INSTITUTO TECNOLÓGICO DE BUENOS AIRES

22.05 ANÁLISIS DE SEÑALES Y SISTEMAS DIGITALES

TRABAJO PRÁCTICO 4

Steganografía sobre Imágen

Alumnos:
Nahuel AGUILAR 54607
J. Agustín BARRACHINA 53790
Gonzalo CASTELLI 55055
Augusto VIOTTI BOZZINI 55138

Profesores:
Daniel JACOBY
Carlos SELMO
Juan Ignacio CARRANO
Andres BERSIER
Julian Andrés TACHELLA

January 30, 2017

CONTENTS

I. INTRODUCCIÓN

Steganografía es el acto de ocultar un documento, mensaje, imagen o algún otro archivo dentro de otro archivo.

Su orígen viene del latín que combina la palabra "steganos" que significa oculto o secreto con la palabra "graphein" que significa escritura. Dicha palabrea fue empleada por primera vez en 1499 por Johannes Tritheminus en su "Steganographia, a treatise on cryptography and steganography".

Sus métodos y formas desde entonces son innumerables, y se extieneden incluso fuera del dominio digital. En este trabajo particularmente se tratará particularmente sobre steganografía sobre Imágenes únicamente. Pero antes de proseguir con éste tema es importante destacar claramente las diferencias de dos términos generalmente confundidos: Steganografía y Criptografía. Así también como la diferencia que poseen con lo denominado WaterMarking.

Si bien es cierto que tanto el objetivo de la steganografía como de la criptografía es ocultar mensajes, su diferencia radica en la forma en que lo hacen. La criptografía se encarga de impedir que un tercero acceda a la información protegida por medios de algoritmos de seguridad que impiden a esta persona abrir el archivo o acceder al mismo, es decir, que ésta persona sabe de la existencia del archivo pero falla en acceder al mismo debido a la protección impuesta. Ésto posee el defecto de atraer demaciada atención sobre el archivo protegido, si despierta el interés en el tercero, el mismo puede terminar quebrando el código y descubriendo el mensaje oculto. Incluso, en algunos países, el uso de criptografía puede ser ilegal. Por otra parte, la steganografía se encarga de ocultar el hecho de que el documento a proteger exista. Su objetivo es disfrazar el mismo dentro de otro archivo que parezca innocente de forma que cuando un tercero vea ese archivo no sepa que hay un mensaje oculto dentro. Muchas veces, para mayor seguridad suelen combinarse ambos métodos de forma que si por algún motivo quien lea el mensaje logra descubrir que existe el mensaje oculto, no pueda desencriptarlo.

Si bien la línea que diferencia el watermarking con la steganografía puede ser muy delgada bajo

ciertos aspectos. Principalmente varían en el objetivo que emplean, el objetivo del watermarking es generalmente una firma de autor que ayuda a combatir la piratería y proporciona seguridad contra el copyright. A veces de hecho, ésta marca de agua es claramente visible lo cual contradice la idea básica de la steganografía.

II. Steganografía en Imágenes

Como ya se mencionó anteriormente, este trabajo se centrará en steganografía en imágenes. La mísma se puede dividir en dos clasificaciones según su domínio (según [1]). Está las que se basan en el dominio de las frecuencias (Transform Domain) y las que se basan en el simple dominio de la imagen (Spatial Domain). Éste último aplica inserción de bits y manipulación del ruido a la imagen portadora. Por el otro lado, las que trabajan con las frecuencias utilizan la transformada de fourier para esconder la información en su espectro de la frecuencia. Dentro de esta clasificación se encuentran innumerables métodos. Entre los más conocidos se encuentran los siguientes (La imágen se basó en el trabajo [2]).

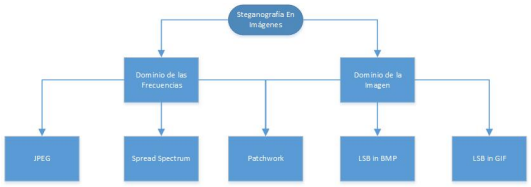

Fig. 1. Distintos Métodos de Steganografía

A. Dominio de la Imagen (Spatial Domain Technique)

1) Least Significant Bit: El método del LSB por sus cifras en inglés es quizás el más conocido y sencillo de tódos. Pero aún así muy efectivo [3]. El método consiste en cambiar cada byte que repserenta un color de un píxel en la imágen (RGB) de una forma muy sutíl para incorporar el archivo a esconder. Se basa en la idea que el ojo humano no es capáz de detectar, por ejemplo el cambio de un solo píxel de una intensidad de 255 a 254. Además, en promedio solo la mitad de los bits deberían cambiarse lo cual reduce aún más el cambio que se efectúa en la imágen portadora.

Se podría proteger éste método con una llave secreta que indique cuales de todos los bytes fueron modificados, siendo que solo se hubieran cambiado algunos de ellos de forma aleatoria y no todos por completo.

El método es muy sensible a la compresión, por eso se necesita formatos de archivos que no comptiman la imágen, motivo por el cual se suele implementar éste método en archivos de formato Bit Map (BMP)

B. Dominio de las Frecuencias (Transform Domain Technique)

Antes de ahondar sobre éste método es necesario conocer el funcionamiento básico de los archivos JPEG. El JPEG es muy importante ya que es el formato de imágen más utilizado sobre todo en internet por su gran capacidad de compresión. Por su popularidad será el formato de archivo en el cual se basará el trabajo principalmente.

El JPEG pasa del sistema RGB al YUV. La componente Y representa la luminocidad mientras que U y V representan la crominancia. El ojo humano es más sensible al cambio en la luminancia de un pixel que cambios en el color. Éste hecho es explotado por la compresión JPEG bajando los datos del color para reducir el peso de la imágen.

El próximo paso consta de transformar la imagen al dominio de las frecuencias utilizando Discrete Cosine Transform (DCT). Se suele agrupar los péxeles en grupos de 8x8 y transforma cada bloque en 64 coeficientes de la DCT de forma que modificando un solo coeficiente afectará a todos los píxeles del bloque.

Luego se aplica otro dato más sobre el ojo humano. El mismo no es bueno viendo grandes cambios de luminocidad en poca área, es decir, es una especie de pasa bajos cuando se trata de brillo. Por lo que entonces se divide los coeficientes por unos valores que reducen enormemente las altas frecuencias sin afectar la apariencia de la imágen. Estos valores por los cuales se dividen los coeficientes se guardan en una tabla que luego se lee al descomprimir para mostrar la imágen. Los

coeficientes son luego tratados con el código de Huffman para mayor reducción del tamaño.

El mayor defecto de los tipos de archivo JPEG ocurre en texto, ya que el mismo suele presentar principalmente altas frecuencias.

1) JPEG Steganografía: Originalmente se creía que no iba a ser posible realizar steganografía en archivos de formato JPEG por su característica "lossy" (pérdida de información). Sin embargo, es la etapa de la transformación mediante la DCT y redondeo a enteros de los coeficientes tras dividirlos lo que le da al JPEG ésta característica. Es posible agregar información luego de ésta etápa y antes de la codificación por Huffman para realizar la steganografía sobre archivos JPEG.

III. Características Importantes

Según describe [2], un buen algoritmo de steganografía debe poseer las siguientes características:

1) Invisibilidad: Debe ser indetectable por el ojo humano. El mísmo no puede ser capáz de detectar que algo en la imágen no está bien.
2) Capacidad de Almacenamiento: Cantidad de información que puede esconder bajo cierto tamaño de la imagen portadora.
3) Resistencia contra el Stegoanalisis: El stegonálisis es la práctica de detectar si cierta imágen posee o no un archivo oculto en la misma. Sobre éste tema se abordará a en la sección IV.
4) Robustez contra la manipulación: Consta de que tan robusto es el algoritmo contra manipulaciones como girar una imágen o recortes.
5) Independencia del tipo de formato: Si solo se utiliza un tipo de archivo, puede resultar sospechoso para un tercero que siempre utilicen el mismo formato.
6) Archivos sospechosos: Incluye cualquier característica que pueda resultar sospechosa como por ejemplo un tamaño excesivamente grande para una imágen.

A continuación se presenta un cuadro que se obtiene de [2] que compara diversos métodos de Steganografía:

Fig. 2. Tabla comparativa entre Distintos métodos de Steganografía

IV. Stegoanálisis

El stegonálisis es el arte de detectar la existencia de archivos ocultos dentro de imágenes. Como el formato de interés para este trabajo es el JPEG se investigó principalmente sobre éste formato.

Un nuevo método de stegoanálisis basado en una representación escasa es tratado en [5]. En [6], Qingzhong Liu propuso un esquema para el stegonálisis de imágenes JPEG que mejoró la detección de sistemas basados en JPHS, CryptoBola, F5, Steghide, etc. En [7], N. V. S. Sree Rathna Lakshimi propuso un algoritmo ganador del novel que permitía detectar la diferencia entre un archivo normal de otro modificado. En [8] presentó un método para diferenciar una imagen stego de su imagen portadora basada en la descomposición por bloques de la imágen. Manu Devi & Nidhi Sharma presentaron en [9] cuatro diferentes técnicas para el stegoanálisis. Finalmente, en [10], Han Zong propuso un método basado en la correlación para la detección de un posible archivo oculto mediante la steganografía.

Una extensa comparación entre éstos métodos explicados fue realizada en [4].

V. Software Comercial

Existen diversos softwares encargados tanto de realizar stegonálisis como steganografía [11]. Entre ellas se encuentran:

- Anubis
- OpenStego
- Invisible Secret Tool
- Camouflage
- StegSecret

Particularmente, StegSecret es una herramienta de stegoanálisis que puede servir para testear el código implementado.

VI. Implementación

A. Lenguaje y Librerías

Se decidió implementar el programa en C++ por diversas razones. En primer lugar, los algoritmos

a implementar requieren de múltiples operaciones a nivel de bits, por lo que los operadores binarios nativos de este lenguaje presentan una gran ventaja en la simpleza y velocidad del código para dichas operaciones. La necesidad de obtener un programa rápido y fluido por la gran cantidad de operaciones que serán necesarias, hace de los lenguajes de bajo nivel como C++ una buena elección frente a lenguajes de alto nivel como Python y Matlab. Finalmente, C++ es un lenguaje con permanentes actualizaciones, y con abundante bibliografía y documentación online. Además ya fue ampliamente utilizado por los autores de este trabajo, permitiendo trabajar en una zona de confort.

En cuanto al manejo de imágenes, se optó por utilizar la librería CImg. Esta es una pequeña librería de código abierto con múltiples y deseables propiedades. Permite trabajar con múltiples formatos de imágen, es portable entre diversos sistemas operativos, es simple de usar, es ligera en cuestiones de espacio de almacenamiento, y permite utilizar funcionalidades de librerías externas, como ImageMagick para la compresion y decompresión JPEG.

B. Librería Implementada

Se realizaron una serie de funciones para la codificación y decodificación de distintos tipos de estenografía. También se implementaron funciones para medir el error entre las imágenes antes y después de procesos de estenografía y compresión-descompresión JPG, y el error en la información recuperada a partir de la decodificación de la esteganografía.

C. LSB Steganography

Se implementó una steganografía de LSB simple, el mismo será reutilizado también más adelante para algoritmos JPEG y éso se aprovecha para profundizar en éste método que si bien es sencillo es de los más utilizados para la esteganografía.

Una de las principales ventajas de éste método es la cantidad de información que puede esconderse sobre una imágen. Se logró incluir una imágenes dentro de imágenes para mostrar la efectividad del algoritmo. La siguiente imagen muestra como dentro de una imagen de 126 MB se incluyó una imágen de 23.7 MB la cual contenía a su vez una imagen HD estándar de 1920x1080, y así sucesivamente hasta concluir con un archivo de texto. Las mísmas se pueden ver en 3.

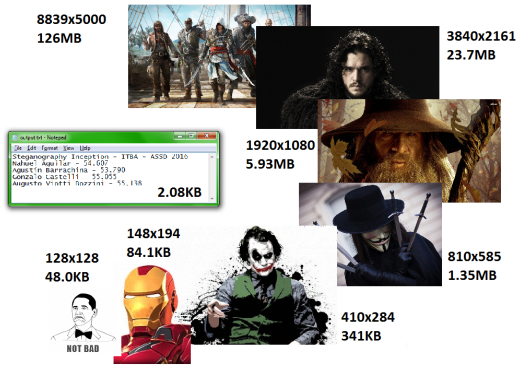

Fig. 3. Esteganografía anidada con 2LSB para imágenes y texto

La realización utilizando un solo LSB resulta imperceptible al ojo humano incluso comparando con la imagen original. Con dos LSB, ya puede ser posible notar alguna diferencia en las muy bajas frecuencias, como planos de color continuos. En la siguiente figura se puede ver como en bajas frecuencias, la zona superior de la imagen, se puede ver un ligero cambio en el color del negro. Por otro lado, en altas frecuencias (zona inferior) no se puede apreciar cambio alguno.

Fig. 4. Esteganografía LSB de 2-bit

D. JPEG Steganography

Para la explicación sobre el algoritmo se suponen conocimientos básicos de la teoría detras de la compresión JPEG, si bien algunos aspectos que se consideren importantes se resaltarán.

Para pasar de BMP o PNG a JPEG se debe utilizar una transformación de RGB (Red Blue Green) a YC_bC_r donde 'Y' corresponde a la coordenada de luma y 'C' corresponde al cromo (diferencia) de azul y rojo respectivamente. Esta simple conversión, en teoría, sin pérdidas, genera error por cuantización ya que se debe redondear a valores enteros entre 1 y 255. Por lo que se debe tener en cuenta que aunque se utilice una compresión sin pérdidas se va a obtener algún mínimo error por el simple hecho de pasar de un sistema de coordenadas RGB a YC_bC_r.

JPEG realiza la conversión DCT (Discret Cosine Transform) sobre un cuadrado de 8 x 8 bytes. Aquí se aprovecha el hecho de que el ojo humano distingue menos las altas frecuencias de las bajas frecuencias y se divide los valores de la DCT por unos coeficientes establecidos en tablas según el grado de compresión. Se obtuvieron numerosas tablas para los distintos factores de compresión utilizando JPEGsnoop (ver 9). El mismo es un programa que te muestra información sobre archivos JPEG. Se obtuvieron tablas, denominadas matrices de cuantización, para los porcentajes de compresión de 50, 60, 70, 75, 80, 95 y 100 %.

La esteganografía se realiza luego sobre los 8 primeros bytes de menores frecuencias ya que son las que poseen coeficientes menores y las que menos información pierden. Además, si la imágen es recortada, resulta más sencillo recuperar el mensaje si se utilizan estos 8 bytes, iterando el algoritmo de decodificación, comenzando desde cada pixel del cuadrado inicial de 8x8 hasta encontrar el mensaje. . El algoritmo consiste ahora en aplicar el anterior método de LSB de 1 bit sobre estos bytes. No se aplica LSB sobre las componentes de altas frecuencias ya que los coeficientes son muy grandes y generan un gran error de cuantización al redondear. Además, hacer eso haría al algoritmo muy sensible al esteganálisis.

Fig. 5. Ejemplo de matríz resultante después de la DCT. Remarcados, los valores de la DCT sobre los cuales se aplica LSB

Se puede realizar una resta de los valores de la imagen original con la imágen luego de utilizar el algoritmo de steganografía por JPEG. A estos errores se los puede amplificar para poder tratar de notar la diferencia entre ellos. Una imágen de muestra se puede ver en 6:

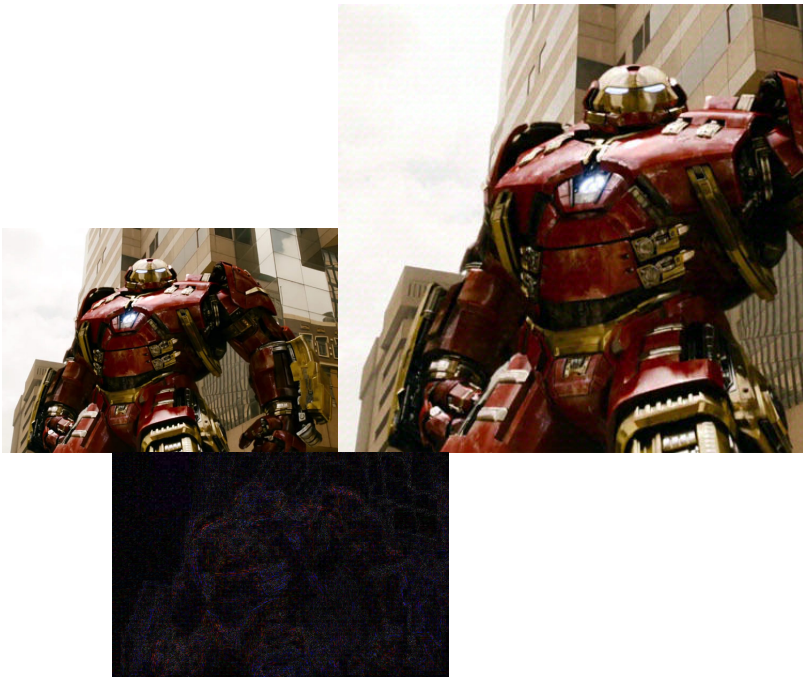

Fig. 6. Imagen original, con esteganografía, y diferencia entre imágenes amplificada x5

En dicha figura se puede observar facilmente como el mayor error se encuentra en las altas frecuencias. Se puede ver claramente la figura del personaje y los contornos de los objetos pero en las partes de bajas frecuencias como el cielo en el fondo el error es mucho menos apreciable.

E. Análisis de resultados

Luego de realizar el algoritmo se probó recomprimiendo la misma imágen numerosas veces para

verificar cual era el error al comprimir cierto numero de veces la imágen. Si se utiliza una tabla de compresión de 80%, la misma no es resistente a compresiones de 75% o menores ya que al utilizar una matríz de compresión con mayores coeficientes de cuantización, los LSB se pierden por redondeo y no se logra recuperar bien el mensaje por más repetición que se utilice. Es decir, el algoritmo solo resiste compresiones de porcentajes de compresión mayores o iguales al grado ustilizado.

Se alcanza un valor de compromiso entre la resistencia a la compresión y la calidad de imagen obtenida.

Para el calculo del error se utilizó la diferencia de Hamming. La misma indica que el número máximo de bits que se pueden tener erroneos está dado por la mitad menos uno de la cantidad de reiteraciones que se coloca en el archivo o imagen de carrier. La probabilidad que se produzcan t errores sobre una cantidad de n bits está dado por:

$$p_{error} = \frac{n!}{t!(n-t)!} \cdot p^t \cdot (1-p)^{n-t} \qquad (1)$$

La probabilidad p se obtiene promediando los errores que se obtuvieron conociendo por supuesto el bit orignal.

Para calcular la probabilidad de error total se debe realizar la sumatoria de todos los t para el cual se obtiene un numero errado. Sin embargo, los mismos se pueden despreciar y aproximar el error como el error para $t = n/2$.

La probabilidad equivalente de error (probabilidad que el mensaje se transmita sin error) entonces se calcula como:

$$p_{equ} = 1 - (1 - p_{error})^{1/k} \qquad (2)$$

Donde k es la cantidad de bits que posee el mensaje.

A continuación se muestra una imágen de los resultados del programa. En la simulación se utilizó una compresión del 90% y se realizaron 30 compresiones de 90%. Se puede ver como la primera probabilidad de error es muy baja y solo se debe al error de cuantización entre RGB y YC_bC_r. A partir de la primer compresión aumenta notablemente. Si todas las compresiones se van realizando con la misma matriz de cuantización, se puede observar como el error tiende a un valor estacionario por ser un proceso de Markov.

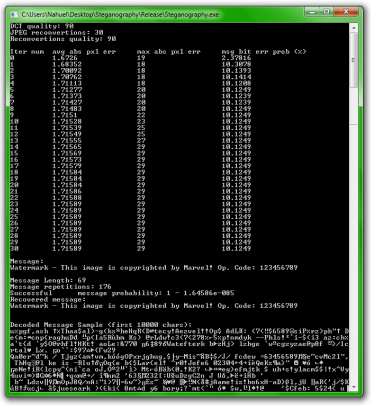

Fig. 7. Simulacion con porcentaje de compresión del 90%

JPEG aprovecha el hecho de que el ojo humano es más sensible a los cambios de lumus (Y) que a los cambios en la cromática. Éste hecho se puede observar claramente en las matrices de conversiones de la JPEG que se muestran en la figura 9. Como se puede observar, los coeficientes de cromática son generalmente mucho mayores a los coeficientes de lumus, ésto genera que para compresiones de bajo porcentaje de compresión, por ejemplo de 50%, los bits en las tablas cromáticas sean prácticamente eliminadas, aumentando el error notablemente.

Suponiendo que existe en el mensaje la mísma probabilidad de error de tener un 1 o un 0, las tablas tan estrictas de estos valores tienden generar un resultado 0 para todos sus bits. ya que responde a la ecuación:

$$p_{error} = \frac{p_{errorY} + p_{errorCb} + p_{errorCr}}{3}$$

Despreciando el error en la coordenada lumus, las probabilidades de error de los cromos se asume de 50%, por lo que el error termina siendo de un 33%, lo cual puede verificarse en la figura 8.

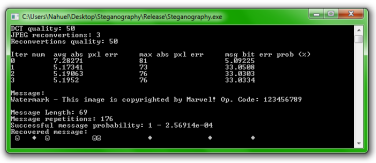

Fig. 8. Probabilidad de error con gran porcentaje de compresión

Sin embargo, no pueden utilizarse las ecuaciones 1 ni 2 ya que ya no puede suponerse que el error afecta indistintamente a un 1 o un 0. Al cero practicamente no lo afecta mientras que al 1 lo tiende a cero. Esto aumenta significantemente el error y genera problemas el recuperar el mensaje la imagen bajo ciertas condiciones de compresión. Si en estos casos se despreciase la información en las componentes Cb y Cr, que es prácticamente 0 en todos sus valores, y solo se buscase el mensaje en la componente Y, este será recuperado exitosamente.

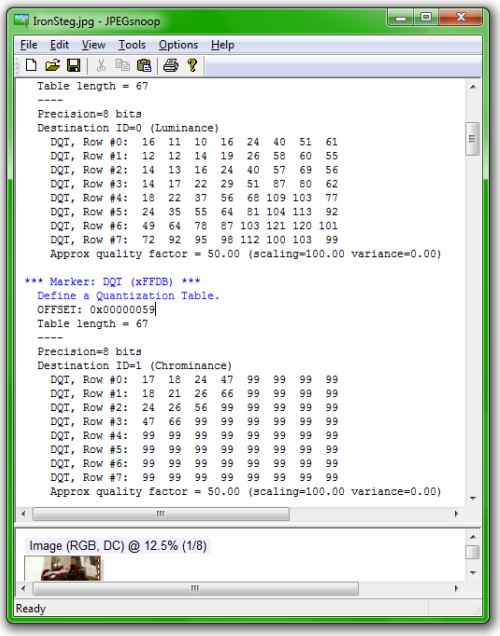

Fig. 9. JPEGsnoop Matrices de Cuantización

F. Comparación de los Métodos

Cada método posee diferentes ventajas respecto al otro. Como se pudo observar en 3, LSB posee gran cantidad de información. Es facil deducir que para LSB de 1 bit, puede incluir un archivo de un tamaño de 1/8 del tamaño de la imagen original, y 1/4 para LSB de 2-bit. En cambio, el método de la DCT puede incluir información de 1/64 en el mejor de los casos, lo cual es muchísimo menos que LSB. Sin embargo, la esteganografía por LSB posee una resistencia a la compresión nula ya que el LSB es lo primero que se suele alterar tras una compresión.

Como pudo observarse claramente, DCT, por el contrario, posee gran inmunidad a la compresión y permite realizar muchas iteraciones.

VII. Detalles finales

Para la esteganografía de imágenes dentro de imágenes se utilizó un Header en el cual se guardan las dimensiónes de altura, ancho profundidad, espectro y presición de bits (0-7) utilizada en la imagen mensaje, para que luego esta pueda ser regenerada apropiadamente.

Dado a los grandes números con los que se opera, para el cálculo del error no es posible utilizar los tipos de datos estandar de C++ debido a que estos ofrecen una presición limitada. Al calcular números grandes, como 300!, los valores superan ampliamente el máximo valor alcanzable para cualquier tipo de dato nativo. Por lo tanto se debe utilizar una librería con tipos de datos de presición arbitraria para poder calcular los errores adecuadamente.

VIII. Conclusiones

Se realizó esteganografia de texto y de imágenes sobre imágenes utilizando LSB y DCT. En el caso de la DCT se consiguió medir el error por efectos de cuantización y por efectos de compresión y decompresión a JPEG para múltiples calidades. Se estudiaron diversos fenómenos en la decodificación del mensaje por errores de cuantización. Se analizó la tendencia del error ante múltiples compresiones. Se estudió la diferencia absoluta entre la imagen inicial y con esteganografía. Se diseñó el algoritmo para que en el futurso puedan implementarse funciones que facilmente recuperen el mensaje de imágenes rotadas en múltiplos de 90 grados, espejadas, y recortadas. La esteganografía en DCT

demostró su fuerte resistencia ante compresiones de igual o preferentemente mayor calidad a la utilizada para escribir el mensaje. Se concluye que para la conservación del mensaje es preferible utilizar calidades bajas para su escritura, y calidades altas para su compresión. Así, una o múltiples compresiones de muy baja calidad tendrá más posibilidades de destruir el mensaje, pero a costo de reducir significantemente la calidad de la imagen original. Se seguirán estudiando estos fenómenos para ampliar este trabajo en el futuro.

REFERENCES

[1] R. Poornima & R. J. Iswarya (2013) "An Overview of Digital Image Steganography" International Journal of Computer Science & Engineering Survey (IJCSES) Vol. 4, No. 1, Februrary 2013.

[2] T. Morkel, J. H. P. Eloff & M. S. Olivier "An Overview of Image Steganography" Information and Computer Security Architecture (ICSA) Department of Computer Science. University of Pretoria, Pretoria, South Africa.

[3] Johnson, N. F. & Jajodia, S. (1998) "Exploring Steganography: Seeing the Unseen" Computer Journal, Februrary 1998.

[4] Sherif M. Badr, Gouda I. Salama, Gamal M. I. Selim & Ashgan H. Khalil (2014) "A Review of Steganalysis Techinques: From Image Format Point Of View". International Journal of Computer Applications. Vol. 102 - No. 4, September 2014.

[5] Zhuang Zhang, Donghui Hu, Yang Yang & Bing Su (2013) "A Universal Digital Image Steganalysis Method Based on Sparse Representation" Ninth International Conference on Computational Intelligence and Security.

[6] Qingzhong Liu, Andrew H. Sung, Mengyu Qiao, Zhongxue Chen & Bernadete Ribeiro (2010) "An improved approach to steganalysis of JPEG Images" Information Sciences 180, 1643-1655, EL SEVIER.

[7] N. V. S. Sree Rathna Lakshimi, (2014) "A Novel Steganalytic Algorithm based on III Level DWT with Energy as Feature" Research Journal of Applied Sciences, Engineering and Techonology, Maxwell Scientific Organization.

[8] Seongho Cho, Byung-Ho Cha, Martin Gawecki & C-C Jay Kuo (2013) "Block-Based image steganalysis: Algorithm and performance evaluation" J. Vis. Commun. Image R., EL SEVIER.

[9] Mansour Sheikhan, M. Shahram Moin & Mansoureh Pezhmanpour (2010) "Blind Image Steganalysis via Joint Co-occurence Matrix and Statistival Moments of Contourlet Transform" 10th International Conference on Intelligent Systems Design and Applications, IEEE.

[10] Han Zong, Fen-lin Liu, Xiang-yang Luo (2012) "Blind image steganalysis based on wavelet coefficient correlation" Digital Investigation.

[11] Pedram Hayatil, Vidyasagar Potdar "A Survey of Steganographic and Steganalytic Tools for the Digital Forensic Investigator" Curtin University of Technology, Perth, Australia.