Cosine Transform » le DCT pour transformer d'une façon successive chaque bloc de 8×8-pixel d'une image en un 64 DCT coefficients. Les bits les moins signifiants du DCT coefficient sont utilisés comme des bits redondants pour dissimuler le message secret.

La modification d'un seul DCT coefficient affect les 64 pixels d'une image.

Format GIF : Dans certaines images de format GIF (Graphic Interchange Format), les couleurs des pixels ne sont pas indépendantes mais codées selon une palette de 256 couleurs. On donne une valeur comprise entre 0 et 255 à chaque pixel et lorsque l'on lit les pixels, on se réfère à la palette pour y mettre la couleur.

Exemple :

11111111 <=> 255 : et la couleur 255 est définie comme étant un blanc vif.

11111110 <=> 254 : et la couleur 254 est définie comme étant un bleu foncé.

En ne changeant que le dernier bit vous allez complètement détruire notre image. Se qui n'est pas vrai pour le format JPEG car la modification se fait dans le domaine des fréquences et non pas dans le domaine spatial.

Il s'agit ici du domaine le plus vaste et certainement du plus intéressant. En effet, les applications sont très nombreuses, et les méthodes à utiliser sont assez complexes (comparées à celles utilisées pour le texte). Le marquage d'images a de nombreux buts: copyright des images (très important à l'époque d'Internet où des milliers d'images circulent sur le web), mais aussi marquage de papiers ou de billets (pour éviter le photocopillage), vérification de l'intégrité de documents, etc. A chaque utilisation correspond une méthode.

En fait les techniques doivent répondre à certaines règles très importantes et souvent difficiles à concilier:

- Endommager le moins possible le support sur lequel le marquage va avoir lieu. (L'œil humain ne doit pas être choqué).
- Le marquage doit pouvoir supporter le plus grand nombre de transformations possible sans être dégradé (compression JPEG, filtres, passage analogique-numérique, changement de palettes etc.. voir les différentes attaques possibles sur le marquage).
- La complexité de l'algorithme doit être minimum afin de pouvoir effectuer le marquage et/ou la détection en temps réel.
- Le marquage peut avoir lieu au niveau de l'image dans le domaine des fréquences.

6.4. HTML

Enfin certains logiciels de stéganographie se proposent de cacher des messages dans des pages HTML : ils ne font que toucher au source pour camoufler le fichier secret en insérant des espaces entre balises, variant minuscules et majuscules dans les balises,... Astucieux mais cela peut toutefois se détecter par analyse statistique et même par un coup d'œil au source dont l'indentation exotique pourra attirer l'attention.

Exemple :

Dans toute page HTML comme celle-ci, il y a (ou il peut y avoir) des indications qui n'apparaissent pas sur l'écran des navigateurs – ce sont les balises 'META'. On peut y dissimuler tout ce qu'on veut. Le non-initié n'y verra rien. Mais le procédé n'est pas très performant, puisqu'il suffit de cliquer, généralement dans le menu Affichage du navigateur, sur Source de la page pour voir tout ce que contiennent les balises META.

6.5. Programmes

- Dans les "zones mortes" du code (commentaires, branche morte d'un organigramme)
- Dans le programme lui-même à l'aide d'une commande jamais utilisée (quintuple clic)

6.6. Disques dur ou CD

Les formes de stéganographie informatique ne s'arrêtent pas à cela : il est également possible de cacher des fichiers à l'intérieur de l'espace disque libre d'un disque dur ou d'une disquette. Car il faut bien différencier 2 choses : ce qui est inscrit sur le disque et ce qu'y est inscrit dans la table d'allocation du disque géré par le système d'exploitation. En effet pour répertorier tous les fichiers d'un disque, le système d'exploitation accède et met à jour une table d'allocation des fichiers : seuls les entrées de fichiers figurant dans cette table sont accessibles par le système d'exploitation. L'idée est alors d'inscrire un fichier physique sur le disque dur sans que la table d'allocation en ait connaissance : ainsi le fichier est sur le disque mais le système d'exploitation ne le voit pas. Pour ce faire il suffit d'utiliser un logiciel spécialisé écrivant et lisant directement sur le disque sans passer par le système d'exploitation. Un tel système de stéganographie est efficace si l'intercepteur n'en a pas conscience, par contre si celui-ci est au courant alors il n'aura aucune difficulté pour retrouver le fichier. L'autre point faible de cette technique est que le disque ne doit subir aucune modification en écriture par le système d'exploitation une fois que la stéganographie

a été effectuée car sinon le système d'exploitation pourrait écraser sans en avoir connaissance le fichier caché car les secteurs du disque abritant ce fichier sont considérés comme espace libre pour le système d'exploitation.

Inconvénients :

Facile d'aller récupérer des fichiers sur le disque à l'aide de logiciels spécialisés.

- En plus, on crypte les données les fichiers non répertoriés sont considérés comme inexistant.
- le système peut malencontreusement réécrire dessus.
- données redondantes.

7. Analyse et sécurité

Les principes de sécurité s'appliquent aussi à la dissimulation d'information, la sécurité doit reposer sur la clé et non sur l'algorithme. La stéganographié pure est donc loin d'être sure.
La dissimulation d'information est sure lorsque en appliquant l'algorithme d'extraction on obtient une suite aléatoire que des informations soit dissimulées ou non dans le médium de couverture.

Les attaques sur les documents stéganographique sont les suivantes :

- **Stego-only attack**. Seul le stégo-medium est connu.
- **Chosen stego attack**. Le stégo-medium et l'algorithme sont connus.
- **Known stego attack**. Le stégo-medium, l'algorithme et le médium de couverture sont connues.
- **Known cover attack**. Le stégo-medium et Le medium de couverture sont disponibles.
- **Known message attack**. Certaines parties du message caché sont connues de l'utilisateur. L'attaquant va essayer de retrouver dans le stégo-medium les parties du message qu'il connaît afin de faciliter l'analyse des documents futurs. Même avec le message cette attaque est très difficile et généralement considérée comme équivalent à l'attaque stego-only
- **Chosen message attack**. Le stéganalyse génère un stégo-medium à l'aide de l'algorithme et du message de son choix. Le but est d'observer le résultat pour cracker l'algorithme.

Comme en cryptographie les attaques peuvent avoir pour but d'effacer les données cachées (rare), de les lire ou de les changer (très difficile).

L'attaquant doit tout d'abord savoir si le document contient des données cachées :

- Détection de différences par rapport à l'original.
- Analyse des discontinuités.
- Recherche de schéma répétitif ou les clés les plus utilisées (codage par déformation) Il faut ensuite repérer les données insérées en se basant sur la phase précédente puis reconstituer les données de départ.

8. Conclusion

La technique qui aurait été employée, la stéganographie, consiste à cacher un message dans un support "innocent". Elle peut, de surcroît, se combiner à la cryptographie, qui se charge de dissimuler le sens de la missive et non plus son existence.

Le résultat est alors particulièrement efficace. Le message secret s'abrite d'abord derrière son invisibilité. En cas de découverte, il restera à le décoder. Un défi pour les services secrets qui, dans le cas du terrorisme, doivent réaliser ces deux opérations au plus vite pour que l'information recueillie ne soit pas obsolète.

Il faut savoir que si la stéganographie est très pratique, son utilisation informatique est détectable. Il ne faut pas oublier que cela est avant tout un code informatique. La sécurité de la stéganographie repose sur le fait que le message ne sera sans doute pas détecté.
Enfin, on peut dire que le filigrane électronique (Watermarking) est sans doute la principale application industrielle des algorithmes de dissimulation.

Chapitre 2

Les méthodes de stéganographie du texte arabe

1. Introduction

La stéganographie des textes a essayé d'implémenter des différentes méthodes et approches. La plupart de ces approches connues cachent des données en faisant des modifications minimales sur l'écriture des caractères ou sur les espaces [4].

La stéganographie du Texte est le type le plus difficile de Stéganographie car il y a peu d'informations redondantes dans un fichier texte par rapport à une image ou un fichier son [5].

La structure des documents texte est identique avec ce qu'on observe, tandis que dans d'autres types de documents tels que l'image, la structure du document est différent de ce qu'on observe. Donc, dans de tels documents, on peut cacher l'information en introduisant des changements dans la structure du document sans faire un changement remarquable sur le document de la sortie concernée [6].

Contrairement à d'autres média tels que les sons et les vidéos, l'utilisation des documents texte a été répandu depuis très ancien temps. Cela s'est étendu jusqu'à aujourd'hui et à perpétuel, l'utilisation du texte est préféré par rapport à d'autres média, parce que le texte occupe moins d'espace, communiquent plus d'informations et besoin du moins prix pour l'impression ainsi que certains autres avantages [6].

Aujourd'hui, les systèmes informatiques ont facilité La dissimilation d'information dans le texte. Plusieurs applications de dissimilation d'informations dans le texte sont développées pour cacher des informations dans les textes électroniques et dans les pages Web [7].

La plupart des méthodes sont développées pour les textes anglais, mais il y a quelques d'autres méthodes stéganographique pour les autres langues [6].

2. Les méthodes de stéganographie et tatouage pour le texte arabe

1.1. Les techniques de signes diacritiques

2.1.1 Introduction

La langue arabe, écrite de droite à gauche, est basée sur un système alphabétique qui utilise 28 lettres fondamentales. À la différence de la langue anglaise, l'arabe ne différencie pas entre le majuscule et minuscule ou entre les lettres écrites et imprimées. De plus, la langue arabe utilise différents symboles comme marques diacritiques, ou simplement des signes diacritiques qui sont connus aussi comme « **Harakat** » [8].

Il y a huit différents symboles diacritiques principaux en arabe, ils sont affichés dans le tableau 2.1. D'autres signes diacritiques existent aussi mais sont en dehors de la portée de notre étude.

Fatha	ـَ	Tanween Fatha	ـً
Dhamma	ـُ	Tanween Dhamma	ـٌ
Kasrah	ـِ	Tanween Kasrah	ـٍ
Sukkon	ـْ	Shaddah	ـّ

Tableau 2.1. *Les huit signes diacritiques arabes principaux*

Comme la plupart des langues écrites basés sur les signes diacritiques, le but principal d'utiliser des signes diacritiques dans la langue arabe est de changer la prononciation d'un phonème ou de différencier entre des mots d'orthographe similaire [8].

Des propositions sont apparues visent à utiliser les avantages de signes diacritiques dans les scripts arabes pour implémenter la stéganographie et pour l'échange d'informations cachées. Les huit symboles diacritiques d'arabe sont utilisés pour cacher les bits binaires dans le media de couverture originale [8].

2.1.2 L'utilisation de signes diacritiques (Aabed et autres, 2007)

L'utilisation de signes diacritiques dans la langue arabe Standard écrite est optionnelle. Cela signifie que les lecteurs arabes originaux peuvent lire correctement un texte sans signes diacritiques en appliquant la grammaire de la langue arabe à ce texte. Dans ce travail, cette propriété est utilisée pour définir un plan stéganographique pour cacher des données binaires dans le texte arabe.

L'équipe, qui a proposé cette approche, a constaté qu'en arabe Standard, la fréquence du signe diacritique, à savoir "Fatha", est égale à la fréquence totale des sept autres signes diacritiques. Donc ils ont assigné dans cette approche le diacritique "Fatha" à la valeur de bit égale "1" et les sept signes diacritiques restants ont été assignés à la valeur de bit égale "0". Donc dans le stégo-texte chaque "Fatha" représente "1" et chaque signe diacritique non "Fatha" représente un " 0".

Pour mettre en application cette approche, un texte arabe diacritisé est utilisé comme un Objet de Couverture (**Cover Object** en anglais). Puis, un programme informatique lit le premier bit de données requises pour être inséré. Si le premier bit était "1" et le premier signe diacritique dans le média de couverture était "Fatha", le signe diacritique est gardé dans le média de couverture et l'index pour les deux données insérées et le média de couverture est incrémenté. Cependant, si le signe diacritique n'est pas "Fatha" et la valeur de bit est "1", le signe diacritique est enlevé de média de couverture et, dans le même temps, l'index de média de couverture seulement est incrémenté pour lire le signe diacritique suivant. Le même processus est exécuté pour la valeur de bit zéro, sauf qu'un zéro cherchera tous les sept autres signes diacritiques au lieu du Fatha.

Un exemple à cette approche peut être vu dans la figure 2.1.

Cover Object	حَدَّثَنَا سُفْيَانُ عَنْ يَحْيَى
Secret Object	**E7=(11100111)**
Stego Object	حَدَّثَنا سُفْيَان عَن يَحْيَى

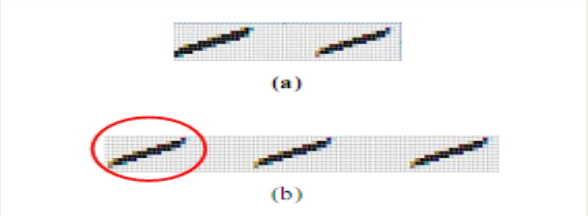
Figure 2.1. *La dissimulation de « E7 » (en hexadécimal) utilisant l'approche de signes diacritiques*

2.1.3 L'utilisation de signes diacritiques multiples (L'approche textuelle)

L'idée est apparue a travers comment les ordinateurs affichent/impriment des signes diacritiques arabes. Pour la plupart des polices de caractères arabes, quand le code d'une marque diacritique est produit, l'image de la frappe correspondante est rendue à l'écran/imprimante sans changer l'endroit du curseur. Un tel affichage sans déplacer mène à la possibilité de taper des multiples instances d'un signe diacritique d'une façon presque invisible.

Un programme informatique conscient de la présence et de la signification de tels signes diacritiques peut les détecter et interpréter. Par exemple, un programme peut être conscient que des signes diacritiques multiples existent dans un message. Alors Il peut les extraire facilement, comme le montre la figure 2.2.

Figure 2.2. *Les signes diacritiques du texte chiffré : (a) avant et (b) après en cercle : La découverte du dernier signe diacritique supplémentaire.*



Nous soulignons sur le mot presque quand on autorisant l'invisibilité de signes diacritiques supplémentaires. Ce fait est parce que la dactylographie multiple d'un caractère diacritique pourrait avoir un effet sur la Sorite d'affichage/d'impression dans quelques polices de caractères.

Il y a deux cas intéressants : un cas pour indiquer seulement la première marque diacritique et cacher des frappes supplémentaires et un cas pour obscurcir les signes diacritiques avec les frappes supplémentaires.

Il y a deux approches pour exploiter les idées au-dessus : l'approche textuelle et l'approche d'image. Chaque approche a ses avantages du point de vue de la métrique de stéganographie typique: la sécurité, la capacité et la robustesse. Dans notre étude on a intéressés seulement pour l'approche textuelle.

L'approche textuelle choisit une police de caractères qui cache complètement les signes diacritiques supplémentaires. Alors, elle utilise un scénario de codage pour cacher les bits secrets dans un nombre aléatoire arbitraire de signes diacritiques répétés mais invisibles. Vraiment, un algorithme spécial est nécessaire pour récupérer l'information cachée.

a. Les scénarios du méthode "Direct and Blocked Value"

Il y plusieurs scénarios pour se servir de cette approche. Un scénario de cette méthode réalise une capacité arbitraire : le message entier peut être caché dans une marque diacritique simple en frappant (ou en générant) un certain nombre de frappes de diacritiques supplémentaires égales au nombre binaire représentant le message. Par exemple, pour cacher la chaîne de caractères binaire $(110001)_2$ = 49 d, on doit ajouter 49 diacritiques supplémentaires à la première diacritique rencontrée dans le média de couverture.

On voie que Le nombre de diacritiques supplémentaires pourrait être énorme! Dans ce cas une solution peut être proposée consiste à réaliser le scénario précédent sur des blocs de nombre limité de bits.

Pour ce scénario, considérer le même exemple de $(110001)_2$ comme un message secret, nous répétons le premier signe diacritique 3 fois plus ($3 = (11)_2$); le deuxième, aucune fois ($0 = (00)_2$); et le troisième, une fois plus ($1 = (01)_2$).

b. Le scénario RLE (run length encoding)

Le scénario RLE (run-length encoding) peut ressembler à la méthode de compression. Dans ce scénario, nous répétons le premier signe diacritique dans le texte autant que le nombre des "1" consécutifs qui apparus au début du flot de message secret. De même, le deuxième signe diacritique est répété de manière équivalente au nombre des "0" consécutifs dans le message secret. De la même manière, tous les signes diacritiques ordonnés **impair** sont répétés selon le nombre des "1" consécutifs suivants, et tous les signes diacritiques ordonnés **pair** selon les "0".

Pour mieux comprendre cette idée, nous reproduisons l'exemple précédent pour le cas **RLE**, aussi. L'algorithme impliquera la répétition du premier signe diacritique 2 fois (2 = le nombre des 1 dans $(11)_2$); le deuxième, 3 fois (3 = le nombre des 0 dans $(000)_2$); et le troisième, une fois (pour 1).
Nous obtenons les résultats dans le tableau 2.2.

Scénario	Diacritiques supplémentaires
1er scénario (Stream)	49.
2éme scénario (taille du bloc=2)	3 + 0 + 1 = 4.
3éme scénario (RLE)	(2-1) + (3-1) + (1-1) = 3.

Tableau 2.2. *Les résultats de codage de la valeur binaire 110001 selon les trois scénarios de l'approche textuelle*

1.2. L'utilisation des points dans la stéganographie (M.H.Shirali-Shahreza et M.Shirali-Shahreza, 2006)

"Shirali-Shahreza" a proposé une méthode de sécurité spéciale pour les lettres arabes et persanes. Leur plan dépend des points hérités dans les lettres arabes et persanes, dont qui sont très similaires. La concentration dans cette étude sera sur la stéganographie relatives à la langue arabe.

Bien que, à la fois les langues arabe et anglais, ont des points dans leurs lettres, la quantité de lettres avec des points très diffèrent. La langue anglaise par exemple a des points dans seulement deux lettres, petit "i" et petit "j", alors que l'arabe a dans 15 lettres de ses 28 lettres d'alphabet comme montré dans Figure 2.3. Ce grand nombre de points dans les lettres arabes a fait les points dans n'importe quel texte arabe donné et peut être utilisé pour la stéganographie et la sécurité d'information.

Lettres sans points	*Lettres avec points*
ا ح د ر س ص ط ع ك ل م ه و	ب ت ث ج خ ذ ز ش ض ظ غ ف ق ن ي

Figure 2.3. *Les lettres arabes*

La stéganographie de point de "Shirali-Shahreza" [4] cache l'information dans les points des lettres. Pour Être spécifique; ils cachent l'information dans l'emplacement des points dans les lettres pointues.

Premièrement, l'information cachée est vue comme binaire avec premiers plusieurs bits (par exemple, 20 bits) pour indiquer le nombre de bits cachés à enregistrer. Ensuite, le texte de medium de couverture est parcouru. Chaque fois qu'une lettre pointue est détectée l'emplacement de son point peut être affecté par le bit caché d'information. Si le bit de valeur caché est "1" le point est légèrement décalé vers le haut; autrement, l'emplacement de point de caractère concerné reste inchangé.

Ce processus de décalage de point est montré dans la Figue 2.4 pour la lettre arabe "Fa". Afin de détourner l'attention de lecteurs, après avoir caché tous les informations, les points des caractères restants sont aussi changés au hasard [4]. Noter que, comme mentionné précédemment, le nombre de bits cachés est connu et également cachée dans les 20 premiers bits.

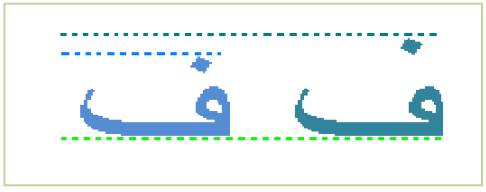


Figure 2.4. *Le décalage du point de la lettre "Fa'a" en haut pour dissimiler la valeur de bit égale "1".*

Cette méthode de décalage de point peut avoir ses avantages dans la sécurité et la capacité; elle présente un bon stockage du secret, c'est-à-dire un grand nombre de bits cachés dans n'importe quel texte arabe. Cependant, elle a l'inconvénient principal dans la robustesse ce qui la rendre non pratique. Par exemple, l'information cachée détruite dans toute nouvelle frappe ou balayage. Le texte de sortie a une structure fixe due à l'utilisation d'un seule police de caractères, donc le récepteur ne sera pas capable d'extraire le message secret si le police de caractères de sortie n'est pas installée sur sa machine. En fait, cette méthode de protection des données est appropriée pour être classifiée comme watermarking au lieu de stéganographie.

Cette méthode n'attire pas d'attention et peut cacher un grand volume d'information dans le texte.

1.3. L'utilisation des extensions dans la stéganographie

2.3.1 L'utilisation des lettres avec points et avec extensions (Gutub et Fattani, 2007)

En profitant du stéganographie de point de (Shirali-Shahreza, 2006) et essayant de surmonter l'aspect négatif de robustesse, une nouvelle méthode a proposé pour cacher l'information dans n'importe quelles lettres au lieu des pointus seulement.

Une autre méthode proposée utilise les lettres pointues avec le caractère d'extension (Kashida "_") pour tenir le bit secret "1" et les lettres non pointues avec Kashida pour tenir le bit secret "0".

Noter que le "Kashida" n'a aucun effet sur le contenu du texte. Il a un code hexadécimal de caractère standard : **0640** dans le système Unicode. En fait, ce caractère Kashida arabe est considéré comme un caractère redondant destiné pour formater la dactylographie électronique arabe.

Le seul motif d'utiliser Kashida est que pas toutes les lettres peuvent être étendues avec ce caractère d'extension en raison de leur position dans les mots, ainsi de la structure naturelle d'écriture arabe. Le "Kashida" peut être ajouté seulement dans les emplacements entre les lettres connectées du texte arabe; c'est-à-dire. Kashida ne peut pas être placé après les lettres à la fin du mot ou avant la lettre au début.

Cette hypothèse proposée de watermarking signifie que chaque fois qu'une lettre ne peut pas avoir une extension ou trouvé intentionnellement sans extension on le considère que ne tenir aucun bit secret.

Cette méthode de tatouage numérique proposée peut avoir le choix d'ajouter Kashida avant ou après les lettres.

Supposer que nous ajoutons Kashida après les lettres. La Figure 2.5 montre un exemple explique ce processus de watermarking en détail. Nous choisissons d'abord les bits secrets à être cachés (disant 110010) commençant avec les bits les moins significatifs, c'est-à-dire on va parcourir la suite des bits de droit à gauche. Le premier bit secret trouvé est '0' pour être caché dans une lettre non pointue. Le texte de couverture est lu du droit à gauche en raison de la direction régulière arabe. On constate que la première lettre non pointue dans le texte de couverture est la lettre 'mim'. Ce 'mim' devrait tenir le premier bit secret '0' en ajoutant le caractère d'extension après lui.

Le deuxième bit secret est '1' et la deuxième lettre pointue du texte de couverture, connue sous le nom de 'noun'. Cependant, cette position de lettre ne peut pas permettre l'extension, nous forçant à l'ignorer. La lettre pointue possible suivante à être étendre est 'ta'. Noter que la lettre pointue ' noun ' avant 'ta' n'est pas utilisée en raison de sa non faisabilité pour ajouter le caractère d'extension après lui.

Watermarking bits	110010
Cover-text	من حسن اسلام المرء تركه مالا يعنيه
Output text	من حسن اسلام المرء تركه مـالا يـعنـيه

Figure 2.5. *L'implémentation de Watermarking en ajoutant "Kashida" après les lettres.*

Le même exemple (110010) pour ajouter extensions avant les lettres, est montré dans la Figure 2.6.

Watermarking bits	110010
Cover-text	من حسن اسلام المرء تركه مالا يعنيه
Output text	من حسن اسلام المرء تركه مالا يعنيه

Figure 2.6. *L'implémentation de Watermarking en ajoutant "Kashida" avant les lettres.*

L'utilisation de ce caractère présente la sécurité et la robustesse. Mais, il pourrait avoir quelques inconvénients dans la capacité du media de couverture si le nombre de bits secrets dans l'Objet Secret est grand.

2.3.2 L'utilisation des extensions (méthode optimisée)

L'objet secret est caché dans la forme des zéros et des uns ce qui représente l'Unicode 8-bit de chaque caractère (utilisant l'encodage **UTF-8**). Un inconvénient commun dans toutes les approches précédentes consiste à insérer les bits sans avoir un peu d'optimisation. Ce signifie que ces approches insèrent 8 bits de chaque caractère dans le texte de Couverture.

La méthode originale profite du travail qui est proposé par Gutub et Fattani par l'utilisation de caractère d'extension.

On ajoute une extension pour représenter le bit=0 et deux extensions consécutives quand le bit=1. L'extension est placée après n'importe quelle lettre qui peut la tenir. La partie d'optimisation de l'algorithme s'occupe du message à cacher, c'est-à-dire l'Objet Secret. On sait que la langue arabe a 28 lettres. Mais il y a des formes spéciales d'une lettre comme dans la lettre "Alef (ا) ": (أ, إ, آ, ...) qui sont utilisés dans l'écriture arabe et chacun a une représentation d'Unicode. Donc le nombre de lettres et de formes résume à plus de 32 et moins de 64, et comme chaque lettre et forme sont représentés sur 8 bits, On utilise un table de caractères dans laquelle chaque lettre a été assignée un code de 6 bits pour sauver 2 bits. Dans la table de caractères, nous assignons des codes de 6 bits commençant par 000000 et incrémentés par "un" pour toutes les lettres et formes ordonnées alphabétiquement. Par conséquent, en sauvant 2 bits par l'implémentation de la table de caractères, signifie qu'on va améliorer la caractéristique de capacité de cette méthode.

Il y a un souci est que quand l'Objet de Couverture est une page de longueur par exemple et l'Objet Secret est seulement un mot, alors, le Stégo-Objet contient des extensions seulement dans les premières lignes. Donc ce sera soupçonneux pour un lecteur qui pourrait déduire l'existence d'un message caché dans le texte.

Donc on doit résoudre ceci par la création d'un caractère spécial appelé "le caractère de terminaison" qui a le code 111111 et il sera inséré juste après la dernière lettre du message, c'est-à-dire l'Objet Secret. Après cette "lettre de terminaison", l'algorithme ajoute aléatoirement des extensions au texte entier juste pour se débarrasser de n'importe quelle sorte de doute. Finalement, le fait d'extraire le message à partir de Stégo-Objet est fait en collectant les extensions et quand 6 bits sont collectés, l'algorithme vérifie la table de caractères pour déterminer la lettre correspondante. Quand il détecte le caractère de terminaison, il s'arrête.

La Figure 2.7 montre comment la méthode fonctionne. La première lettre dans l'Objet Secret est "Ba'a" qui a le code 000001 dans la table de caractères. Nous pouvons voir que la première et la deuxième lettre dans le premier mot dans l'Objet de Couverture peuvent tenir des extensions, donc une extension est ajoutée après chacun représentant le premier et le deuxième bit = 0 du code **00**0001, comme montré dans le Stégo-Objet. La troisième lettre ne peut pas tenir une extension, donc elle est gardée comme elle est. Le deuxième et le troisième mot dans le Stégo-Objet tiennent le 4ème, le 5ème et le 6ème bit = 0 du code, c'est-à-dire 00**000**1. Finalement, le quatrième mot tient deux extensions consécutives pour représenter le bit restant du code qui est **1**, c'est-à-dire 00000**1**.

Secret Object	بدأ اختبار
Cover Object	ميزة هذا النوع من المعالجات أنه يقضي مدة ثابتة في تنفيذ أي تعليمة, ومقدار هذا الوقت هو دورة واحدة يحدد زمنها أطول تعليمة من مجموعة التعليمات وهي تعليمة القراءة من وحدة الذاكرة, وهذا يعني سهولة كبيرة في تصميم المعالج
Stego Object	ميزة هذا النوع من المعالجات أنه يقضي مدة ثابتة في تنفيذ أي تعليمة, ومقدار هذا الوقت هو دورة واحدة يحدد زمنها أطول تعليمة من مجموعة التعليمات وهي تعليمة القراءة من وحدة الذاكرة, وهذا يعني سهولة كبيرة في تصميم المعالج

Figure 2.7. *L'insertion des bits secret en utilisant les extensions.*

Le fait d'ajouter une extension quand le bit de l'Objet Secret est **0** et deux extensions consécutives quand le bit est **1** après n'importe quelle lettre ou forme qui peut accepter les extensions est méfiant d'une façon ou d'une autre. Un lecteur trouvera des extensions à n'importe quelle position applicable pour une extension dans le Stégo-Objet.

Une double solution d'impact est proposée qui peut résoudre ce problème de méfiance et peut améliorer plus la caractéristique de capacité. Alors l'algorithme de cette méthode est modifié pour ne pas ajouter une extension quand le code de lettre dans la table de caractères a deux bits **zéro** consécutifs. En d'autres termes, le code de lettre "Ba'a" qui est 000001 ne sera pas inséré dans l'Objet de Couverture comme : 1 extension, 1 extension, 1 extension, 1 extension, 1 extension, 2 extensions. Il sera inséré plutôt comme : aucune extension, aucune extension, 1 extension, 2 extensions. Évidemment, on a sauvé encore deux emplacements supplémentaires de bits (extension) et on a permis la diminution d'extensions dans le Stego Objet pour enlever la méfiance qui peut survenir.

Une modification finale a été apportée pour améliorer plus la capacité. La méthode est modifié pour détecter l'Objet Secret, avant de l'insérer et y faire des statistiques sur lui pour trouver les lettres (existantes) les plus fréquentes. Alors, la méthode assignera les codes des lettres qui ont des zéros consécutifs plus dans la table de caractères aux lettres les plus fréquentes. Par exemple, si nous constatons que l'Objet Secret a la lettre "Lam" (ل) comme la lettre la plus fréquente qui peut se produire et ensuite la lettre " Noun " (ن), alors la méthode assignera les codes 000000 à "Lam" et 000001 au " Noun " dans la table de caractères. Donc nous améliorerons la capacité en utilisant la fréquence de l'occurrence des lettres. Noter que le calcul des fréquences est un processus dynamique et chaque fois que l'Objet Secret change, le calcul est appliqué, et par conséquent, l'attribution des codes pour les lettres les plus occurrent est changée en conséquence.

Cette méthode a présenté la sécurité, la capacité et la robustesse, les trois aspects nécessaires de stéganographie qui la rend utile dans l'échange de données cachées par les documents de texte et l'établissement d'une communication secrète. Cette technique de stéganographie est

utile aussi pour d'autres langues ayant des textes semblables à l'arabe tels que les scripts persan et Urdu, les langues officielles de l'Iran et du Pakistan, respectivement. Ces caractéristiques promettent que cette méthode originale du stéganographie du texte arabe en utilisant des extensions de lettre, est attrayante pour la sécurité d'information.

2.4 Les Caractères PSEUDO-SPACE et PSEUDO-CONNECTION (2008)

En persan et arabe, quelques lettres sont connectées ensemble en un mot, mais d'autres lettres ne peuvent pas se joindre ensemble.

Le **zero width joiner** (ZWJ) est un caractère non imprimable qui est une fois placé entre deux caractères qui ne seraient pas connectés, un ZWJ les fait être imprimés sous leurs formes connectées. Il est connu aussi comme **pseudo-connexion**. L'Unicode du ZWJ est **U+200D**.

Mais il y a aussi un caractère non imprimable en persan et arabe qui empêche les lettres persanes et arabes de se joindre sans ajouter un espace entre les deux et garde les mots plus près ensemble. Ce caractère est appelé le **zero-width nonjoiner** (**ZWNJ**). Il est connu aussi comme **pseudo-space** ou **non-breakable space** (**NBSP**). Son code est **U+200C** dans la notation d'Unicode hexadécimale.

Cette méthode est basée sur les caractères : le zero-width nonjoiner (ZWNJ) et Le zero width joiner (ZWJ).

Dans cette méthode nous cachons un bit dans chaque lettre. Donc pour cacher des données avec cette méthode, d'abord nous regardons si la lettre d'un mot est connectée à la lettre suivante ou non. Si elle est connectée à la lettre suivante, nous insérons le caractère ZWJ entre deux lettres pour cacher le bit 1 et n'ajoutons rien pour cacher le bit 0. Puisque les lettres sont connectées ensemble, le fait d'ajouter ZWJ pour connecter les lettres ensemble n'a aucun effet sur l'apparence du texte.

Mais si la lettre n'est pas connectée à la lettre suivante, nous insérons le caractère ZWNJ entre deux lettres pour cacher le bit 1 et n'ajoutons rien pour cacher le bit 0. Aussi dans ce cas-là l'apparence du mot n'est pas changé; parce que les lettres ne sont pas connectées ensemble et l'ajout de ZWNJ pour séparer les lettres l'un de l'autre n'a pas d'effets sur l'apparence du mot. Pour cacher des données dans la dernière lettre d'un mot, nous insérons toujours le caractère ZWNJ après cela pour cacher le bit 1 et n'ajoutons rien pour cacher le bit 0.

En persan, en plus de l'espace qui est fourni entre les mots, dans quelques mots tel que " می باشد ", il y a un petit espace entre les deux parties du même mot, c'est-à-dire "می" et "باشد". Pour insérer ce type d'espace, les caractères ZWNJ sont insérés entre les deux lettres au lieu de l'espace normal. Si nous avons un mot de ce type dans notre texte, nous insérons deux caractères ZWNJ plus entre deux lettres pour cacher le bit 1 et insérons un caractère ZWNJ plus

entre deux lettres pour cacher le bit 0. Par conséquent les données cachées peuvent être détectées correctement dans la phase d'extraction.

Pour extraire l'information à partir du texte ayant caché l'information (le stégo-texte), nous étudions respectivement les lettres des mots de texte. Si après la lettre un ou trois ZWNJ ou un caractère ZWJ, cela signifie que le bit 1 est caché dans ce mot. Mais si après la lettre aucun (ZWNJ et ZWJ) ou il y a deux ZWNJ, cela signifie que le bit 0 est caché dans cette lettre. En mettant tous les bits de 0 et de 1 à côté l'un de l'autre nous pouvons extraire l'information cachée du texte.

Chapitre 3

Implémentation de quelques algorithmes

1. Introduction

Apres avoir cité et expliqué les travaux faits sur la dissimulation de données dans le texte arabe dans le chapitre précédant, on voudrait maintenant implémenter et programmer quelques algorithmes concernant ces travaux ensuite on va faire une comparaison entre eux dans le chapitre suivant et finalement on termine avec une conclusion.

Pour cela on a choisit d'expliquer trois algorithmes correspondant aux méthodes ou approches Notre choix est de prendre une méthode de chaque type d'utilisation dans la stéganographie:

L'utilisation de signes diacritiques: Le signe diacritique FATHA est utilisé pour cacher le bit "1" et les autres signes pour cacher le bit "0".

L'utilisation de caractère d'extension (Kashida "_"): Insertion d'extension " Kashida" après les lettres pointues pour cacher le bit "1" et après les lettres non pointues pour cacher le bit "0".

L'utilisation de Caractères PSEUDO-SPACE (**ZWNJ**) et PSEUDO-CONNECTION (**ZWJ**).

2. L'environnement de programmation

On a choisi de programmer avec le compilateur G++ sous LUNIX. Au départ on a essayé de programmer d'application graphique mais on a rencontré des erreurs compliquées, et comme le g++ muni par un débuggeur difficile à dérouler Pas à Pas, on s'est contenté de programmer en utilisant line de commande.

Pour le compiler notre programme sous linux on fait :

g++ <nom>.cpp -o <nom d'exécutable>

Et pour exécuter le programme:

./<nom d'exécutable>

3. L'implémentation des algorithmes pour chaque méthode

Pour implémenter les algorithmes choisis pour la stéganographie, on a choisi d'utiliser le texte pour représenter le message secret, car il ne prend pas beaucoup d'espaces dans les supports de stockage par rapport a un son ou une vidéo ou une image, et peut contenir beaucoup d'informations, Ainsi le texte nous aide à manipuler les chaines de caractères et manipuler les fichiers textes facilement.

Premièrement on va expliquer ces points:

L'algorithme de dissimulation a pour entrée deux textes

- **Le texte de couverture** : On utilise un texte diacritisé pour la méthode de diacritiques ou un texte non obligatoirement diacritisé pour les deux autres méthodes, Ce texte va cacher notre message secret.

- **Le texte secret** : c'est le message secret qu'on veut cacher.

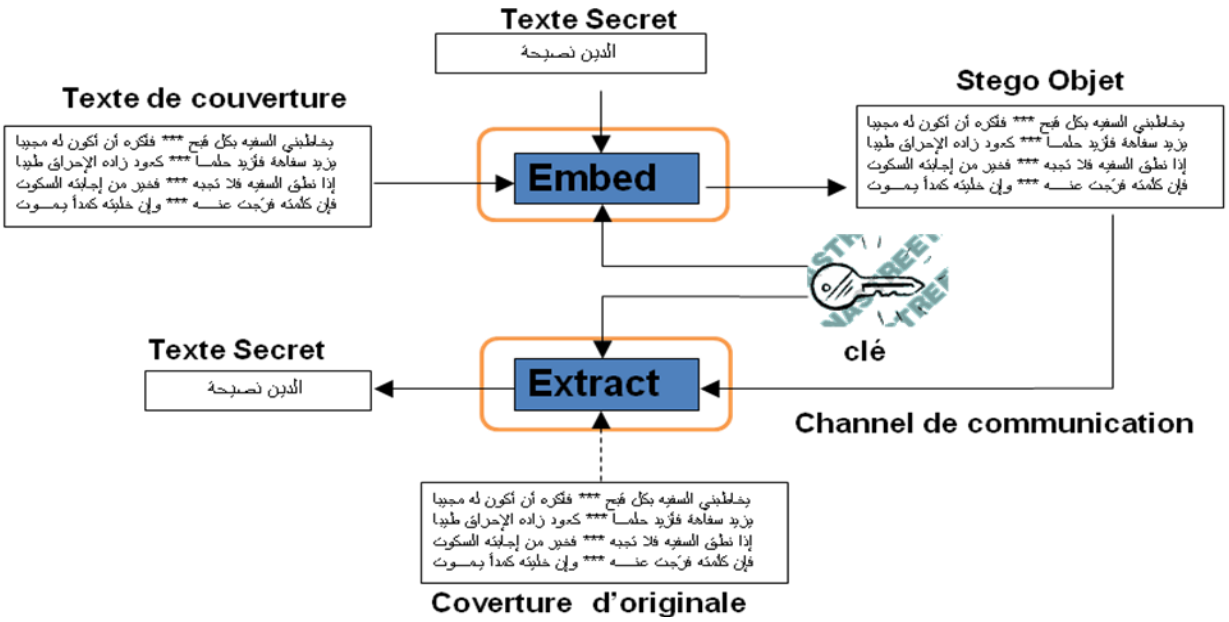
En sortie on obtient un texte :

- **Le Stego Texte** : c'est le texte qui cache notre message secret, ce n'est qu'un texte de couverture avec quelques modification sur le Corps du texte par l'algorithme, ces modifications indiquent les positions des bits secrets qui aident à extraire le message secret.

L'algorithme d'extraction a pour entrée le **Stego-Text** et en sortie le **Texte Secret extrait**, comme montré dans la figure 3.1.

La structure de l'algorithme de stéganographie est commune pour toutes les trois méthodes, elle comporte deux fonctions principales qui varient avec le changement de la méthode:

- **Dissimulation** : La dissimulation du message secret dans le Cover-Text.
- **Extraction** : L'extraction du message à partir du Stego-Text.



. ***Figure 3.1 –*** *Principe d'un système dissimulation d'information.*

3.1. L'algorithme de la technique de signes diacritiques

Voici la fonction de dissimulation pour cette technique en général avec des commentaires :

Début

Initialiser la Position du 1er Caractère du Cover-Text a 0;

Initialiser la Position du 1er Caractère du texte secret a 0;
Initialiser le nombre de bits cachés a 0;
Initialiser la chaine Stego-Text ;

i=0 ; ***/* compteur de bits */***

TantQue *(Caractère Courant* ***(****Cover-Text****)*** *≠ Caractère Fin de chaine)* ***Faire***
Si *(Le Caractère courant n'est pas un signe diacritique)* ***Alors***
Concaténer *Le Caractère courant avec la chaine Stego-Text ;*
SiNon
Extraire le i ème bit du Caractère courant du texte Secret;
Si *((bit****==****1* ***ET*** *Caractère courant****==Fatha****)* ***Ou***
*(bit****==****0* ***ET*** *Caractère courant****≠Fatha****))* ***Alors***
Concaténer Le Caractère courant avec la chaine Stego-Text ;
Incrémenter le nombre de bits cachés ;
/*On calcule le nombre de bits cachés dans le cover-text*/
Si *(i==8)* ***Alors /*la fin d'extraction des bits du caractère courant du texte secret*/***
Passer au caractère suivant du texte secret ;
i=0 ;
Si *(la fin du texte secret est atteint (Avec la prise en compte le caractère de fin de chaine)* ***Alors***
Initialiser la position du caractère courant du texte secret au début ;
/* Recommencer la dissimulation du secret texte jusqu'à la fin du Cover-Text.
On inclut le caractère de fin de chaine dans le cover-text pour pouvoir Extraire exactement le secret texte dans la phase d'extraction. */
FinSi
FinSi
FinSi
FinSi
Passer au Caractère Suivant du Cover-Text ;
FinTantQue
Fin.

Dans la phase d'extraction on va récupérer notre message secret, La fonction d'extraction cherche sur les signes diacritiques dans le Stego-Text pour extraire les bits et construire le message secret

Et voici le corps de la fonction :

Dedut
Initialiser la Position du 1er Caractère du Stego-Text a 0;
Initialiser le caractère extrait a 0 ;
Initialiser le nombre de décalages de bits a 0 ;
TantQue *(Caractère Courant* ***(****Stego-Text****)*** *≠ Caractère Fin de chaine)* ***Faire***
Si *(Le Caractère courant est un signe diacritique)* ***Alors***
Décaler le caractère extrait un bit à gauche ;
Si *(Caractère courant****==Fatha****)* ***Alors***

Caractère extrait= Caractère extrait ***XOR*** *1 ;*
FinSi
Incrémenter le nombre de décalages de bits;
Si *(le nombre de décalages de bits atteint 8)* ***Alors***
Si *(le Caractère extrait est un caractère de fin de chaine)* ***Alors***
Sortir de la boucle ;
/* Car on a déjà caché le caractère de fin de chaine dans le Cover-Text et que notre message secret est complet */
FinSi
Ajouter le Caractère extrait au secret texte extrait ;
Réinitialiser le nombre de décalages de bits a 0 ;
Réinitialiser le caractère extrait a 0 ; ***/*Pour extraire le caractère suivant */***
FinSi
FinSi
FinSi
Passer au Caractère Suivant du Stego-Text ;
FinTantQue
Fin.

3.2. L'algorithme de la technique de Kashida (Apres les lettres)

Voici la fonction de dissimulation :

Dedut
Initialiser la Position du 1er Caractère du Cover-Text a 0;
Initialiser la Position du 1er Caractère du texte secret a 0;
Initialiser la chaine Stego-Text ;
Initialiser le nombre de bits cachés a 0;
i=0 ; ***/* compteur de bits */***
TantQue *(Caractère Courant* ***(****Cover-Text****) ≠*** *Caractère Fin de chaine)* ***Faire***
Si *(Le Caractère courant est un caractère Kashida)* ***Alors***
Si*((Le Caractère courant n'est pas une lettre extensible avec points)* ***ET***
*(Le Caractère courant n'est pas une lettre extensible sans points))****Alors***
Concaténer *Le Caractère courant avec la chaine Stego-Text ;*
SiNon
Extraire le i ème bit du Caractère courant du texte Secret;
Si *((*(bit==1 ***ET*** *Le Caractère courant est une lettre extensible avec points)* ***Ou***
(bit==0 ***ET*** *Le Caractère courant est une lettre extensible sans points))* ***ET***
(*Le caractère suivant est une lettre arabe****)*** *)* ***Alors***
Concaténer *Le Caractère courant avec la chaine Stego-Text ;*
Concaténer *Le Caractère* ***Kashida*** *avec la chaine Stego-Text ;*
/*On calcule le nombre de bits cachés dans le cover-text*/
Incrémenter le nombre de bits cachés ;
Si *(i==8)* ***Alors /*la fin d'extraction des bits du caractère courant du texte secret*/***
Passer au caractère suivant du texte secret ;
i=0 ;

Si *(la fin du texte secret est atteint (Avec la prise en compte le caractère de fin de chaine)* ***Alors***
Initialiser la position du caractère courant du texte secret au début ;
/* Recommencer la dissimulation du secret texte jusqu'à la fin du Cover-Text.
On inclut le caractère de fin de chaine dans le cover-text pour pouvoir Extraire exactement le secret texte dans la phase d'extraction. */
FinSi
FinSi
SiNon
Concaténer *Le Caractère courant avec la chaine Stego-Text ;*
FinSi
FinSi
FinSi
Passer au Caractère Suivant du Cover-Text ;
FinTantQue
Fin.

voici la fonction d'extraction :

Dedut
Initialiser la Position du 1er Caractère du Stego-Text a 0;
Initialiser le caractère extrait a 0 ;
Initialiser le nombre de décalages de bits a 0 ;
TantQue *(Caractère Courant* ***(****Stego-Text****) ≠*** *Caractère Fin de chaine)* ***Faire***
Si *(Le Caractère courant est un caractère* ***Kashida****)* ***Alors***
Décaler le caractère extrait un bit a gauche ;
Si *(le caractère précédant de cover-text est un lettre extensible avec points)* ***Alors***
Caractère extrait= Caractère extrait ***XOR*** *1 ;*
FinSi
Incrémenter le nombre de décalages de bits;
Si *(le nombre de décalages de bits atteint 8)* ***Alors***
/*car la code de caractère sur 8 bits*/
Si *(le Caractère extrait est un caractère de fin de chaine)* ***Alors***
Sortir de la boucle ;
/* Car on a déjà caché le caractère de fin de chaine dans le Cover-Text et que notre message secret est complet */
FinSi
Ajouter le Caractère extrait au secret texte extrait ;
Réinitialiser le nombre de décalages de bits a 0 ;
Réinitialiser le caractère extrait a 0 ; ***/*Pour extraire le caractère suivant */***
FinSi
FinSi
Passer au Caractère Suivant du Stego-Text ;
FinTantQue
Fin.

3.3. L'algorithme de La technique de ZNJ_ZWNJ

Voici la fonction de dissimulation :

Dedut

Initialiser la Position du 1er Caractère du Cover-Text a 0;

Initialiser la Position du 1er Caractère du texte secret a 0;

Initialiser la chaine Stego-Text ;

Initialiser le nombre de bits cachés a 0;

i=0 ; ***/* compteur de bits */***
TantQue *(Caractère Courant* ***(****Cover-Text****) ≠*** *Caractère Fin de chaine)* ***Faire***
Si *(Le Caractère courant ou le caractère prochaine n'est pas un lettre arabe)* ***Alors***
Concaténer *Le Caractère courant avec la chaine Stego-Text ;*
SiNon
Concaténer *Le Caractère courant avec la chaine Stego-Text ;*
Extraire le i ème bit du Caractère courant du texte Secret;
Si *((bit==1* ***ET*** *lettre prochaine connectable)* ***Alors***
Concaténer Le Caractère ZWJ avec la chaine Stego-Text ;
FinSi
Si *((bit==1* ***ET*** *lettre prochaine non connectable)* ***Alors***
Concaténer Le Caractère ZWNJ avec la chaine Stego-Text ;
FinSi
/*On calcule le nombre de bits cachés dans le cover-text*/
Incrémenter le nombre de bits cachés ;
Si *(i==8)* ***Alors /*la fin d'extraction des bits du caractère courant du texte secret*/***
Passer au caractère suivant du texte secret ;
i=0 ;
Si *(la fin du secret texte est atteint (Avec la prise en compte le caractère de fin de chaine)* ***Alors***
Initialiser la position du premier caractère du chaine secret texte au début ;
/* Recommencer la dissimulation du secret texte jusqu'à la fin du Cover-Text.
On inclut le caractère de fin de chaine dans le cover-text pour pouvoir Extraire exactement le secret texte dans la phase d'extraction. */
FinSi
FinSi
FinSi
Passer au Caractère Suivant du Cover-Text ;
FinTantQue
Fin.

Et voici l'algorithme d'extraction :

Dedut
Initialiser la Position du 1er Caractère du Stego-Text a 0;

Initialiser le caractère extrait a 0 ;
Initialiser le nombre de décalages de bits a 0 ;
TantQue** (Caractère Courant **(**Stego-Text**) ≠** Caractère Fin de chaine) **Faire
Si** ((Le Caractère courant est un lettre arabe **ET
*Le prochain Caractère du Cover-Text n'est pas un lettre arabe**) OU***
*aussi Le Caractère courant n'est pas un lettre arabe) **Alors***
Décaler le caractère extrait un bit a gauche ;
Si** (Caractère prochain ==**ZWJ ou** Caractère prochain ==**ZWNJ) Alors
*Caractère extrait= Caractère extrait **XOR** 1 ;*
FinSi
Incrémenter le nombre de décalages de bits;
Si** (le nombre de décalages de bits atteint 8) **Alors
Concaténer le caractere courant avec le secert text

/* Initialiser le compteur de bits dans un car
initialiser le nouveau caractere */
FinSi
Incrémenter le nombre de décalages de bits;
Réinitialiser le nombre de décalages de bits a 0 ;
Réinitialiser le caractère extrait a 0 ;
/* Pour extraire le caractère suivant */
FinSi
Passer au Caractère Suivant du Stego-Text ;
FinTantQue
Fin.

4. conclusion

Malgré la diversité des approches de la stéganographie sur les textes, chaqu'une de ces approches a des points forts et des points faibles , mais toutes ont des points communs tels que l' impuissance et la difficulté d'être implantées ; pour que le stego-medium soit plus résistant et sécurisé et être imperceptibles, alors qu'à l'aide d'autres techniques comme l'encryptage de texte secret ou on ajoute une clé privilège on améliore la sécurité et la confidentialité de la stéganographie. ce qui a permis aux chercheurs d'élargir la stéganographie dans des domaines déférents (transmission des fichiers ; copyright…). En plus les algorithmes de la stéganographie se développent et deviennent plus compliqués.

Chapitre 4

Comparaison entre les algorithmes implémentés

1. Introduction

Après l'implémentation des algorithmes on a besoin de faire une comparaison entre eux pour connaitre la méthode la plus efficace.

Dans ce chapitre on va faire une comparaison entre les algorithmes choisis en respectant les conditions requises notés dans le premier chapitre : **l'imperceptibilité, La capacité** et **La robustesse**.

2. Comparaison entre les algorithmes implémentés

2.1 Comparaison du point de vue capacité

Pour faire la comparaison de point de vue capacité on va estimer la quantité de données qu'on peut cacher dans le texte, Pour ça on a besoin de compter le Nombre de bits significatifs dissimulés dans le texte utilisé entièrement qu'on va nommer **α**, et le Nombre de bits total dans le texte qu'on va nommer **T**, et ce comptage de bits est fait durant l'exécution de l'algorithme.

Donc la capacité de dissimulation en **bit/kilobyte** est obtenue par cette relation:

Capacité de dissimulation = 1024 x α / (T/8)

On a fait des statistiques sur quelques fichiers texte de tailles 1.046, 0.969, 0.17 et 1.127 kilobyte.

On a appliqué cette relation sur les résultats de chaque fichier texte pour chaque algorithme parmi les algorithmes implémentés et on a obtenu le tableau suivant :

Approche	Taille du Cover-Texte (kilobyte)	Capacité (bit/kilobyte)	Capacité (%)	Capacité moyenne (bit/kilobyte)	Capacité moyenne (%)
Diacritiques	1.046	280	3.88	300	3.56
	0.969	300	3.77		
	0.17	298	3.1		
	1.127	323	3.49		
Kashida (Apres les lettres)	1.046	211	2.58	224	2.40
	0.969	220	2.77		
	0.17	234	2.06		
	1.127	230	2.20		
ZWJ et ZWNJ	1.046	667	8.14	794	9.03
	0.969	735	9.25		
	0.17	861	8.92		
	1.127	912	9.81		

Tableau 4.1. *Capacité de dissimulation pour les trois approches.*

Premièrement, l'approche de **diacritiques** a besoin d'un texte diacritisé pour faire le teste sur lui, par contre les deux autres approche n'ont pas besoin d'un texte diacritisé puisque les signes diacritiques ne sont pas utilisés.

Comme il est montré dans le tableau 4.1**,** la méthode **Zwj et Zwnj** a une très haute capacité de dissimulation par rapport aux deux autres méthodes ; elle permet de cacher une grande capacité de texte secret et atteint jusqu'à 1 sur 10 fois de capacité de texte de couverture. Par contre les autres ne peuvent cacher qu'une faible capacité de texte secrète, qui ne dépasse pas 1 sur 30 fois de capacité de texte de couverture.

En deuxième position vient l'approche de **signes de diacritiques,** mais en général, elle est n'exploite pas au max. le nombre des diactriques (Fatha) flottants sur le Stego texte) même quand le nombre de Fatha est important, car on cache un bit d'information dans chaque lettre arabe tandis que pour les deux autres méthodes on cache les bits dans quelques caractères en fonction du cas d'utilisation de ces caractères. la méthode de **signes Diacritiques** cache les bits secrets dans les Diacritiques seulement et avec des conditions.

La méthode **Kashida (Après les lettres)** cache les bits secrets dans les lettres extensibles seulement mais avec des conditions aussi.

Alors le risque d'avoir un grand nombre de bits secrets dans l'Objet Secret est très possible pour les deux méthodes **Kashida** et **Diacritiques,** ce qui représente un inconvénient parce que la taille du média de couverture (Cover-Medium) ne satisfait pas la dissimulation de message secrète.

Si on compare la méthode **Kashida (Après les lettres)** avec la méthode de **signes Diacritiques** on peut en juger que la deuxième est un peu mieux du point de vue capacité.

2.2 La comparaison de coté imperceptibilité

Avec la méthode (**Pseudo-Space** et **Pseudo-Connection**) les caractères **ZWJ** et **ZWNJ** sont des caractères non-imprimables, donc cette méthode n'apporte aucun changement sur l'apparence du texte original, et a une transparence perceptuelle parfaite.

Ainsi, même si le lecteur a le texte original, Il est impossible pour lui de réaliser la dissimulation des données en observant simplement l'apparence du texte.

Cependant les textes originaux ne sont pas disponibles pour les observateurs habituels. Donc, le but principal, qui est l'impossibilité de détection de la présence de données, a été accompli Ce qui garantie la sécurité pour cette méthode.Avec la méthode **Kashida (Apres les lettres)** la sécurité n'est pas surement assuré, car il est probable qu'une personne remarque l'existence d'un message dans le Stego Texte.

Et pour ajouter plus de sécurité à la méthode **Kashida (Après les lettres)**, l'option d'ajouter des extensions avant et après les lettres peuvent être utilisées dans le même document, mais dans différents paragraphes ou lignes. Par exemple, les lignes ou les paragraphes pairs utilisent le watermarking d'extensions après les lettres. et les lignes ou les paragraphes impairs utilisent les extensions avant ou vis versa.

Et finalement la méthode de **Diacritiques** peut élever des doutes car, de nos jours, il est rare d'envoyer un texte diacritisé, à moins que les média de couverture utilisés ne soient pas des documents religieux ou politiques par exemple, donc cette méthode est faible du côté imperceptibilité.

2.3 La comparaison de coté robustesse

La méthode de ZWJ et ZWNJ n'est pas dépendante du format du texte, et on peut sauvegarder le Stego Texte dans plusieurs formats comme les pages HTML, documents Word ou même un format de texte simple, parce que le Stego Texte avec un texte Unicode ne changera pas pendant la copie et la colle entre les programmes informatiques, les données cachées dans les textes restent intactes pendant ces opérations.

La méthode de **signes Diacritiques** est aussi robuste car elle peut résister à l'imprimerie, les techniques d'OCR, le changement de fonte, en retapant aussi longtemps que le texte peut contenir l'arabe.

Mais l'algorithme **Kashida (Après les lettres)** n'est pas robuste car généralement, il n'est pas capable de garantir la récupération de texte secret exacte, surtout quand on trouve un kashida entre deux lettres pointues tel que on cache '0' après la première lettre ou '1' avant le deuxième. Pour résoudre ce problème on a donné une deuxième version (kashida before adding/ avant les lettres).

3. Conclusion

Les méthodes qu'on a implémentées assurent avec des proportions différentes, la sécurité, la capacité et la robustesse ce sont les trois aspects nécessaires de dissimulation de données et le watermarking.

La méthode la plus récente qui consiste à utiliser les caractères **ZWJ** et **ZWNJ,** a montré clairement son efficacité de dissimulation avec toutes les conditions requises de stéganographie par rapport aux autres méthodes malgré des optimisations et des améliorations faites sur quelques algorithmes.

Conclusion Générale

Dans cette étude on a compris le sens informatique de dissimulation d'une information dans une autre information, pour ce la on a implémenté quelques méthodes qui exploitent les avantages de la langue arabe comme l'utilisation des signes diacritiques ou comme l'utilisation des extensions entre les lettres, ces méthodes cache un texte quelconque dans le texte arabe, chaque méthode a sa proportion de capacité de dissimulation, son niveau de robustesse et sa caractéristique d'imperceptibilité.

Si on veut cacher un texte On peut optimiser plus ces méthodes au niveau de capacité de dissimulation et au niveau d'imperceptibilité, par la diminution de la taille des codes des caractères par un bit ou plus, et sa se fait par la création des nouveaux codes.

Pour éviter d'avoir une information incomplète ou perceptible par l'œil caché dans le texte de couverture on doit choisir un texte a capacité grande qui peut accueillir notre information.

Ces méthodes respectent les conditions requises à savoir, la sécurité, la capacité et la robustesse, ces trois aspects nécessaires à la stéganographie et qui la rendent utile dans l'échange caché d'informations par les documents de texte et l'établissement de la communication secrète, ainsi que pour la prévention de la reproduction et de la distribution des textes illégales.

La langue arabe reste la langue officielle pour les pays arabes, et Par conséquent, une large gamme d'utilisateurs peuvent utiliser ces méthodes, En plus les langues Pashto (la langue officielle de L'Afghanistan) et Urdu (la langue officielle de Pakistan) sont semblables à l'arabe et au persan, nous pouvons appliquez aussi ces méthodes à ces deux langues.

On remarque aussi qu'avec le temps il ya eu une apparition de méthodes plus en plus robustes et efficaces ce qui permet de garantir de plus en plus la sécurité des informations échangées.

Références

Chapitre 1:

[1] A. ALI-PACHA; N. HADJ-SAID; A. BELGORAF, A. M'HAMED, ***Stéganographie : Sécurité par Dissimulation.***

[2] URL: http://fr.wikipedia.org/wiki/Stéganographie

[3] R.Dumont, ***Chapitre18 : La Stéganographie.***

Chapitre 2:

[4] M. Hassan Shirali-Shahreza, Mohammad Shirali-Shahreza, "***A New Approach to Persian/Arabic Text Steganography***", 5éme IEEE/ACIS International Conference on Computer and Information Science (ICIS 2006), pp. 310-315, Honolulu, USA, 10-12 juillet 2006.

[5] W. Bender, D. Gruhl, N. Morimoto, "***Techniques for data hiding***", IBM Systems Journal, Vol 35, 1996.

[6] M.H. Shirali-Shahreza ET M. Shirali-Shahreza, "***Steganography in Persian and Arabic Unicode texts using pseudo-space and pseudo connection characters***", Journal of Theoretical and Applied Information Technology: **Jatit,** 2008.

[7] M. Shirali-Shahreza, "***A New Method for Steganography in HTML Files*** ", 2005.

[8] M. A. Aabed, S. M. Awaideh, A. M. Elshafei, and A. A. Gutub, "***Arabic Diacritics Based Steganography***", IEEE International Conference on Signal Processing and Communications (ICSPC 2007), pp. 756-759, Dubai, UAE, 24- 27 Novembre 2007.

[9] Katzenbeisser (S.) et Petitcolas, "***Information hiding techniques for steganography and digital watermarking***", 1999.

[10] Adnan Gutub, Lahouari Ghouti, Alaaeldin Amin, Talal Alkharobi, and Mohammad K. Ibrahim, "***Utilizing Extension Character 'Kashida' With Pointed Letters For Arabic Text Digital Watermarking***", International Conference on Security and Cryptography - SECRYPT, Barcelone, Espagne, juillet 28 - 31, 2007.