ESTEGOANÁLISIS EN IMÁGENES CONTENEDORAS DE TEXTO OCULTO UTILIZANDO EL ALGORITMO DE SUSTITUCIÓN LSB.

PEDRO NEL JIMÉNEZ BELLO

UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA – UNAD
ESCUELA DE CIENCIAS BÁSICAS, TECNOLOGÍA E INGENIERÍA - ECBTI
ESPECIALIZACIÓN EN SEGURIDAD INFORMÁTICA
BOGOTÁ
2021

ESTEGOANÁLISIS EN IMÁGENES CONTENEDORAS DE TEXTO OCULTO UTILIZANDO EL ALGORITMO DE SUSTITUCIÓN LSB.

PEDRO NEL JIMÉNEZ BELLO

Proyecto de Grado – Monografía presentado para optar por el título de
ESPECIALISTA EN SEGURIDAD INFORMÁTICA

Joel Carroll Vargas.
Director proyecto de grado

UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA – UNAD
ESCUELA DE CIENCIAS BASICAS, TECNOLOGIA E INGENIERIA - ECBTI
ESPECIALIZACIÓN EN SEGURIDAD INFORMÁTICA
BOGOTÁ
2021

NOTA DE ACEPTACIÓN

Firma del Presidente de Jurado

Firma del Jurado

Firma del Jurado

Bogotá., octubre 14 de 2021

DEDICATORIA

A Dios mi creador, por su infinito amor y misericordia, por darme sabiduría, inteligencia y disciplina para poder lograr una nueva meta. A mi madre, por su amor, tolerancia y comprensión; quien es mi confidente, que con su amor y comprensión me ha brindado los recursos para formarme en un profesional competente, pero lo más importante en un ser humano integro con valores y principios.

AGRADECIMIENTOS

Un sincero agradecimiento a los docentes y directores de las diferentes asignaturas que componenen la especialización en Seguridad Informática, gracias a su empeño, dedicación y excelente trabajo han compartido sus conocimientos para forjarnos en este hermoso campo de la seguridad de la información. A mi Tutor y Director de trabajo de grado, Ing Joel Carrol Vargas, por su dedicación y valioso apoyo, que fue de gran ayuda para lograr consolidar el documento de la presente monografía.

CONTENIDO

LISTA DE FIGURAS

GLOSARIO

ESTEGANOGRAFÍA [1] "Arte de ocultar información. Del griego steganos (oculto) y graphos (escritura), la esteganografía se puede definir como la ocultación de información en un canal encubierto con el propósito de prevenir la detección de un mensaje oculto".

ESTEGOANÁLISIS[2] "Disciplina dedicada al estudio de la detección de mensajes ocultos usando esteganografía". Dichos mensajes pueden estar ocultos en diferentes tipos de medio, como pueden ser por ejemplo las imágenes digitales, los archivos de vídeo, los archivos de audio o incluso un simple texto plano."

[1] FIRSTCLOUDIT. Cuaderno de notas del Observatorio. *Esteganografía el arte de ocultar información*. [www.pgarciab.firstcloudit.com]. Publicación del articulo en firstcloudit, marzo de 2010 [consultado febrero 24 de 2020]. Disponible en: https://pgarciab.firstcloudit.com/documentos/esteganografia.pdf

[2] WIKIPEDIA. Estegoanálisis. [es.wikipedia.org]. Publicación de la información en Wikipedia, última actualización diciembre de 2019 [Consultado febrero 24 de 2020]. Disponible en: https://es.wikipedia.org/wiki/Estegoanálisis

RESUMEN

Las redes de datos en este siglo han tomado un papel muy importante; pues gracias a ellas el hombre puede realizar muchas tareas con facilidad, eficacia y velocidad. Servicios donde la información ha tomado un valor incalculable y es necesario utilizar procesos para proteger los datos ante el acecho de delincuentes informáticos (Black Hacker - Lamer), reduciendo al máximo posibles ataques y amenazas. Para garantizar la seguridad de la información a los usuarios en la red; se han diseñado una gran variedad de algoritmos de encriptación, muy robustos y complejos. La utilización de algoritmos de cifrado actualmente tiene dos inconvenientes: primero (1). Al utilizar canales inseguros de trasmisión los datos pueden ser interceptados por personas no autorizadas y se convierte en una amenaza sin importar que la información esté cifrada y dos (2) La perdida de confidencialidad de llaves de encriptación. Debido a los dos inconvenientes latentes en el cifrado de datos, expertos en seguridad informática han optado por utilizar otra técnica para proteger la información, es ocultándola dentro de archivos portadores, los cuales pueden ser imágenes, archivos de música o vídeos siendo imperceptibles al ser humano.

El presente documento, es el desarrollo del proyecto de grado en la especialización de seguridad informática, el cual tienen como finalidad realizar estego-análisis a una variedad de imágenes con formatos digitales más utilizados como son (jpeg, bmp y gif) que tienen mensajes secretos a través del algoritmo del bit menos significativo (LSB). Las pruebas de ocultar información en estos formatos de imágenes se realizan con la herramienta denominada StegoCrypto, la cual fue diseñada y programada por el autor. StegoCrypto permite ocultar y recuperar información en imágenes portadoras con la utilización del algoritmo del LSB, adicionalmente el usuario puede configurar el numero de bit a utilizar (1 a 5 bit menos significativos) donde se podrá evaluar visualmente sus efectos en el algoritmo. Adicionalmente existen dos opciones en aplicar el algoritmo LSB de forma continua en un tramo de Pixeles o en forma aleatoria. También, se analizó la función o librería que permite guardar la clave de cifrado y descifrado dentro de la imagen portadora (Criptografía Simétrica), para esto se utilizó la función Hash a la llave simétrica obteniendo un valor único que es ocultado dentro de la imagen portadora y solo si el receptor conoce la misma llave logrará visualizar el mensaje oculto dentro de la imagen. Finalmente se almacena el tamaño (bytes) del texto oculto y tipo de cifrado si fue secuencial o aleatorio, valores que son importantes para lograr visualizar el mensaje oculto al receptor.

Palabras claves: Cifrado, mensaje oculto, Estegonagrafía, algoritmos Simétricos, algoritmos Asimétricos, algoritmo del bit menos significativo (LSB). Marca de agua digital (Watermarking). Estegoanálisis. Algoritmo por sustitución. Imagen portadora. Confidencialidad, integridad, disponibilidad, autenticación, Algoritmo Hash, Firma digital.

ABSTRACT

Data networks in this century have taken on a very important role; because thanks to them man can perform many tasks with ease, efficiency and speed. Services where the information has taken on incalculable value and it is necessary to use processes to protect the data from the stalking of computer criminals (Crackers - YYY), minimizing possible attacks and threats. To guarantee the security of information to users on the network; A wide variety of very robust and complex encryption algorithms have been designed. The use of encryption algorithms currently has two drawbacks: first (1). By using insecure transmission channels, the data can be intercepted by unauthorized persons and becomes a threat regardless of whether the information is encrypted and two (2) The loss of confidentiality of encryption keys. Due to the two latent drawbacks in data encryption, computer security experts have chosen to use another technique to protect information, is by hiding it within carrier files, which can be images, music files or videos, being imperceptible to the human being.

This document is the development of the degree project in the specialization of computer security, which aims to carry out stego-analysis to a variety of images with most used digital formats such as (jpeg, bmp and gif) that have secret messages through the least significant bit (LSB) algorithm. The tests of hiding information in these image formats are carried out with the tool called StegoCrypto, which was designed and programmed by the autor. StegoCrypto allows hiding and recovering information in carrier images with the use of the LSB algorithm, additionally the user can configure the number of bits to use (1 to 5 least significant bits) where its effects on the algorithm can be visually evaluated. Additionally, there are two options to apply the LSB algorithm continuously in a segment of Pixels or randomly. Also, the function or library that allows saving the encryption and decryption key within the carrier image was analyzed (Symmetric Cryptography), for this the Hash function was used to the symmetric key, obtaining a unique value that is hidden within the carrier image and only if the receiver knows the same key will he be able to visualize the hidden message within the image. Finally, the size (bytes) of the hidden text and type of encryption are stored, if it was sequential or random, values that are important in order to display the hidden message to the receiver.

Keywords: Encryption, hidden message, Stegonagraphy, Symmetric algorithms, Asymmetric algorithms, least significant bit (LSB) algorithm. Digital watermark (Watermarking). Stegoanalysis. Substitution algorithm. Carrier image. Confidentiality, integrity, availability, authentication, Hash algorithm, Digital signature.

INTRODUCCIÓN

Las redes de computadores, en este siglo han tomado un papel muy importante, pues gracias a ellas el hombre puede realizar muchas tareas, con facilidad, eficacia y velocidad. Servicios donde la información a tomado un valor incalculable, y se hace necesario utilizar procesos para protegerla ante amenazas, como acceso no autorizado a la información, vulnerando la confiabilidad e integridad de los datos. Por tal motivo, dentro del campo de seguridad de la información, se encuentra la criptografía, que es una ciencia de la escritura secreta, que, aunque el atacante pueda obtener el mensaje cifrado y conocer el algortimo de encriptación el mensaje sólo podrá ser descifrado con la llave de encripción. Sin embargo, la criptografía deja en evidencia la transmisión de un mensaje que aunque es secreto puede ser víctima de ataques como hombre en el medio para vulnerar su seguridad y confidencialidad.

Debido a la anterior situación, la esteganografía que es muy diferente a la criptografía, pero tan antigua con ésta, surge como técnica para ocultar la información pasando inadvertida en canales inseguros donde al asecho existen piratas y delincuentes informáticos tratando de encontrar información valiosa para sacarle provecho. La esteganografía utiliza objetos contenedores como archivos multimedia (imágenes digitales, audios o vídeos) para ocultar mensajes secretos. Su desventaja es que si un delincuente informático para este caso partícular estegoanalista encuentra el algoritmo o patrón esteganográfico utilizado en la ocultación de los datos, la información será descubierta. Por lo anterior, con la finalidad de garantizar la confiadencialidad del mensaje, actualmente se combina la esteganografía con la criptografía; primero cifrando el mensaje secreto y luego ocultandolo dentro de un objeto contenedor (imagen, audio o vídeo), de esta manera el mensaje pasará inadvertido por canales inseguros sin levantar sospecha alguna, y de suceder será imposible que conozcan su significado dado que se requiere de una llave para lograr descifrarlo.

Con el desarrollo de este proyecto, se pretende investigar una temática, que poco se ha profundizado en el campo de la seguridad informática; pero que hoy en día tiene gran importancia después de los atentados del 11 de septiembre de 2001. Para ello, con el cumplimiento de los objetivos planteados se diseñaron cuatro prototipos que utilizan la técnica del LSB para ocultar de diferentes formas un mensaje secreto. Finalmente, se utilizan aplicaciones de uso libre para detectar posibles mensajes ocultos en archivos BMP (estego objetos) creados con los diferentes prototipos. Estas aplicaciones, basadas en técnicas estadísticas que permiten identificar mensajes secretos son las herramientas computacionales más utilizadas en lo que conocemos como estegoanálisis.

1. PLANTEAMIENTO DEL PROBLEMA

2.1 ANTECEDENTES DEL PROBLEMA

Desde tiempos remotos se conoce que el hombre ha tenido la necesidad de compartir y transmitir información. Desde antes de Cristo, grandes civilizaciones como la egipcia, romana, persa, etc. luchaban por conquistar nuevos territorios, para ello tenían que enviar mensajes a grupos de soldados para aplicar estrategias de combate. El ser humano siempre es creativo y al tener problemas o necesidades se idea soluciones. Para enviar información confidencial en papiros consideraban que no era segura, dado que enemigos podían tener acceso a ella o en algún punto del envío de la información se podría filtrar o alterar la información. Una de las primeras técnicas que me ha llamado la atención para proteger la información, fue la de rapar la cabeza del mensajero, tatuarle el cráneo con el mensaje y al crecer un poco el cabello enviarlo al destinatario (receptor). Esta técnica sencilla garantizaba los principios de la seguridad informática "Confidencialidad, disponibilidad e integridad". Con esta técnica se considera el surgimiento de una nueva ciencia denominada Estegonagrafía.

Actualmente en el mundo de las comunicaciones y los sistemas de información bajo grandes tecnologías de infraestructura tecnológica se hace más recurrente el uso de métodos de cifrado. El uso de internet ha permitido la globalización, gracias a la estandarización de protocolos y medios de comunicación liderados por la IEEE. El protocolo TCP/IP es el protocolo estándar de internet por el cual las personas pueden comunicarse y utilizar la gran variedad de servicios que existen en las redes de datos.

La seguridad en las redes de datos es un tema de continua actualidad en la vida cotidiana para todas las personas que directa o indirectamente necesiten de una computadora. De ahí que, en los últimos años, el estudio de las bases teóricas y de las implementaciones prácticas para asegurar la confidencialidad en el intercambio de información haya adquirido un gran auge. Dado que Internet es una red pública, los datos que envíe o reciba cualquier usuario pueden ser leídos, obviamente sólo por alguien que sabe del tema; atentando contra los principios integridad y confidencialidad. Para garantizar la integridad y privacidad de la información por la red, se han realizado una gran cantidad de algoritmos de encriptación.De lo anterior, podemos afirmar que al ser interceptados por un delincuente informático sin importar el algoritmo de cifrado utilizado, está vulnerable la información. Es por esto, que actualmente la Esteganografía se a posesionado en un lugar privilegiado dentro del campo de la seguridad informática como herramienta para evitar estos ataques conocidos como hombre en el medio (man in the middle attack - MITM), el cual enviar información confidencial en un canal inseguro a través de una imagen portadora es seguro, dado que para el ser humano es dificil percibir los pequeños cambios que se producen en este tipo de archivos.

2.2 FORMULACIÓN DEL PROBLEMA

Con el desarrollo de esta monografía, se pretende investigar una temática, que poco se ha profundizado en el campo de la seguridad informática; pero que hoy en día tiene gran importancia después de los atentados del 11 de septiembre de 2001. Los temas de esteganografía y estegoanálisis han tomado un papel importante en el mundo de las TIC's y en el campo de la seguridad de la información.

¿Cómo identificar información oculta en imágenes gráficas que circulan por la internet por medio del uso de una herramienta computacional?

2. JUSTIFICACIÓN

En la actualidad la tecnología y comunicaciones, es el factor tecnológico por el cual gira el planeta; en el cual la información tiene un valor incalculable; razón que conlleva a garantizar la integridad, confiabilidad y autenticidad de ésta, a los usuarios

Actualmente, no se puede pensar en ser profesional competitivo en el campo de la seguridad informática sin lograr entender el funcionamiento de los sistemas de criptografía y esteganografía. Para lograr entender como conseguir una buena seguridad de los medios de transmisión independiente de sus niveles de seguridad, debemos entender el cifrado de los datos. No es necesario conocer y entender las complejas fórmulas matemáticas que implican los algoritmos de cifrado, pero sí comprender sus resultados. La esteganografía permite ocultar información en archivos portadores en especial imágenes, audios y vídeos los cuales son de díficil percepción para el ser humano, su aplicabilidad es en diferentes entornos de la seguridad informática como por ejemplo para derechos de autor (Watermarking – Marca de agua).

Con la realización de esta monografía, se pretende indagar posibles vulnerabiidades de seguridad en aplicaciones de esteganografía, gracias a la utilización de herramientas para estegoanálisis. De lograrse encontrar debilidades en los métodos de cifrado por sustitución en especial el LSB, este proyecto será un insumo para nuevas generaciones de antivirus y sistemas de seguridad como Firewall, sistemas de detección y prevención de intrusos.

3. OBJETIVOS

4.1. OBJETIVO GENERAL

Identificar porcentajes significativos de información oculta con el algoritmo del bit menos significativo (LSB) dentro de imágenes digitales a través del uso de herramientas de estegoanálisis.

4.2. OBJETIVOS ESPECÍFICOS

- Identificar posibles ataques al software de esteganografía diseñado y codificado para el proyecto (StegoCrypto).

- Identificar estrategías con el uso de técnicas esteganográficas para la protección de datos en sus tres pilares: Confidencialidad, integridad y disponibilidad.

- Investigar herramientas de estegoanálisis, determinando la más efectiva para las pruebas de información oculta con “StegoCrypto”.

- Identificar posibles vulnerabilidades en el software de esteganografía “*StegoCrypto*” por medio de una herramienta de estegoanálisis.

- Generar un informe técnico mencionando las vulnerabilidades encontradas y recomendaciones para reducir el riesgo.

4. MARCO CONCEPTUAL

5.1. ESTADO DEL ARTE

Más de 400 años antes de Cristo, Herodoto ya reflejó en su libro *Las Historias* el uso de la esteganografía en la antigua Grecia. En su libro describe, como un personaje toma un cuadernillo de dos hojas o tablillas; raya bien la cera que las cubre y en la madera misma graba un mensaje y lo vuelve a cubrir con cera.

Otra historia, en el mismo libro, describe como otro personaje rasura a navaja la cabeza de uno de sus esclavos y le tatúa un mensaje en el cuero cabelludo. Así, espera a que le vuelva a crecer el cabello y lo manda al receptor del mensaje con instrucciones de que le rasure la cabeza.

Figura 1 Tabla con mensaje oculto cubierto de cera.

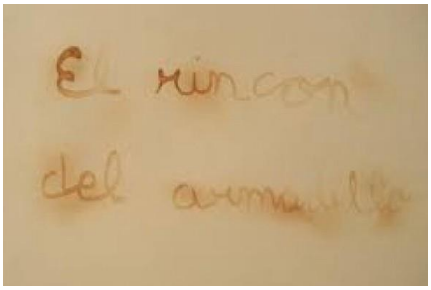
Fuente: https://core.ac.uk/download/pdf/79176967.pdf

Un ejemplo histórico más de uso de la esteganografía es el libro *Hypnerotomachia Polophili* de Francesco Colonna, que data de 1499. En él, tomando la primera letra de sus 38 capítulos se puede leer “Poliam frater Franciscus Columna peramavit”, que se traduce por “El hermano Francesco Colonna ama apasionadamente a Polia”.

De manera similar, durante la Segunda Guerra Mundial se hacen pequeñas perforaciones sobre letras de interés de un periódico de tal forma que al sostenerlo a la luz se puede observar todas aquellas letras seleccionadas e interpretarlas en forma de mensaje.

Un ejemplo más familiar, es el de la tinta invisible. Son muchos los niños que juegan a enviarse mensajes escritos con zumo de limón o sustancias similares (de alto contenido de carbono), de tal forma que al calentar la superficie sobre la que se escribe el mensaje, éste aparece en un tono color café. Esta técnica se puede hacer más compleja si se involucra reacciones químicas.

Figura 2 Mensaje oculto con tinta invisible.



Fuente: https://100cia.site/index.php/quimica/item/3114-como-revelar-mensajes-de-tinta-invisible.

5.2. MARCO TEÓRICO

La esteganografía y la criptografía son campos distintos. En la criptografía, el objetivo es asegurar la confidencialidad e integridad de la información ante los ojos de un atacante o delincuente informático que puede lograr ver el criptograma, aún cuando éste conoce el algoritmo que lo genera. En cambio, la esteganografía busca ocultar la presencia del mensaje en sí; ya que si se llega a identificar la posición del mensaje se conoce directamente la comunicación, lo que no sucede con el criptograma.

El desarrollo de esta monografía se centra en el objeto contenedor más utilizado que son las imágenes digitales, con extensiones (*.bmp, *.jpeg, *.gif y *.png). De estos formatos, el más sencillo es el -BMP dado que no se le realiza comprensión, pero la aplicación del patrón esteganográfico aplica a diferentes objetos contenedores siempre y cuando se respeten las particularidades de cada formato.

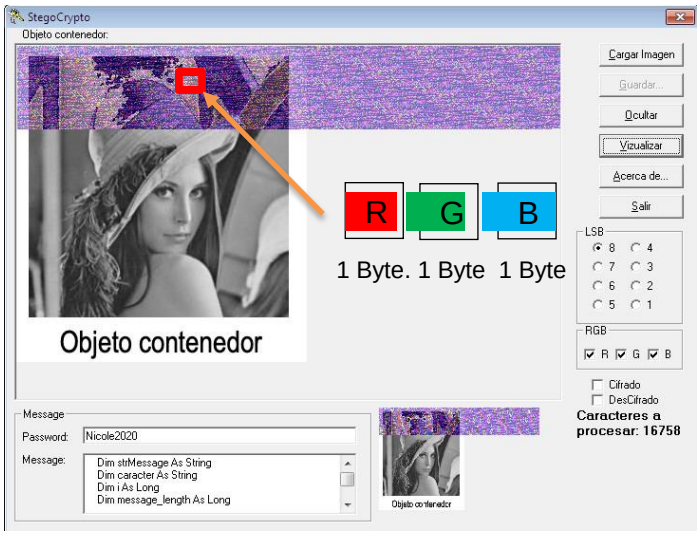
5.2.1. Sustitución de bits del objeto contenedor.

Esta técnica consiste en sustituir ciertos bits del objeto contenedor por los de la información a ocultar. La ventaja de este enfoque es que el tamaño del fichero contenedor no sea modificado y la calidad de la imagen se mantenga siempre y cuando se utilice alguno de los últimos 4 bits menos significativos.

En las imágenes, el método tradicional consiste en sustituir los bits menos significativos (LSB), en una escala de color de 24 bits (RGB) 8 bits asignados a cada color (Red – Green - Blue) nos permite obtener 2^24 (16.777.216), que equivalen a más de 16 millones de colores. Al reemplazar por ejemplo el último bit de color Rojo dentro del píxel se observará en un 0,39% más oscuro o igual si el bit LSB de ese píxel es uno (1). En un alto porcentaje (90%) de casos son cambios inapreciables a los sentidos humanos que tan sólo pueden ser detectados con análisis computacionales de la estructura del archivo contenedor.

Los archivos con extensión BMP, son imágenes estándar de mapa de bit para diversos sistemas operativos como Windows, Linux y MAC. Su configuración básica es de 8 bits, lo cual proporciona sólo 256 colores posibles, cantidad que no es suficiente para utilizarlos como objetos contenedores dado que, aunque se utilice el bit menos significativo (LSB) sería muy fácil su detección visual. Los archivos BMP de 24 Bit ofrecen una gran variedad de colores, exactamente 16’777.216 variedades de tonalidades que permiten trabajar en escala de grises, RGB y CMYK. Para el caso particular en el desarrollo del presente proyecto se trabajarán este tipo de archivos a 24 bit a escala RGB. En la siguiente figura se representa el píxel dentro de un objeto contenedor.

Figura 3 Estructura del Píxel en una imagen BMP.

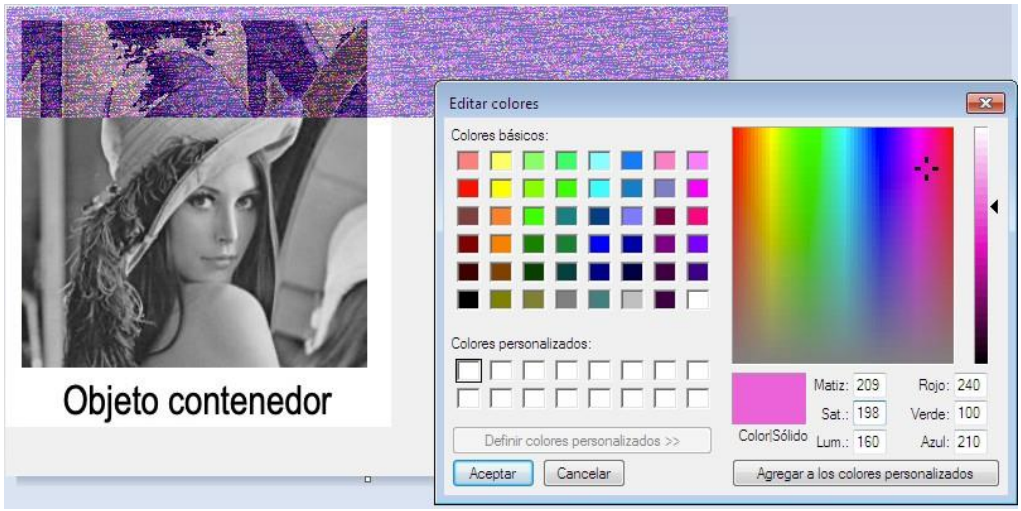


Fuente: El autor.

Cada píxel está representado por tres (3) bytes, cada uno de estos bytes tiene respectivamente la intensidad de Red, Green ,Blue. De igual forma cada byte tiene un valor de cero (0) a 255 que en el sistema binario sería [00000000] – [11111111],

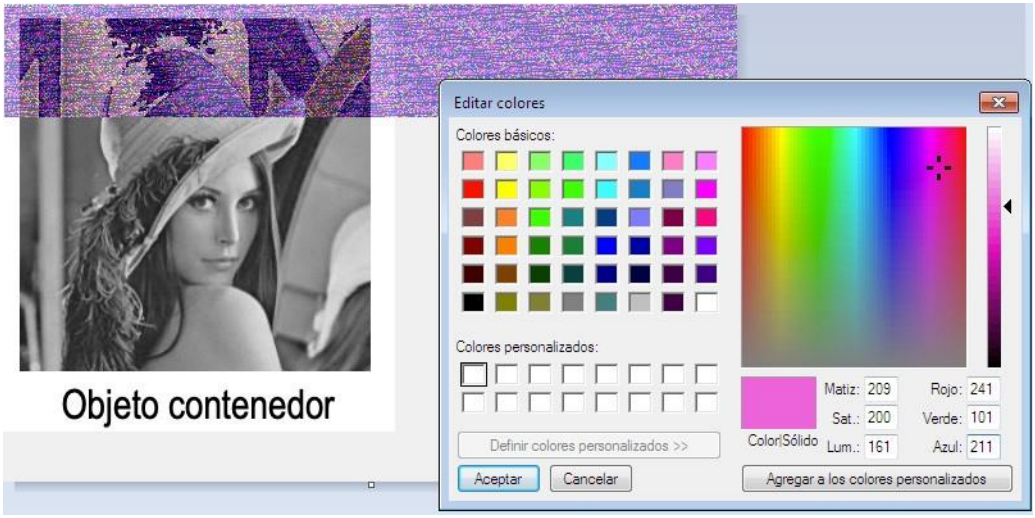
donde el dígito de la izquierda es el de mayor peso [2^7] y el de la derecha el de menor [2^1], lo que significa que se puede modificar el bit de la derecha produciendo variación en un rango de [-1, 1]. Con la utilización del accesorio MS Paint del sistema operativo Windows, en la siguiente figura se observa la variación del bit LSB del color Azul (Blue), confrontando que visualmente es imposible apreciar la variación de la tonalidad para el ojo del ser humano.

Figura 4 Variación de la escala RGB en su LSB dentro de un pixel.



Fuente: El autor.

Figura 5 Inserción de tres bit en la escala RGB.

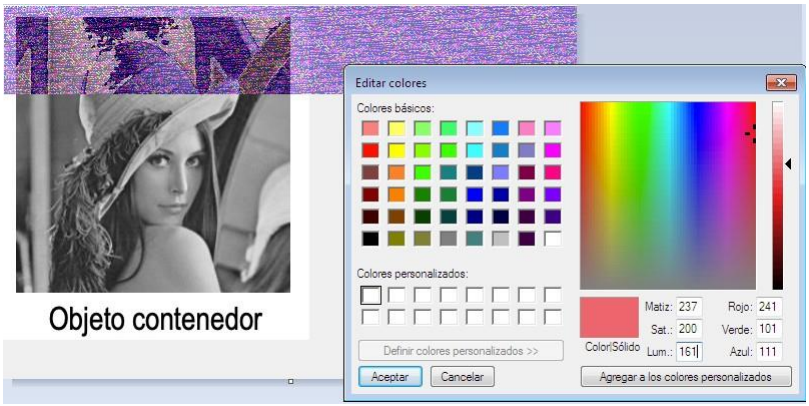


Fuente: El autor.

Como se puede comparar entre la figura 4 y 5, la variación del bit LSB en el píxel, su efecto es inapreciable a la vista del ser humano. En la práctica un píxel está rodeado por otros, lo cual su efecto visual que es del 0,31% de variación por estegoanálisis manual es imposible detectar que hay información oculta.

La figura 6, nos ilustra en la paleta de colores RGB, la variación del color azul (Blue) en su tercer bir más significativo (3 bit de derecha a izquierda). Comparando con las figuras anteriores (5 y 4), visualmente es claro el cambio de tonalidad en el pixel.

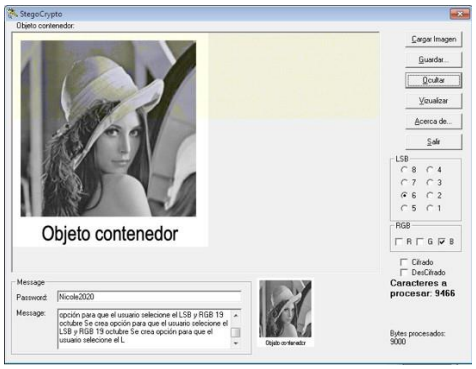
Figura 6 Inserción de un (1) bit en el 5 bit LSB del canal azul (Blue).



Fuente: El Autor.

Utilizando el prototipo número dos (2) del proyecto con la misma imagen contenedora de la figura 6, se oculta un mensaje secreto de más de nueve mil caracteres utilizando el 5 bit LSB del color Blue dentro de la escala RGB. Como se puede apreciar en la siguiente figura número 7, visualmente se aprecia el cambio de tonalidades en la parte superior, deduciendo que es un estego objeto.

Figura 7 Mensaje oculto bit 6 LSB color Blue.



Fuente: StegoCrypto Protoripo II.

Con la utilización de la escala RGB, almacenado 1 bit en cada color, podemos deducir que se requieren 8 Píxeles para ocultar 3 bytes de información. De acuerdo con lo anterior, en un mapa de bit de 800x 600 se tendrían 480.000 píxeles, haciendo una regla de tres por cada 8 píxeles tenemos 3 bytes en 480.000 podemos ocultar 180.000 bytes (caracteres), equivalente a un 37.5% de tamaño de la imagen.

Cuando son mapas de bit (bmp), el proceso de esteganografía por el LSB es una actividad relativamente sencilla con relación a otro objetos contenedores. La técnica del LSB se dificulta con otros formatos, pero la filosofía es la misma. Por ejemplo en archivos GIF, se requiere la modificación de los índices que apuntan a la paleta de colores y en los archivos JPG se debe sustituir el coeficiente cuantificado DCTs. En archivos multimedia como MP3, se requiere insertar los bits en los espacio de silencio para no afectar el audio.

5.2.2. Estructura de un Mapa de bit - BMP

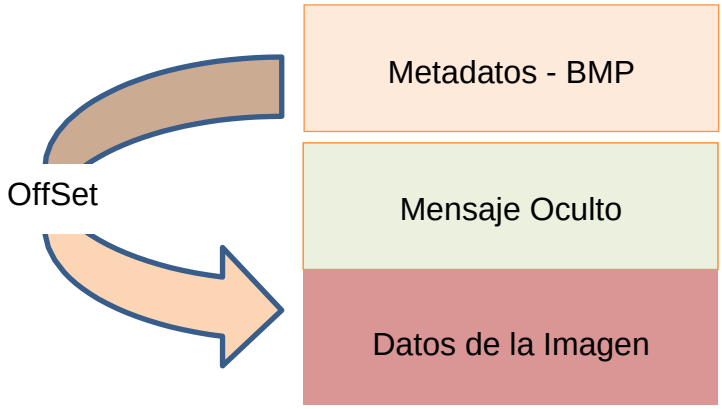
Una técnica de esteganografía es ocultando el mensaje en una de las partes de la estructura del fichero, como el final (EOF) o encabezado de éste. Sin importar la opción, la inserción presenta el inconveniente de que se modificará el tamaño del objeto contenedor (Archivo BMP), a mayor información a ocultar directamente proporcional será el tamaño del archivo contenedor, produciendo como desventaja la sospecha de ser un estego objeto.

Para entender la inserción de bits en el objeto contenedor (BMP), se debe comprender la estructura de este tipo de formato. El mapa de bit se compone de dos estructuras: La primera son los metadatos y la segunda son los datos de la Imagen. Los metadatos son 54 bytes que se dividen así:

- 2 Bytes: almacena la cadena ‘BM’, informando que el archivo se trata de un mapa de bit.
- 4 Bytes: Tamaño del archivo expresado en bytes.
- 4 Bytes: Contiene ceros, está reservado para usos futuros.
- 4 Bytes: OffSet, valor que indica la distancia entre la cabecera y el primer píxel de la imagen.
- 4 Bytes: tamaño del metadato.
- 4 Bytes: especifica número de píxeles horizontales (Ancho de la imagen).
- 4 Bytes: especifica número de píxeles verticales (Alto de la imagen).
- 2 Bytes: número de planos de color.
- 2 Bytes: profundidad de color.
- 4 bytes. Tipo de comprensión. Para mapa de bits (BMP) su valor es cero dado que este formato no es comprimido.
- 4 bytes: cantidad de colores usados (8 o 24 bits).
- 4 Bytes: cantidad de colores importados.

De acuerdo con la descripción de la estructura de un mapa de bit, la forma de ocultar información es entre las dos estructuras que la componen. Es decir, entre los metadatos y los datos de la imagen, pero se debe modificar el campo OffSet para mantener la integridad de la imagen. La siguiente figura ilustra la inserción de bits dentro de un mapa de bit.

Figura 8 Esteganografía por inserción en mapas de bit (BMP).



Fuente: El Autor.

Con la representación de la estructura de un archivo BMP, se deduce que esta técnica sólo es recomendable para ocultar mensajes muy pequeños, pues resultaría sospechoso tener imágenes como iconos con resolución de 20x20 (400 píxeles) y con tamaño del archivo de cientos de KB. Una solución, planteada para esta técnica de inserción de bits, es dividir el mensaje secreto en tramas o bloques e insertarlos en varios archivos BMP.

5.2.3. EstegoAnálisis.

Estegoanálisis, es una técnica o uso de varias técnicas para recuperar o detectar mensajes ocultos dentro de objetos contenedores y en algunas ocasiones modificar el mensaje para vulnerar la integridad de éste. Existen dos técnicas de estegoanálisis: (1) Manual y (2) Estadístico.

- Estegoanálisis manual. Se caracteriza por identificar diferencias entre el objeto contenedor y el estego objeto buscando cambios en la estructura que permita identificar datos ocultos. El principal inconveniente de esta técnica es disponer del objeto contenedor y si se detecta información oculta en el estego objeto es casi imposible recuperarla.

Sin embargo, en muchos casos que no se dispone del objeto contenedor se pueden identificar irregularidades en el estego objeto para tratar de encontrar indicios de la existencia de información oculta.

Con la utilización de filtros se realizan ataques visuales que alertan al analista de la presencia de datos ocultos. Por ejemplo, el caso de un mapa de bit donde el LSB se almacene información secreta, se aplica manualmente un filtro en el LSB de cada píxel en sus tres componentes RGB.

La figura 9, nos muestra como al aplicar el filtro en el LSB de la parte superior del archivo BMP, se observa pequeñas líneas uniformes horizontales.

Figura 9 Estegoanálisis manual en el LSB de un archivo BMP.

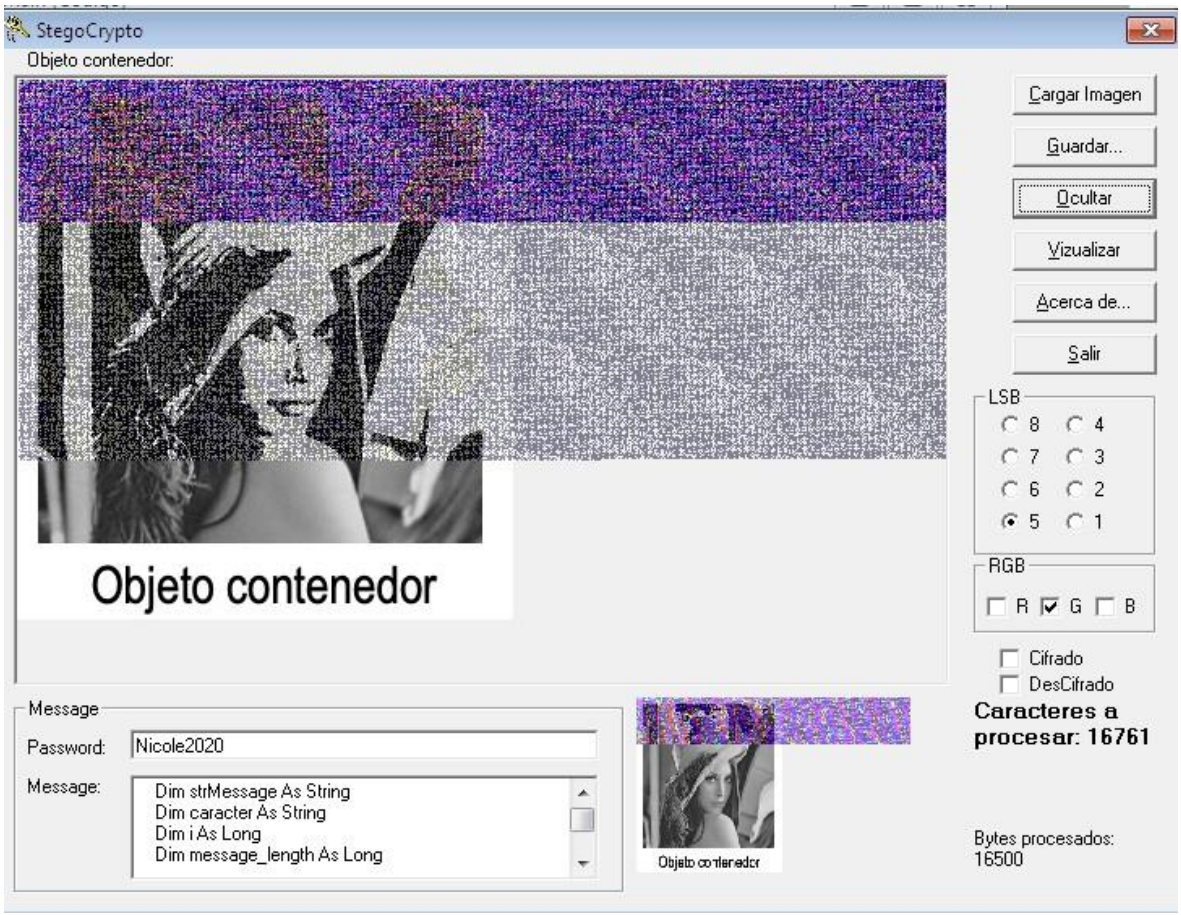
Fuente: StegoCrypto

La esteganografía manual es exitosa, cuando se utilizan imágenes con poca variedad de colores y regiones uniformes, por lo cual siempre se recomienda el uso de imágenes naturales no artificiales con una gran variedad de tonos y colores.

- Estegoanálisis estadístico. Se basa en el análisis de frecuencias de distribución de colores en el estego objeto. Esta técnica tiene como desventaja su lentitud y para su aplicación debe ser con software especializado. Se obtienen buenos resultados cuando el mensaje secreto fue oculto con programas típicos de esteganografía, pero es prácticamente imposible detectar información cuando el mensaje fue ocultado manualmente.

Una de las técnicas más utilizada en el estegoanálisis estadístico es Chi-Cuadrado, que permite estimar el tamaño de la información oculta en el estego objeto. La figura 10 ilustra el filtrado a un estego objeto.

Figura 10 Estegoanálisis estadístico, filtrado a un estego objeto.



Fuente: Stegocrypto.

5.3. MARCO CONCEPTUAL

- **ESTEGANOGRAFÍA** [3] Arte de ocultar información. Del griego steganos (oculto) y graphos (escritura), la esteganografía se puede definir como la ocultación de información en un canal encubierto con el propósito de prevenir la detección de un mensaje oculto.

  La esteganografía estudia el conjunto de técnicas cuyo fin es insertar información sensible dentro de otro fichero. A este fichero se le denomina archivo contenedor (gráficos, documentos, programas ejecutables, etc.). De esta forma,

[3] FIRSTCLOUDIT. Cuaderno de notas del Observatorio. *Esteganografía el arte de ocultar información*. [www.pgarciab.firstcloudit.com]. Publicación del articulo en firstcloudit, marzo de 2010 [consultado febrero 24 de 2020]. Disponible en: https://pgarciab.firstcloudit.com/documentos/esteganografia.pdf

se consigue que la información pase inadvertida a terceros, de tal forma que sólo sea recuperada por un usuario legitimo que conozca la clave de extracción de la misma.

- **ESTEGOANÁLISIS** [4] "Disciplina dedicada al estudio de la detección de mensajes ocultos usando esteganografía". Dichos mensajes pueden estar ocultos en diferentes tipos de medio, como pueden ser por ejemplo las imágenes digitales, los archivos de vídeo, los archivos de audio o incluso un simple texto plano."

  A diferencia del criptoanálisis donde es necesario descrifrar el mensaje para considerar roto un criptosistema, con el estegoanálisis basta con lograr detectar la existencia de un menaje oculto dentro de un objeto contenedor para considerar el sistema vulnerable.

- **Objeto contenedor:** Se trata de la entidad que se emplea para portar el mensaje oculto. Acudiendo al ejemplo de los mensajes sobre el cuero cabelludo, el objeto contenedor es el esclavo en sí.

- **Estego-objeto:** se trata del objeto contenedor más el mensaje encubierto. Para nuestro ejemplo anterior, se trata del esclavo una vez se ha escrito en su cuero cabelludo el mensaje y se le ha dejado crecer el cabello.

- **Adversario:** Son todas aquellas personas o herramientas de análisis a los que se les trata de ocultar la información encubierta. Los adversarios pueden ser pasivos o activos. El adversario pasivo sospecha que se puede estar produciendo una comunicación encubierta y trata de descubrir el algoritmo que se extrae del estego-objeto, pero no trata de modificar dicho objeto. Un adversario activo, además de tratar de hallar el algoritmo de comunicación encubierta, modifica el estego-objeto con el fin de corromper cualquier intento de mensajería subliminal.

- **Estegoanálisis:** ciencia que estudia la detección (ataques pasivos) y/o anulación (ataques activos) de información oculta en distintos objetos contenedores, así como la posibilidad de localizar la información útil dentro de la misma.

5.4. MARCO LEGAL

[4] WIKIPEDIA. Estegoanálisis. [es.wikipedia.org]. Publicación de la información en Wikipedia, última actualización diciembre de 2019 [Consultado febrero 24 de 2020]. Disponible en: https://es.wikipedia.org/wiki/Estegoanálisis

Ley 1273 del 5 de enero de 2009. De la protección de la información y de los datos. [5]"por medio de la cual se modifica el Código Penal, se crea un nuevo bien jurídico tutelado – denominado "*de la protección de la información y de los datos*" y se preservan integralmente los sistemas que utilicen las tecnologías de la información y las comunicaciones, entre otras disposiciones".

Art.269 C INTERCEPTACIÓN DE DATOS INFORMÁTICOS. El que, sin orden judicial previa intercepte datos informáticos en su origen, destino o en el interior de un sistema informático o las emisiones electromagnéticas, provenientes de un sistema informáticos que los transporte incurrirá en pena de prisión de treinta y seis (36) a setenta y dos (72) meses"

[5] SIC. Diario oficial. Ley 1273 de 2009 [sic.gov.co]. Última actualización septiembre de 2011 [Consultado febrero 27 de 2020]. Disponible en: https://www.sic.gov.co/recursos_user/documentos/normatividad/Ley_1273_2009.pdf

6. DISEÑO METODOLÓGICO

6.1. Tipo de investigación: según Colciencias[6], el tipo de *investigaciones experimentales* se emprenden principalmente para obtener nuevos conocimientos de fenómenos y hechos observables, sin pensar en darles ninguna aplicación o utilización específica, por lo cual esta monografía se clasifica desde un enfoque cuantitativo al observar que mediante estegoanálisis estadístico se obtienen tramas de mensajes ocultos en objetos contenedores de mapas de bit (BMP).

6.2. Alcance de la investigación: Identificar posibles vulnerabilidades en el software de esteganografía "*StegoCrypto*" por medio de una herramienta de estegoanálisis.

[6] Departamento Administrativo de Ciencias, Tecnología e Innovación- Colciencias. Tipo de Investigación [en línea]. Publicación Gestión de conocimiento. Colombia. 2020. [Fecha de consulta 27 agosto 2020]. Disponible en: https://www.funcionpublica.gov.co/web/eva/tipos-de-investigacion

7. DESARROLLO DE LOS OBJETIVOS.

7.1. DESARROLLO OBJETIVO 1. Identificar posibles ataques al software de esteganografía diseñado y codificado para el proyecto (StegoCrypto).

7.1.1. StegoCrypto – Prototipo I

El primer prototipo de StegoCrypto, inicia con el diseño y codifiación de la ventana “Splash” que tiene como función visualizar los datos informativos de la aplicación. El splash es la ventana de bienvenida que se diseña para todo software. La ventana se puede observar por 6 segundo y por medio de un *timer* se activa para terminar la ejecución del formulario y visualizar la ventana principal de la aplicación (frmMain). El código del formulario splash se describe en el Anexo I.

Al cargar el formulario Splash, se ejecuta la función NewGradient, que se encarga de dibujar el degradado a través de todo el formulario. La Figura 11, visualiza el formulario Splash en ejecución.

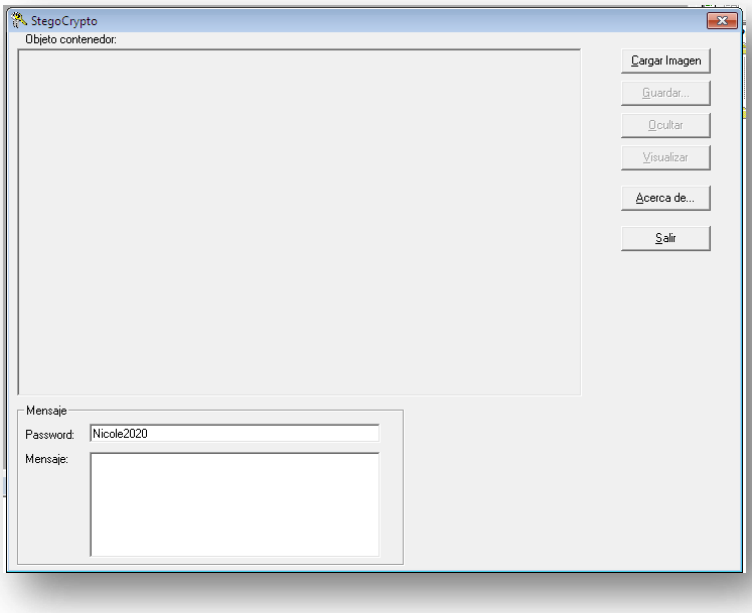
Figura 11 Formulario Splah – StegoCrypto Ver 1.0.

Fuente: El Autor.

Al pasar 6 segundos se ejecuta el proceso que termina la ejecución del formulario y visualiza la ventana principal (frmMain) de StegoCrypto. Esta ventana cuenta con dos cajas de texto, la primera es el password a utilizar para ocultar o visualizar la información oculta de un objeto contenedor. Por defecto, se deja el password *Nicole2020* y por motivos de pruebas, se visualiza legible al usuario. La segunda

caja de texto es para que el usuario introduzca el texto a ocultar o si el proceso es inverso se mostrará el mensaje oculto dentro del objeto contenedor. La Figura 12, visualiza el formulario Principal (frmMain) en ejecución.

Figura 12 Formulario Principal – StegoCrypto Ver 1.0.

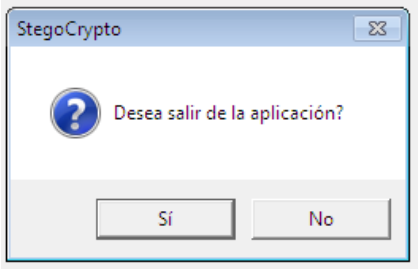


Fuente: El Autor.

StegoCrypto por defecto tiene activo tres botones de pulsación: Cargar imagen, Acerca de… y Salir. El Botón de pulsación “Salir” permite cerrar el programa StegoCrypto, previamente confirma al usuario si desea cerrar la aplicación. El código del botón “Salir” al evento de realizar click se describe en el Anexo I.

Al suceder el evento Click del Botón Salir, se visualiza una ventana emergente, preguntando la confirmación si el usuario desea cerrar StegoCrypto. La Figura 13, visualiza la ventana emergente de confirmación para cerrar la aplicación.

Figura 13 Ventana emergente, confirmar el cierre de StegoCrypto.



Fuente: El Autor.

La ventana emergente, se caracteriza en que el usuario sólo puede tener iteracción con ésta, hasta que seleccione una de las opciones dadas. De hacer click en el botón Si, se cerrará la aplicación y el caso de hacer click en el botón No, regresará a la ventana principal del software.

El segundo botón de pulsación activo es “Acerca de…”, visualiza información comercial del programa. El código del botón “Acerca de…” al evento de realizar click se describe en el Anexo I.

Al suceder el evento Click del Botón “Acerca de…”, con la propiedad Show del formulario FrmAcercaDe se visualiza este formulario como una ventana emergente, gracias al parámetro vbModal. La Figura 14, visualiza el formulario “AcercaDe” en ejecución.

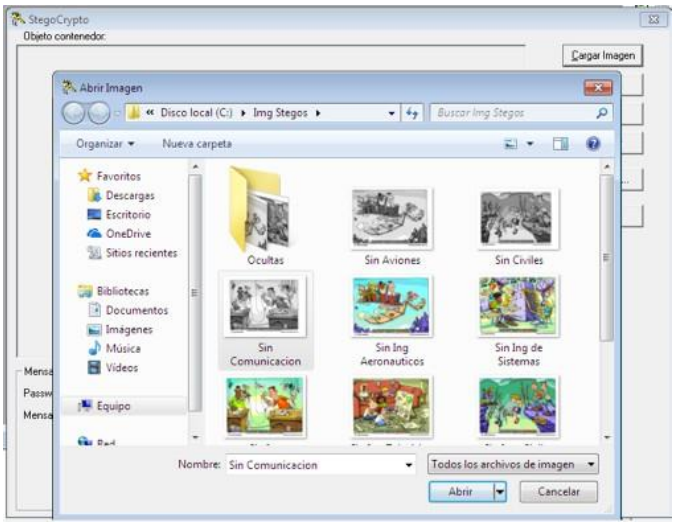
Figura 14 Formulario AcercaDe – StegoCrypto Ver 1.0.

Fuente: El Autor.

El tercer botón de pulsación activo es “Cargar imagen”, visualiza un cuadro común de diálogo que es predefinido por Microsoft para acciones estándar dentro de los sistemas operativos Windows como: Abrir,Guardar, Guardar como e imprimir. El código del botón “Cargar imagen” al evento de realizar click se describe en el Anexo I.

La propiedad “ShowOpen”, especifica que el control Cdialog se utilizará para abrir archivos y con la propiedad “Filter” se específica las extensiones de archivos preestablecidos a cargar. Para los alcances de este proyecto, los archivos contenedores con imágenes con las principales extensiones como *.bmp, *.jpg, *.gif y *.png. La Figura 15, visualiza el control CommonDialog (CDialog) funcionando como una ventana estándar para abrir archivos previamente de haber realizado click en el botón “Cargar imagen”.

Figura 15 Ventana estándar Abrir archivo – StegoCrypto Ver 1.0.



Fuente: El Autor.

Al seleccionar una de las imágenes dentro de una carpeta contenedora de la unidad de almacenamiento (Disco local C:\) y reallizr click en el botón “Abrir”, la imagen respectiva se cargará en el picture como archivo contenedor para el proceso de esteganografía. La Figura 16, visualiza el resultado al haber seleccionado una de las imágenes contenidas dentro de la carpeta C:\Img Stegos.

Figura 16 Ventana principal visualizando el objeto contenedor.

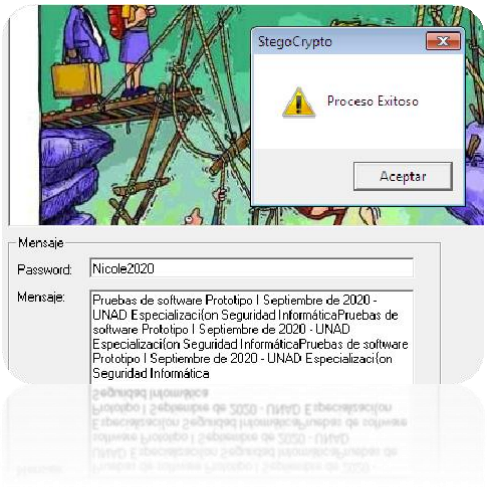
Fuente: El Autor.

Ahora en la ventana principal de StegoCrypto, dos botones de pulsación se encuentran activos: “Ocultar” y “Visualizar”.El botón “Ocultar” nos permite esconder el mensaje secreto digitado en la caja de texto Mensaje. El código del botón “Ocultar” al evento de realizar click se describe en el Anexo I.

El anterior proceso, utiliza la función CalculaHash, que está basada en el algoritmo MD5, se envía como parámetro el password y la función retorna un número único de tipo long. Este número se oculta como parte del mensaje oculto y será la referencia para visualizar el mensaje secreto por el receptor. La otra función es EncodeByte, que se encarga de recibir el valor numérico Ascii de cada carácter que hace parte del mensaje secreto. Ésta función realiza un ciclo en 8 ocasiones para almacenar cada bit del valor Ascii recibido como parámetro. El código de la función “EncodeByte” se describe en el Anexo I

La función EncodeByte, almacena el bit de cada posición del valor Ascii (valor numérico de cada carácter que constituye el mensaje secreto) con la sentencia “R And &HFE” deja en cero (0) el bit menos significativo (LSB) del pixel Red y con el operador lógico Or bit suma éste al pixel. Cuando termina de realizar el ciclo for al llegar al número de caracteres que contiene el mensaje secreto, StegoCrypto visualiza un mensaje informando que el proceso se realizó con éxito, como se observa en la figura 17:

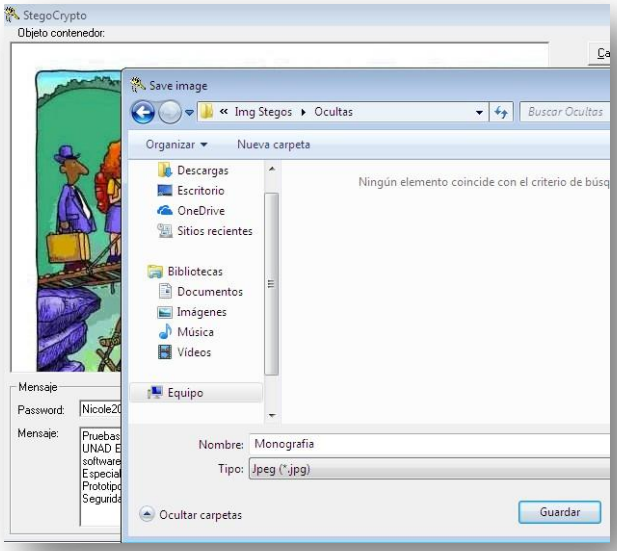
Figura 17 Proceso éxitoso de ocultar el mensaje en el objeto contenedor.



Fuente: El Autor.

Al dar click en el botón Aceptar de este cuadro de mensaje, observamos que el botón Guardar se encuentra activo para que el usuario guarde el objeto contenedor como una imagen digital y lo pueda enviar al usuario receptor. A manera de ejemplo guardaremos el Estego Objeto con el nombre monografía, como se visualiza en la figura 18.

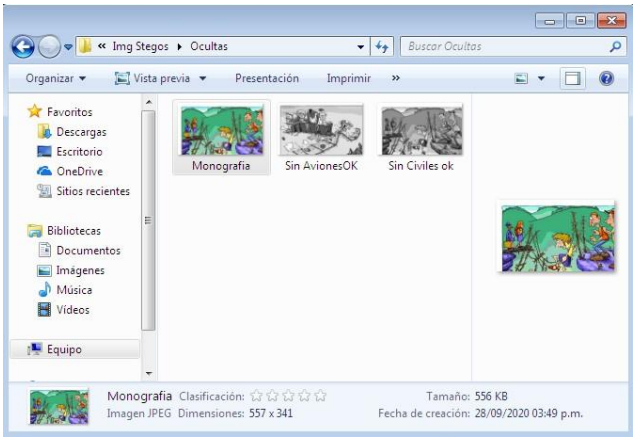
Figura 18 Formulario Estándar “Gua**rdar” – Estego Objeto.**



Fuente: El Autor.

Con el explorador de Windows nos desplazamos en el path donde se almacenó el Estego Objeto y verificamos su existencia. La figura 19 visualiza la existencia del archivo monografía.jpg dentro del path c:\Img Stegos\Ocultas\

Figura 19 Monografia.jpg Estego Objeto creado con StegoCrypto.



Fuente: El Autor.

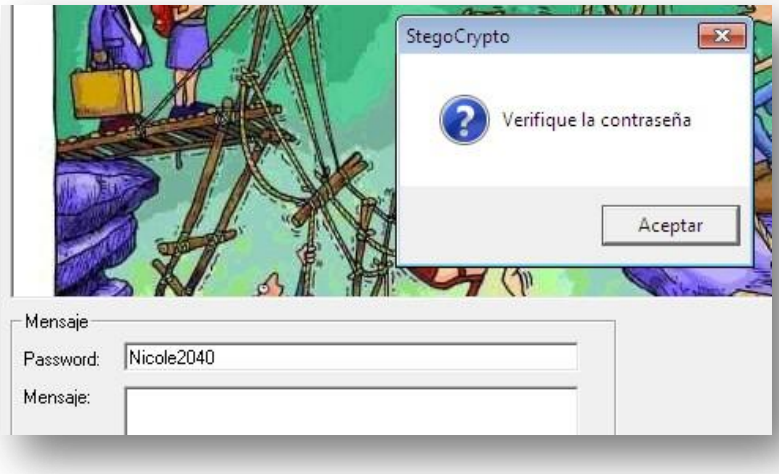
El procedimiento del botón “Guardar”, es similar a la codificación del botón “Cargar imagen”, gracias al control CommonDialog que nos ofrece el diseño y gran parte del

codigo del sistema operativo. El código del botón “Guardar” al evento de realizar click se describe en el Anexo I.

La novedad de este código es la propiedad showsave, para indicar que el cuadro preestablecido a utilizar es el de guardar archivos y la función predeterminada SavePicture incluida en el lenguaje de programación para guardar un objeto picture como archivo.

La finalización de este primer prototipo es con el procedimiento inverso a ocultar el mensaje secreto, es decir **Visualizarlo**. Por lo cual el usuario debe realizar click en el botón cargar imagen, ir al path donde tiene almacenado el Estego Objeto (monografía.jpg), click en el botón Abrir. Una vez realizados estos pasos, se observará el archivo monografía.jpg en el objeto contenedor de la ventana principal de la aplicación. El usuario debe digitar el respectivo Password, para nuestro caso de pruebas es Nicole2020 y finalmente realizar click en el botón “Visualizar”. Relizaremos la prueba (1) digitando un Password incorrecto y (2) digitando el password correcto. La figura 20 visualiza un cuadro de mensaje informando que el password es incorrecto.

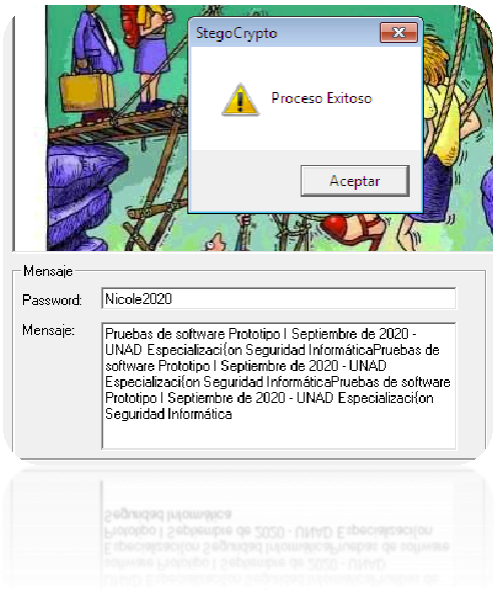
Figura 20 Cuadro de Mensaje “Verifique la contraseña” para el Estego Objeto.



Fuente: El Autor.

La segunda alternativa, es si el usuario digita el Password correcto. La figura 21 visualiza un cuadro de mensaje informando que el proceso fue exitoso! Y visualiza el mensaje secreto ocultado dentro del Estego Objeto.

Figura 21 Visualización del mensaje secreto – Estego Objeto monografia.jpg.



Fuente: El Autor.

Como se describió en los parráfos anteriores, el botón "Visualizar" nos permite conocer el mensaje secreto oculto dentro del Objeto contenedor (Estego Objeto). El código del botón "Visualizar" al evento de realizar click se describe en el Anexo I

Los TAG, en especial el "@", es el indice base para poder determinar el tamaño del Hash y del mensaje secreto. La función char convierte un valor numérico en carácter de acuerdo a la tabla Ascii.

```
Select Case C
    Case 0, 1, 2
        bit = (R And &H1)
End Select
' Incremento el valor del byte
If bit Then
    Value = Value Or offset
End If
offset = offset * 2
```

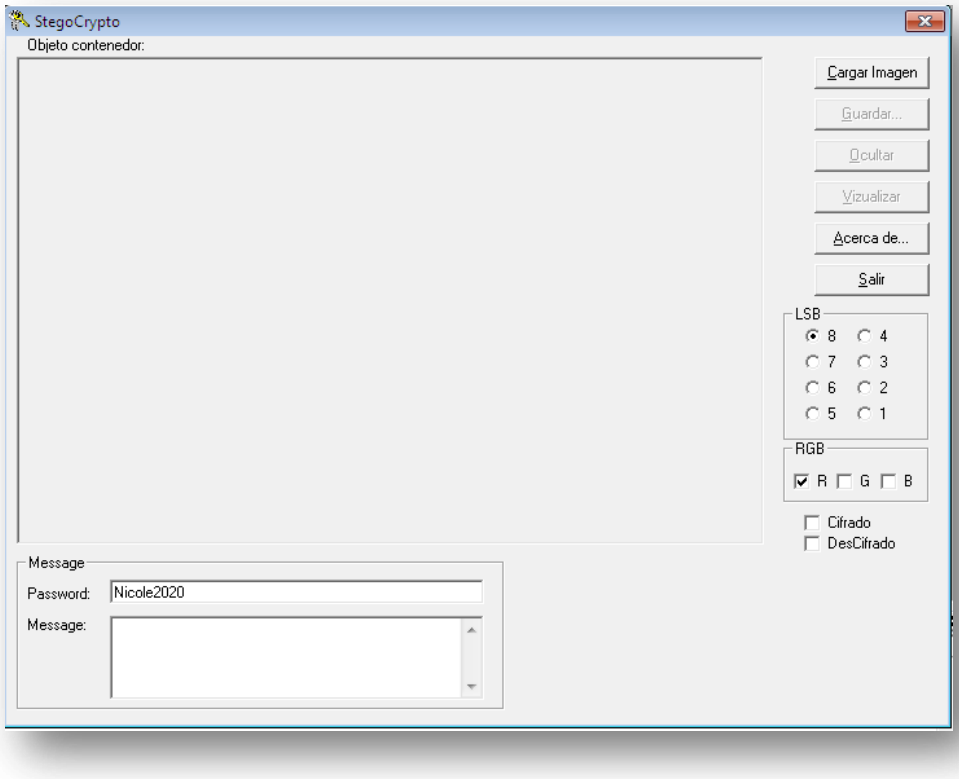
Como se observa en el código anterior, la instrucción bit = (R And &H1) es la que me permite extraer el bit (LSB) del pixel Red y la instrucción Value = Value Or offset es la suma del bit según su posición dentro del byte para determinar el valor numérico del carácter almacenado en el Estego Objeto.

7.1.2. StegoCrypto – Prototipo II

Con el prototipo I, se logró el proceso de esteganografía con la técnica del LSB, con el diseño y codificación del segundo prototipo su finalidad es proporcionar una interface más completa donde como usuario y experto en la temática logremos

observar los efectos que surgen en el Estego Objeto con la variación de parámetros en el bit menos significativo (1 al 8) y las posibles combinaciones de los colores de pixel a utilizar (R, G, B, RG, RB, GB). La figura 22 visualiza la ventana principal de StegoCrypto en su segundo prototipo.

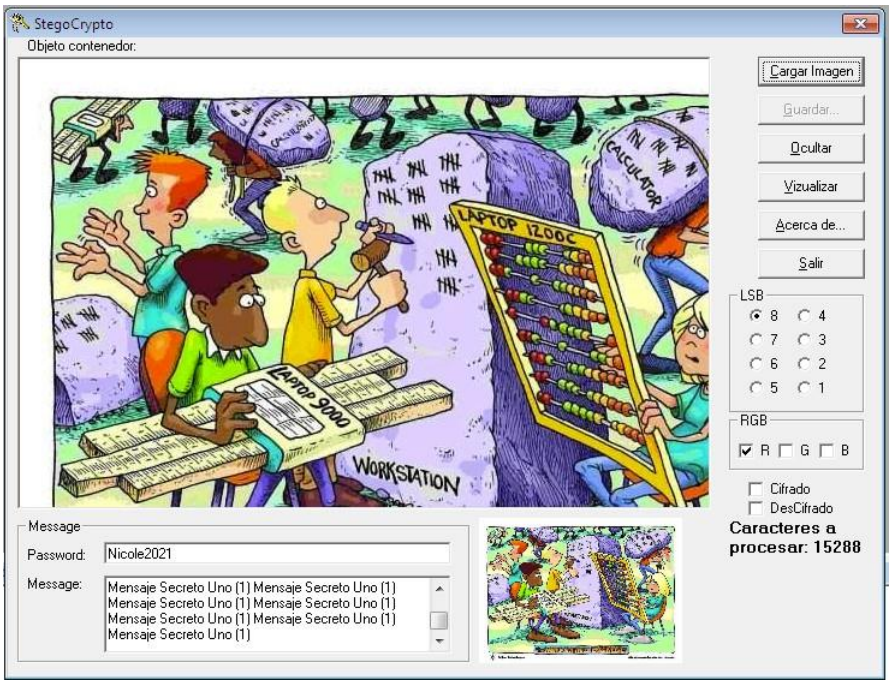
Figura 22 Ventana principal visualizando el objeto contenedor.



Fuente: El Autor.

La primera característica del prototipoII, es la opción Cifrado que nos permite por medio de la combinación de los métodos de transposición y sustitución dejar el mensaje secreto en ilegible. La base fundamental de este procedimiento es la función StrReverse que se encarga de convertir un texto legible de izquierda a derecha en invertirlo y que sea legible de derecha a izquierda. Luego con la tecnica de sustitución y con un algoritmo aleatorio cambia el valor de cada carácter gracias al operador Xor. El código del botón check “Cifrado” al evento de realizar click se describe en el Anexo II. La figura 23 visualiza la ventana principal de StegoCrypto con un mensaje secreto de más de quince mil (15000) caracteres.

Figura 23 Ventana principal con un mensaje de 15288 caracteres.

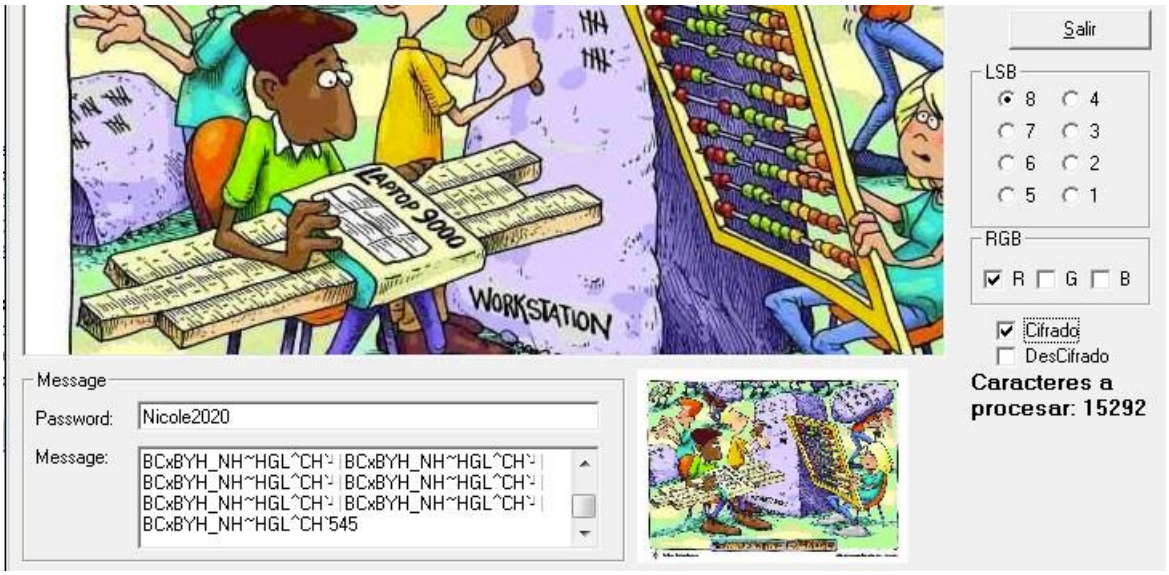


Fuente: El Autor.

En la figura 24, podemos observar el resultado de aplicar el proceso de cifrado sobre el mensaje secreto. Al final se obtiene un codigo 545 el cual es parte de la clave para el proceso inverso, es decir el Descifrado. El número se interpreta de derecha a izquierda, el valor cinco se le resta uno y divide por 2: (5-1) /2 = 2 que significa que los siguientes dos caracteres (54) son el Rand del proceso de sustitución. El valor correcto del Rand se obtiene aplicando la función StrReverse lo cual arrojará 45 y es la llave para proceder a aplicar el descifrado sobre el texto por medio del operador lógico Xor. Por último al texto se le aplica la función StrReverse para invertir la técnica de transposición y obtenemos el texto nuevamente legible, como se observa en la Figura 25. Es importante resaltar que esté procedimiento de cifrado ofreceré una mejor seguridad al mensaje secreto dentro del Estego Objeto para el proceso de Estegoanálisis que será otro objetivo a desarrollar dentro de este proyecto.

La segunda opción que nos ofrece este nuevo prototipo es la de seleccionar el color o colores de cada píxel en que se ocultará cada bit del mensaje secreto y como tercera opción encontramos que el usuario puede seleccionar el LSB en el cual se almacenará cada bit.

Figura 24 Ventana principal visualizando el mensaje secreto cifrado.



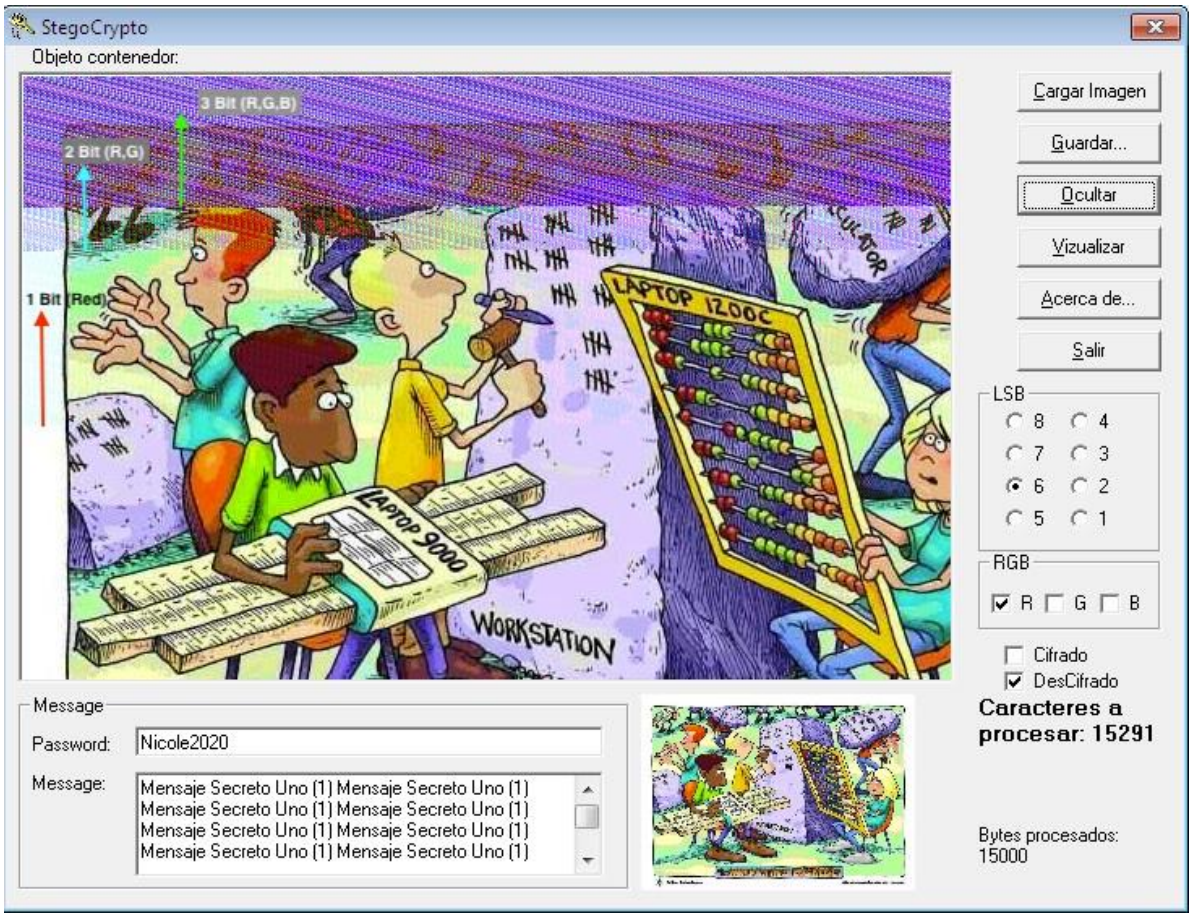
Fuente: El Autor.

Figura 25 Ventana principal visualizando el mensaje secreto cifrado.

Fuente: El Autor.

La figura 26 nos permite observar y concluir que al combinar los colores RGB menos pixeles de la imagen utilizamos, por consiguiente más información permitirá almacenar dentro de una Estego Objeto. Para nuestro ejemplo del mensaje secreto de 15288 caracteres al utilizar un sólo color (R), aproximadamente ocupa el 50% de los píxeles que componen la imagen contenedora. Si el usuario selecciona dos colores como por ejemplo (GB), el porcentaje de utilización será de un 25% y si seleccionamos utilizar los tres colores (RGB) la utilización de píxeles será tan sólo de un 17%. De manera ilustrativa utilizaremos el bit 8 para un solo color (Red), el bit 7 para dos colores (GB) y el bit 6 para los tres colores (RGB).

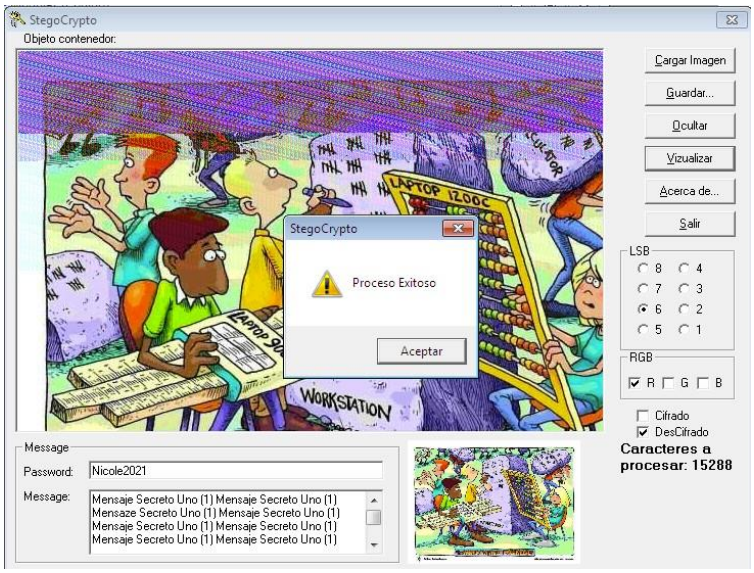
Figura 26 Ventana principal visualizando el objeto contenedor.



Fuente: El Autor.

Un interrogante que puede surgir en este momento es porqué con la utilización de los tres colores (R,G;B) el porcentaje es del 17% y no el 12%. La respuesta es que un byte (carácter) son 8 bit, y la función en dos píxeles almacena 6 bit y en el tercer píxel almacena 2 bit (7 y 8). El otro aporte importante que podemos deducir con este ejemplo es que en un mismo Estego Objeto puedo almacenar 3 mensajes secretos diferentes, claro está utilizando diferente bit en el LSB. La figura 27 se ilustra la visualización del mensaje secreto número 1, dónde se utilizó un solo color (Red) y se almacena en el LSB 6 con la llave de esteganografía (Nicole2021). Es importante resaltar que si el usuario no proporciona estos tres paramétros exactatos el software le informará en uncuadro de mensaje que no existe mensaje secreto. De lo anterior, nos permite concluir que al habilitar estos tres parámetros en la aplicación, ésta más segura o requerirá de mmayor tiempo y procesos para llegar a ser vulnerable.

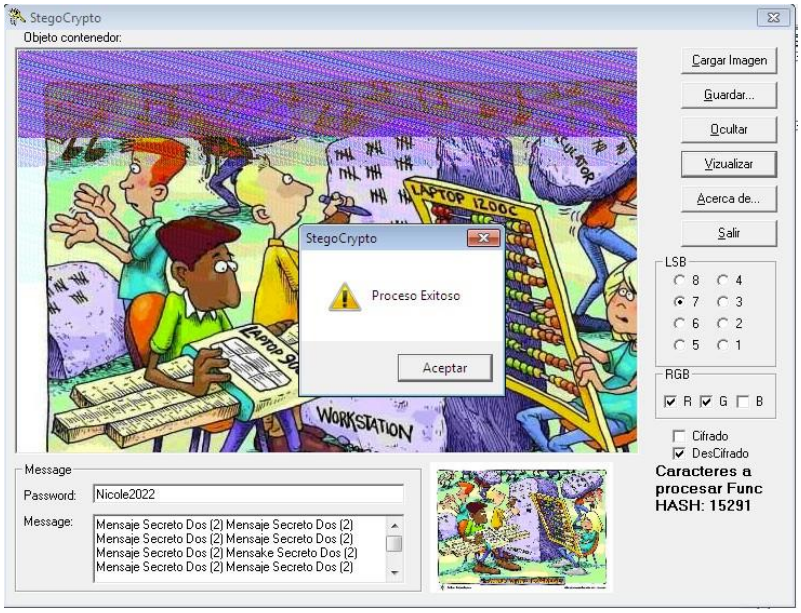
Figura 27 Ventana principal visualizando el primer mensaje secreto.



Fuente: El Autor.

La figura 28, ilustra la visualización del mensaje secreto número 2, dónde se utilizó dos colores (Red, Green) y se almacena en el LSB 7 con la llave de esteganografía (Nicole2022).

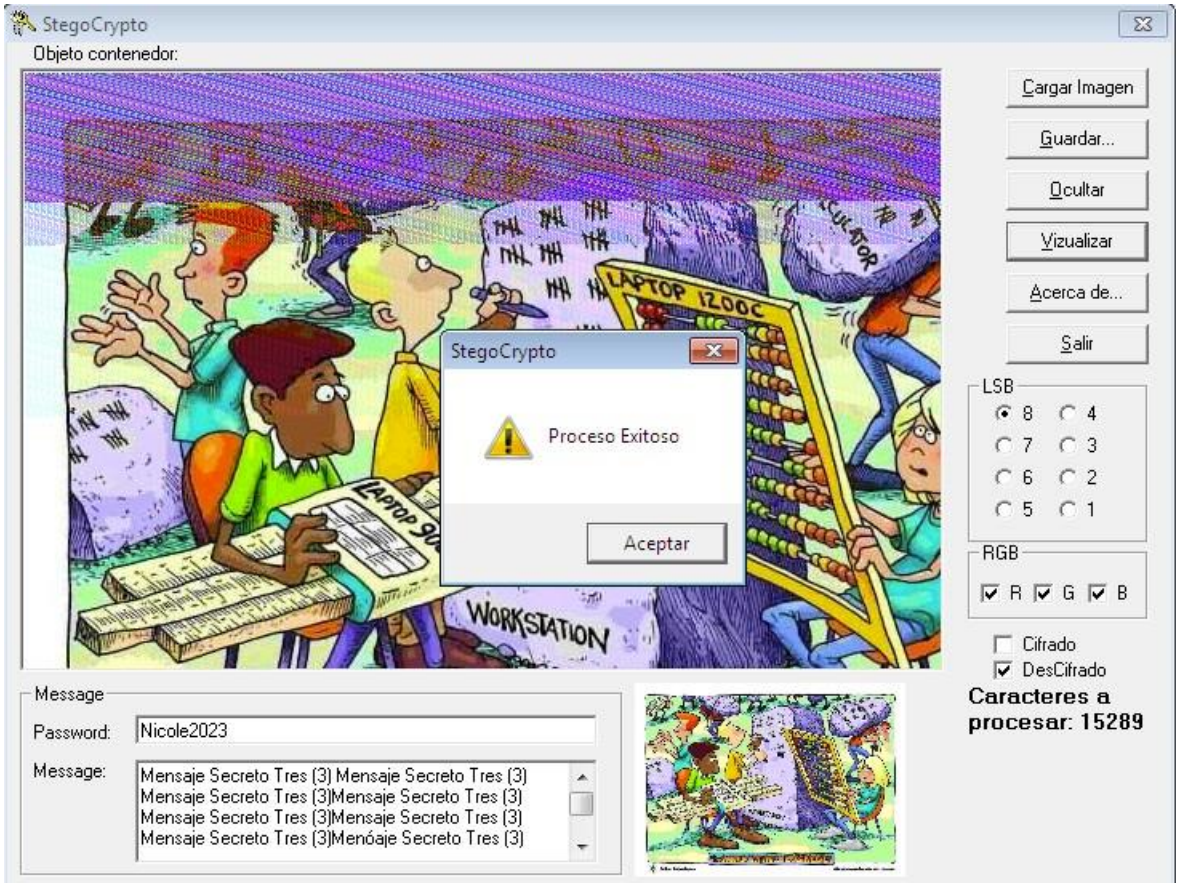
Figura 28 Ventana principal visualizando el segundo mensaje secreto.



Fuente: El Autor.

La figura 29, ilustra la visualización del mensaje secreto número 3, dónde se utilizó los tres colores (Red, Green, Blue) y se almacena en el LSB 8 con la llave de esteganografía (Nicole2023).

Figura 29 Ventana principal visualizando el tercer mensaje secreto.

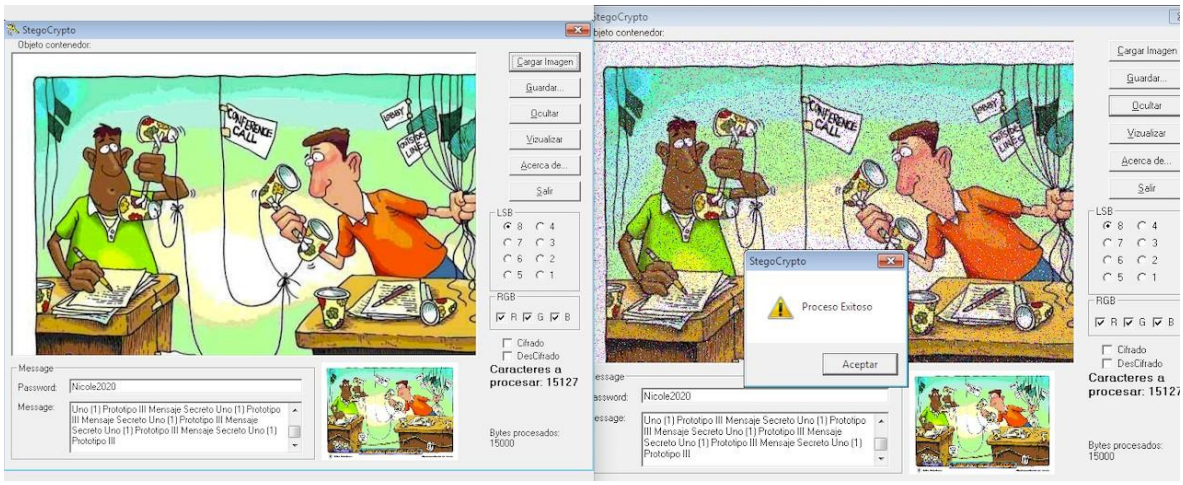


Fuente: El Autor.

7.1.3. StegoCrypto – Prototipo III

Con la codificación del tercer prototipo, la finalidad es mejorar la técnica del LSB. En el prototipo II, el almacenamiento del mensaje secreto se realiza de manera secuencial en un número constante de desplazamiento horizontal (X) y luego vertical (Y). Con este prototipo, se modifica su almacenamiento por aleatorio fundamentado en el HASH que arroja la contraseña de ocultamiento. El valor Hash, se utiliza como semilla inicial (S0), para iniciar la secuencia de números aleatorios, la cual será la misma para visualizar el mensaje secreto. La figura 30 visualiza la ventana principal de StegoCrypto con un mensaje secreto de más de quince mil (15000) caracteres, el mismo mensaje utilizado como ejemplo en el prototipo II.

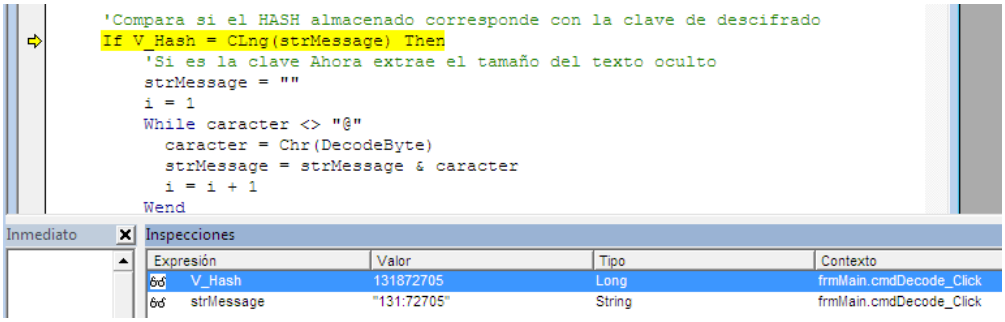
Figura 30 Ventana principal ocultando un mensaje secreto.



Fuente: El Autor.

En la ventana izquierda de la figura 30, se observa el objeto contenedor y el mensaje secreto a ocultar con más de 15000 caracteres. La ventana del lado derecho se observa la gran cantidad de puntos distribuidos alrededor de toda la imagen, lo cual presenta un inconveniente con la función Rnd, la cual sin importar la semilla el período de números aleatorios generados es pequeño a la cantidad de bit que se pretenden ocultar, lo que trae como consecuencia que un número significativo de bits sean sobre escritos por otros. Para este ejemplo particular se realizó seguimiento paso a paso al código fuente y se encontró que al sobre escribir en los bits se pierde valores críticos en la estructura como es el tamaño del HASH y el valor de éste, los cuales son los valores fundamentales para determinar la relación de la contraseña y la secuencia de almacenamiento de los bits que componen el mensaje secreto. La figura 31, visualiza el seguimiento paso a paso del código fuente.

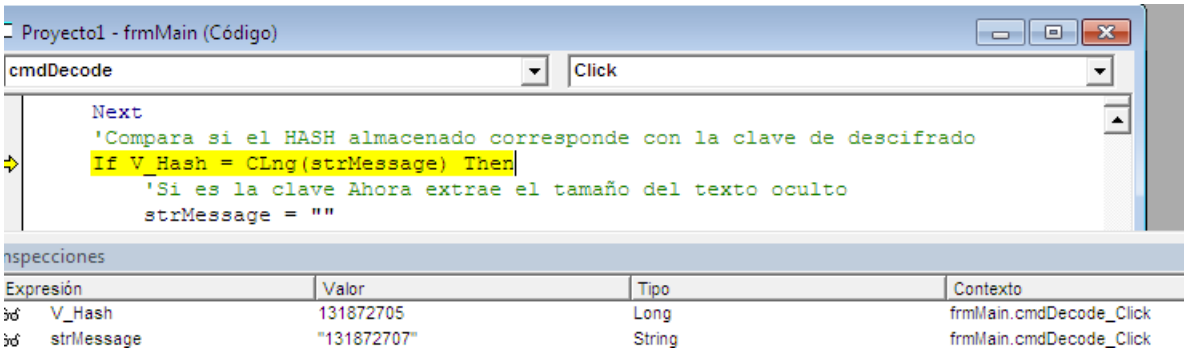
Figura 31 Inspección paso a paso código fuente - mensaje secreto (15000).



Fuente: El Autor.

Como se aprecia en la figura anterior, el valor del HASH en su cuarta posición es un ocho(8) y por sobre posición del bit pasa a ser el carácter dos puntos (:), afectando el número y por consiguiente el programa informa al usuario que “Contraseña incorrecta”. Al volver en realizar el intento con 1000 caracteres de tamaño del mensaje secreto en la depuración paso a paso se observa que el último digito del valor Hash es alterado, como se ilustra en la figura 32.

Figura 32 Inspección paso a paso código fuente - mensaje secreto (1000).



Fuente: El Autor.

Luego de varios intentos y seguimiento de inspección del código fuente se logra obtener que el código funciona con menos de 500 caracteres, pero parte del mensaje secreto es alterado en algunos caracteres, como se observa en la figura 33.

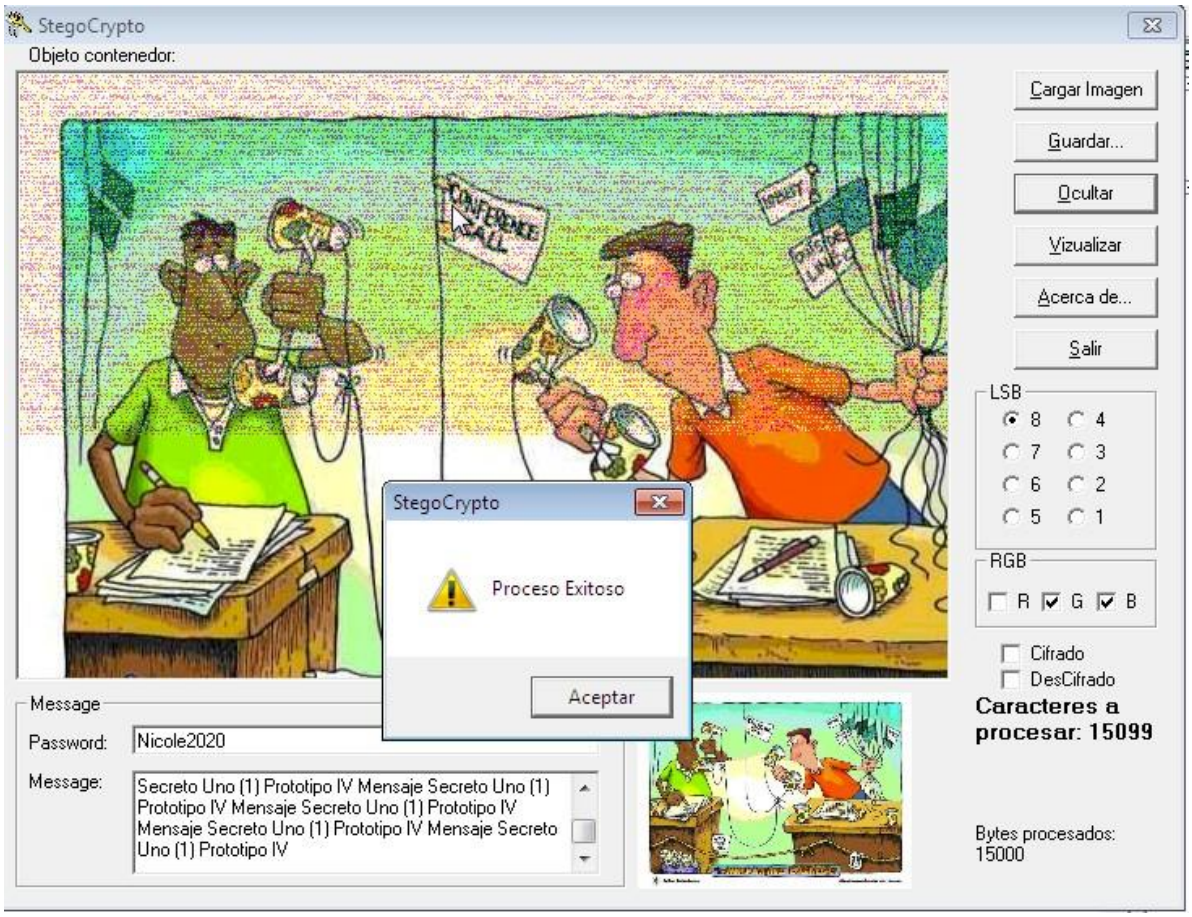
Figura 33 Ventana principal visualizando el mensaje secreto con pequeñas alteraciones.

Fuente: El Autor.

7.1.4. StegoCrypto – Prototipo IV

Dado que los resultados del tercer prototipo no fueron tan eficientes con respecto al segundo prototipo; se procede a crear el cuarto y último prototipo, el cual consiste en un híbrido entre estos dos, es decir, un almacenamiento seudo - aleatorio. La figura 34 visualiza el resultado de ocultar un mensaje secreto de 15099 caracteres en el LSB (8) con los colores GB de cada Píxel.

Figura 34 Prototipo IV - Ventana principal proceso ocultando mensaje.



Fuente: El Autor.

El código de este prototipo se encuentra en el Anexo 4. Este prototipo nos garantiza seguridad dado que aunque el almacenamiento de los bits se realiza en un barrido horizontal a crecimiento vertical la posición X,Y no es constante dado a que un algoritmo aleatorio con base en el valor Hash determina al azar su siguiente posición horizontal. Finalmente, el porcentaje de almacenamiento con respecto al segundo prototipo es menor pero su nivel seguridad contra estegoanálisis es mayor.

7.2. DESARROLLO OBJETIVO 2. Identificar estrategías con el uso de técnicas esteganográficas para la protección de datos en sus tres pilares: Confidencialidad, integridad y disponibilidad.

Para garantizar la protección de la información en sus tres pilares, quiero citar la siguiente frase:

"***Un secreto que es conocido por más de uno, ya no es un secreto***"
Sun Tzu

A diferencia de uno de los principios de Kerkhoff formulados a inicios del siglo XX, para garantizar los pilares de la información dice: "asume que el delincuete informático conoce todos los procedimientos de cifrado". Si se llegaré a aplicar este principio en la esteganografía, se asumiría que el delincuente informático conoce cuando intercepta una imagen digital identifica si es un Estego-objeto, lo cual afectaría los principios de integridad

Por lo anterior, para garantizar la integridad y confidencialidad del mensaje oculto dentro de un objeto, los expertos en seguridad informática han optado por combinar la criptografía con la estegonagrafía De esta forma si el delincuente informático identifica ya sea por estegoanálisis manual o estadístico que una imagen tiene un mensaje oculto, será imposible conocer el mensaje porque no podrá descifrarlo a menos que conozca el password de desencripción.

Finalmente, con el software desarrollado en este proyecto de grado (StegoCrypto) se garantiza al usuario la disponibilidad e integridad de la información, gracias al algoritmo Hash que se utiliza con la clave para ocultar los datos. Esta clave al procesarla en el algoritmo Hash genera un número único de más de 10 dígitos el cual es la semilla para encontrar la secuencia de píxeles en el prototipo IV. Otro aspecto del software es que puedo guardar la información no necesariamente en el LSB, puede ser en el segundo o tercer bit LSB y variando en la combinación de colores (RGB), en un sólo color, en dos colores o cada bit en cada uno de los RGB. Lo anterior son aspectos que hacen más complejo el proceso de stegoanálisis para imágenes con mensajes ocultos con el algoritmo LSB.

7.3. DESARROLLO OBJETIVO 3. Investigar herramientas de estegoanálisis, determinando la más efectiva para las pruebas de información oculta con "StegoCrypto".

Para el estegoanálisis se utilizarán las siguientes herramientas:

- StegSecret

StegSecret, es el nombre que recibe el proyecto de software libre que tiene la intención de desarrollar y mantener un conjunto de herramientas libres que permitan la detección de información esteganografiada en diferentes medios de información.

Su objetivo principal consiste en recopilar, implementar y facilitar el uso de los numerosos estudios estegoanalíticos que faciliten la detección de información ocultada en diferentes medios, especialmente medios digitales, como imágenes, audio y video. Este proyecto pretende alertar sobre la inseguridad que presenta la utilización de multitud de herramientas y algoritmos esteganográficos, así como el uso indebido de ciertas técnicas más seguras.

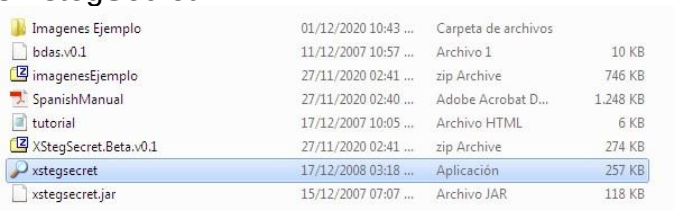
Actualmente, el proyecto StegSecret está constituido por dos herramientas. StegSecret una herramienta en modo consola que facilita la automatización de la detección en diferentes fuentes de datos (próximamente) y Xstegsecret una herramienta visual.

Actualmente las herramientas desarrolladas en este proyecto están realizadas en JAVA JDK1.5 (multiplataforma) y están liberadas bajo licencia GPL.

Para la instalación de Xstegsecret se debe:

- Bajar en un mismo directorio los ficheros, xstegsecret.jar (o su versión .exe), BDAS v.0.1 y tutorial.html, como se visualiza en la figura 35.

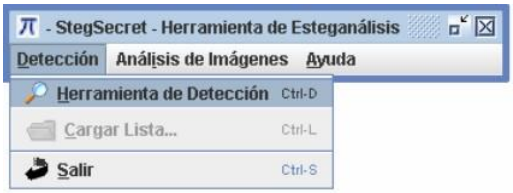
Figura 35 Archivos XstegSecret.



Fuente: El Autor.

- Ejecutar la herramienta. #java –jar xstegsecret.jar o #xstegsecret.exe

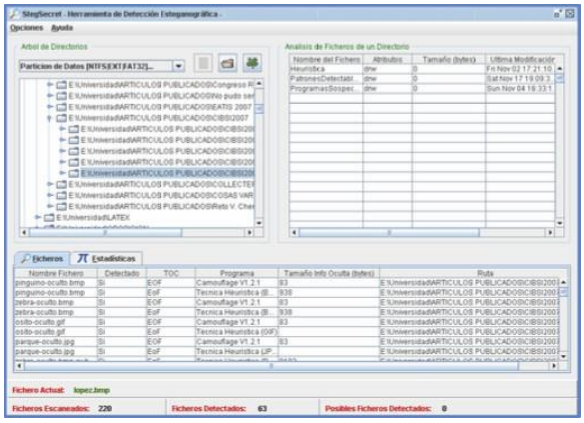
Figura 36 Ejecución de la herramienta xstegsecret.exe.



Fuente: El Autor.

Al seleccionar la opción “Herramienta de Detección” que tiene el icono Lupa, se inicia el escaneo automático a partir de la ruta indicada en el árbol. Puede elegir una ruta específica utilizando el icono “Carpeta”, para esto se debe hacer click en un fichero del directorio al cual se quiere acceder.

Figura 37 Ventana Herramienta de Detección.



Fuente: XstegSecret.

Trébol de cuatro hojas: Este icono indica que el proceso de escaneo está en ejecución. Para detener el proceso se debe realizar click sobre este icono.

Figura 38 Icono Trébol para detener el escaneo.

Fuente: XstegSecret.

Una vez terminado el escaneo de la carpeta con sus respectivas subcarpetas, los resultados viene todo tipo de información sobre el fichero donde se ha encontrado información oculta. Por ejemplo, Técnica EOF= End of File (informa de información oculta al final de fichero), LSB (información oculta en el LSB), PoS (presencia de programas sospechosos).

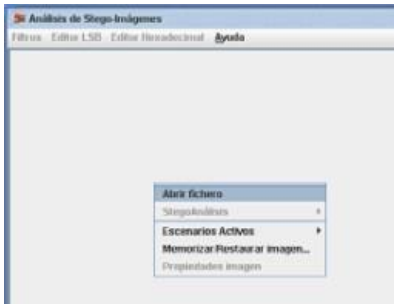
Figura 39 Resultado del escaneo.

Fuente: XstegSecret.

Como se puede observar en esta versión, la utilización de XStegsecret es muy sencilla, todo se reduce a pulsar en un botón y escanear todas las técnicas conocidas. En futuras versiones, se dará la opción de elegir que técnicas concreta de detección quiere el usuario utilizar.

La otra opción que ofrece esta herramienta es el análisis de imágenes, está formada por 2 escenarios (izquierda y derecha), con lo que es posible operar con 2 imágenes a la vez si se desea. Lo cual, es muy útil. Pulsando el botón derecho del ratón se ven las operaciones que se pueden realizar sobre cada escenario.

Figura 40 Ventana Análisis de Stego - Imágenes.



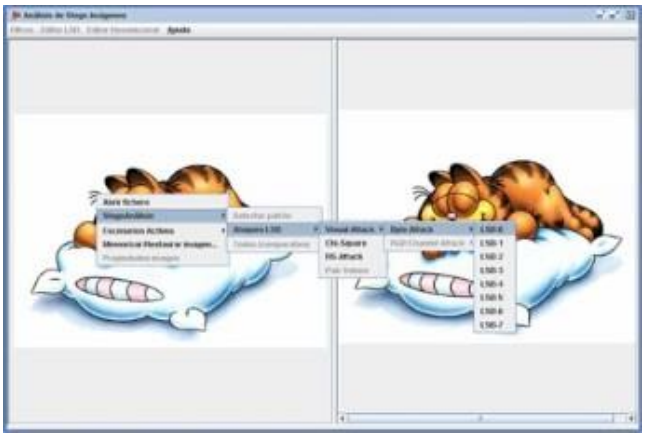
Fuente: El Autor.

Al pulsar “Abrir Fichero” se puede abrir una imagen en un escenario, actualmente sólo se puede aplicar operaciones de estegoanálisis sobre mapas de bits (BMP). Al cargar una imagen esta quedará memorizada, si se realiza alguna operación sobre la imagen en un escenario y quiere recuperar la imagen original, pulse “Memorizar/Restaurar Imagen”

Al realizar click derecho sobre la imagen, se visualiza una ventana emergente con las opciones para realizar Estegoanalisis ofreciendo tres técnicas:

- ✍ Ataque visual. El la técnica del LSB, el usuario seleciona entre cero (0) a siete (7) el bit que se desea analizar.
- ✍ ChiSquare (Chi-Cuadrado)
- ✍ RS Attack

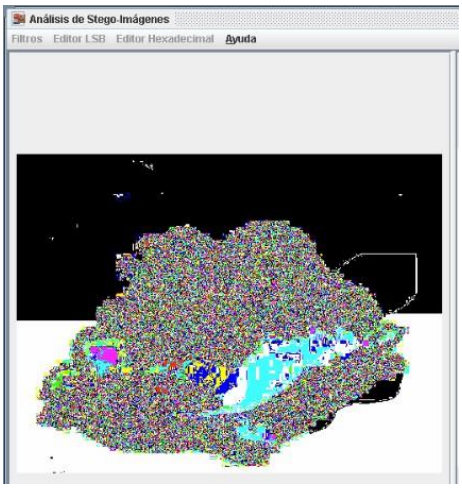
Figura 41 Estegoanálisis técnica por Ataque Visual LSB-0.



Fuente: XstegSecret.

La figura 42, visualiza el resultado del Ataque Visual con el algoritmo del bit menos significativo en su posición cero (LSB-0). Como se observa, la técnica por Ataque Visual representa en un área uniformidad representando un conjunto de bits que tiene información oculta.

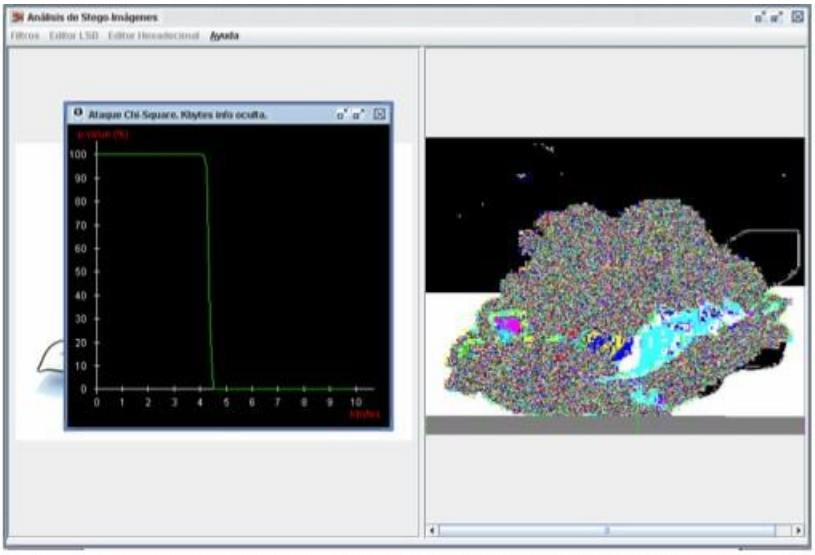
Figura 42 Resultado Ataque Visual LSB-0.



Fuente: XstegSecret.

Finalmente la figura 43, muestra resultados exitosos de información oculta con la utilización de la técnica ChiSquare en un Estego Objeto que utilizó la técnica de LSb secuencial.

Figura 43 Resultado Ataque Visual LSB-0 y ChiSquare.



Ataque Visual (LSB0) y ChiSquare a imagen con 4KB de info oculta por LSB secuencial.

Fuente: XstegSecret.

7.4. DESARROLLO OBJETIVO 4. Identificar posibles vulnerabilidades en el software de esteganografía "*StegoCrypto*" por medio de una herramienta de estegoanálisis.

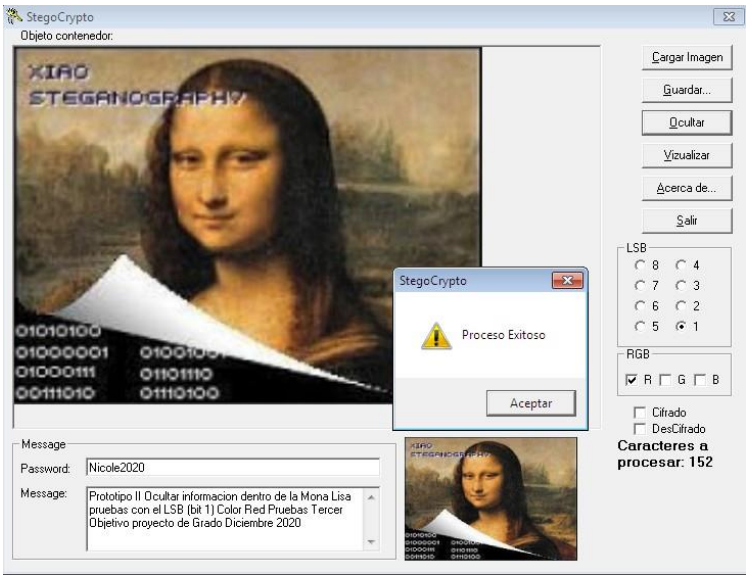
Se procede a identificar posibles vulnerabilidades con la utilización de los prototipos II – III y IV de StegoCrypo y el programa GLPI OpenPuff

- StegCrypto – Prototipo II

Para todas las pruebas utilizaremos la mima imagen (monalisa.jpeg), para este prototipo ocultaremos información en el LSB (1,2,3) y se volverá a ocultar la misma información en esos bits, pero cifrada, claro está generando archivos independientes. Finalmente se combinará ocultar información utilizando un sólo color, dos colores y tres colores (RGB).

La figura 44, visualiza el proceso exitoso de ocultar el mensaje a transmitir en el LSB (bit 1) y el color del bit utilizado Red.

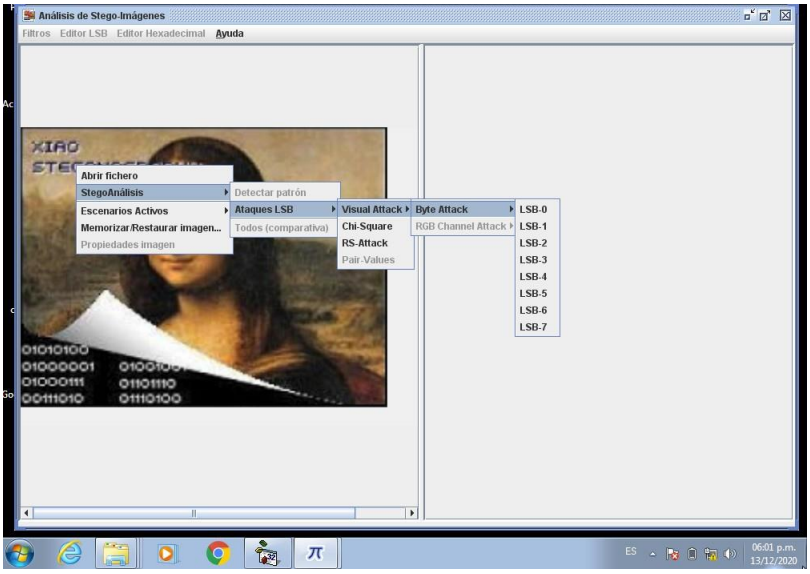
Figura 44 Prototipo II – Ocultando en el LSB (bit 1) color Red.



Fuente: StegoCrypto – Prototipo II.

Con el software de estegoanálisis “XstegSecret”, procedemos en tratar de identificar información oculta en el archivo generado de la figura 44. Es importante enunciar que para la ejecución de XstegSecret se requiere instalar Java Versión 8 Update 151 o una versión superior.

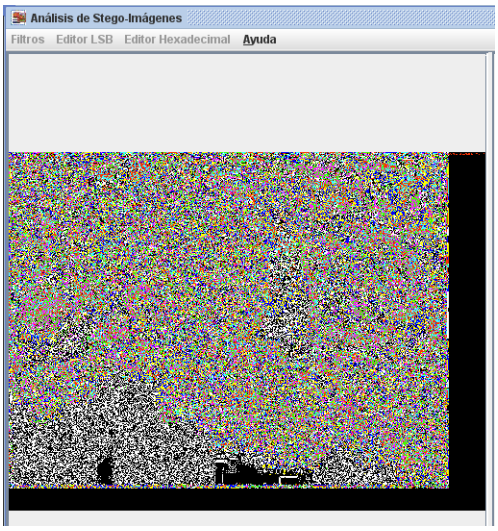
Figura 45 XstegSecret– Ataque LSB - 0.



Fuente: El Autor.

La figura 46, ilustra el resultado del Ataque LSB-0. Como se observa dado que es una cantidad insignificativa de caracteres ocultos, visualmente es casi imposible identificar que existe un conjunto de píxeles que tienen diferencia con respecto a los demás.

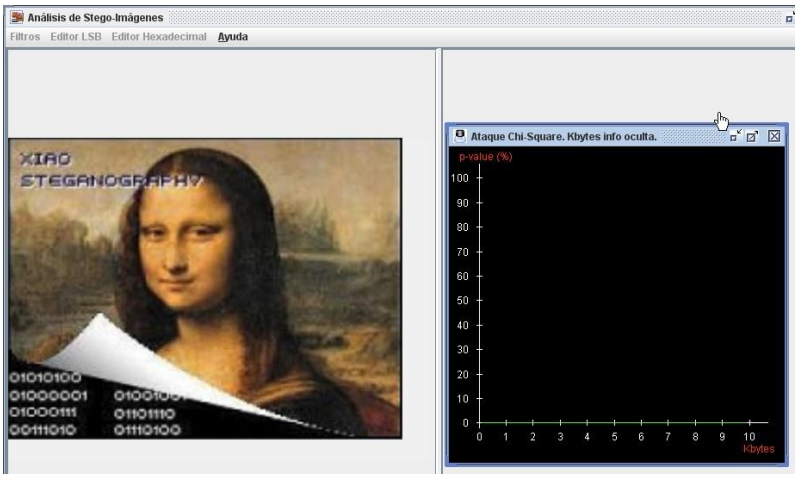
Figura 46 Resultado Ataque LSB – 0 con XstegSecret.



Fuente: El Autor.

La herramienta XstegSecret ofrece un segundo tipo de ataque cuando es muy difícil identificar si existiere información oculta por ataques de LSB-X; este tipo de ataque se denomina Chi-Square (Chi-Cuadrado). La figura 47 ilusta el resultado.

Figura 47 Resultado Ataque Chi Cuadrado con XstegSecret.



Fuente: XstegSecret.

Con el algotitmo RS (Jessica Fridrich), se procede a realizar el ataque RS, arrojando los siguientes resultados: Ocupación del canal Rojo de 0,64977% Ocupación del canal Verde y Azul de 0,02% y 0,04 % respectivamente. Ocupación total posible 0,24% para un tamaño aproximado de información oculta de 193 Bytes. El resultado es casí 100 % efectivo dado que con la utilización de StegoCrypto se oculto un mensaje de 151 caracteres como se aprecia en la figura 44. La figura 48 nos visualiza el resultado del ataque RS.

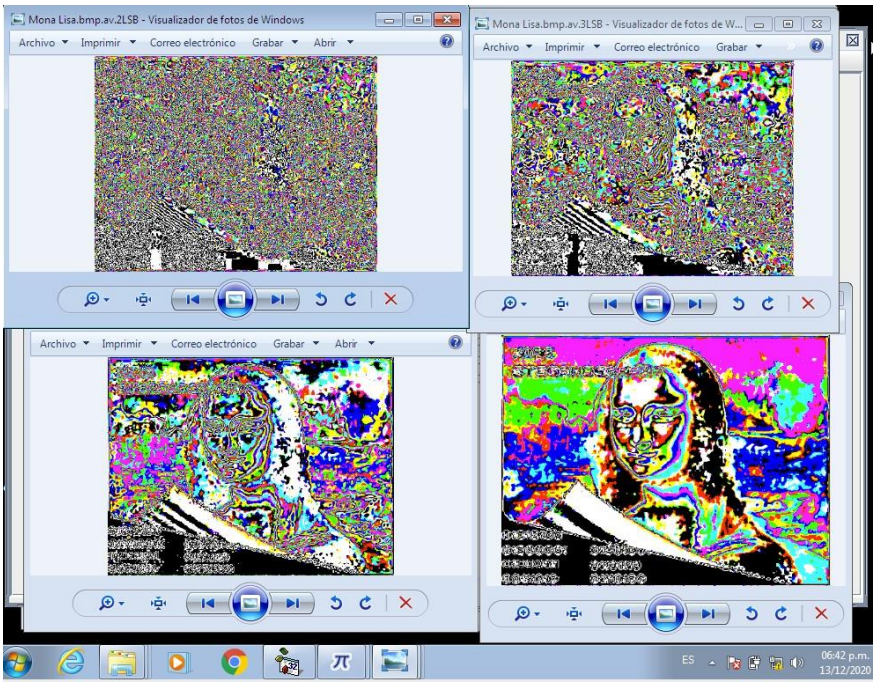
Figura 48 Resultado AtaqueRS con XstegSecret.

Fuente: https://es.slideshare.net/erojaso/dlp-stegano-pi

Una vez realizado los tres tipos de ataques con XstegSecret a nuestro Estego Objeto, podemos deducir que los ataques LSB-X, no son efectivos dado el número insignificativo de caracteres ocultos. En cuanto al ataque Chi-Cuadrado el método implementado de LSB en los prototipos de StegoCrypto son seguros y finalmente el ataque RS es efectivo y preciso.

Ahora procedemos a realizar los mismo tres ataques a la imagen Mona Lisa tomada de Internet. Para el caso de los ataques visuales se encuentran unas sombras en la parte superior derecha con los ataques LSB-2, LSB-3, LSB-4 y LSB-5. La figura 49 ilustra en conjunto los resultados mencionados. Con este ejercicio, podemos deducir que los ataques LSB-X son efectivos cuando el usuario utiliza una herramienta de esteganografía y oculta información utilizando más de un canal simultaneamente como: (Red-Green), (Red- Blue), (Green-Blue) o quizá más efectivo cuando se utilizan los tres canales simultaneamente (Red-Green-Blue).

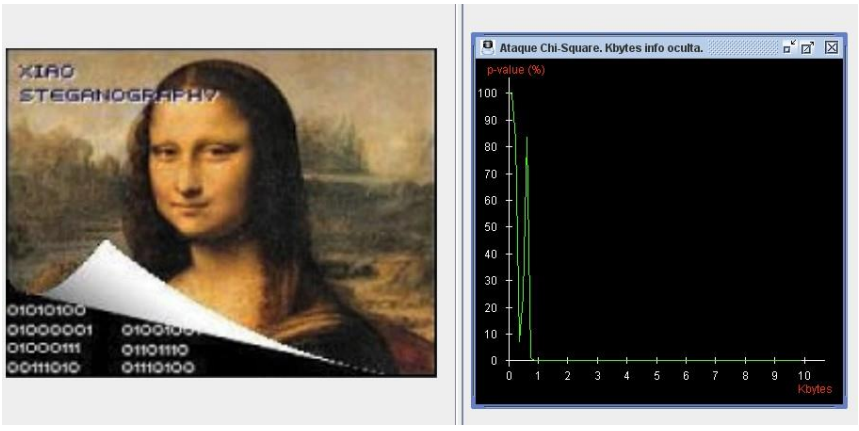
Figura 49 Consolidado Ataque LSB (2,3,4 y 5) con XstegSecret.



Fuente: El Autor.

Procedemos a confirmas la hipótesis de posible información oculta en la imagen “Mona lisa” descargada por internet por medio del ataque de Chi – Cuadrado. En la figura 50 observamos que el cuadro cartesiano nos muestra que aproximandamente en 1000 bytes de la imagen se encuentra información oculta.

Figura 50 Mona Lisa descarga de Internet Ataque Chi-Cuadrado con XstegSecret.



Fuente: XstegSecret.

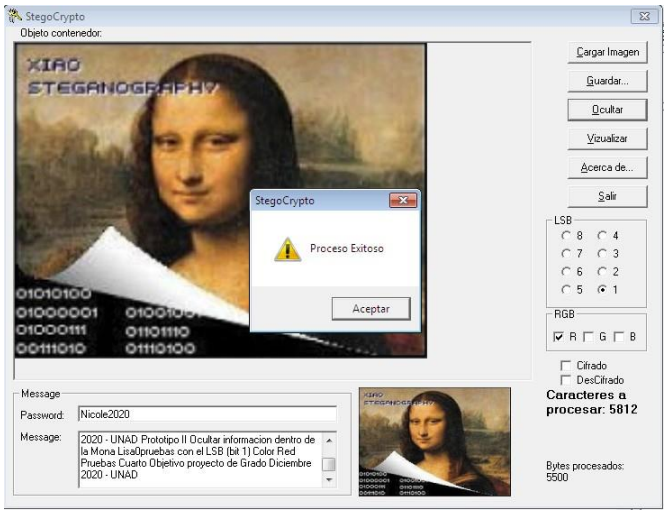
Finalmente, el ataque RS a la Mona Lisa descargada por internet nos confirma que en los tres canales tiene información oculta y un tamaño aproximado de 287 caracteres. Los datos se pueden apreciar en la figura 51:

Figura 51 Mona Lisa descarga de Internet Ataque RS con XstegSecret.

Fuente: https://es.slideshare.net/erojaso/dlp-stegano-pi

Con la finalidad de obtener resultados fiables, realizaremos otra prueba con la misma imagen portadora. Con el prototipo II de StegoCrypto Ocultaremos un número significativos de caracteres en el mensaje secreto (5812 caracteres). Como se observa en la siguiente imagen

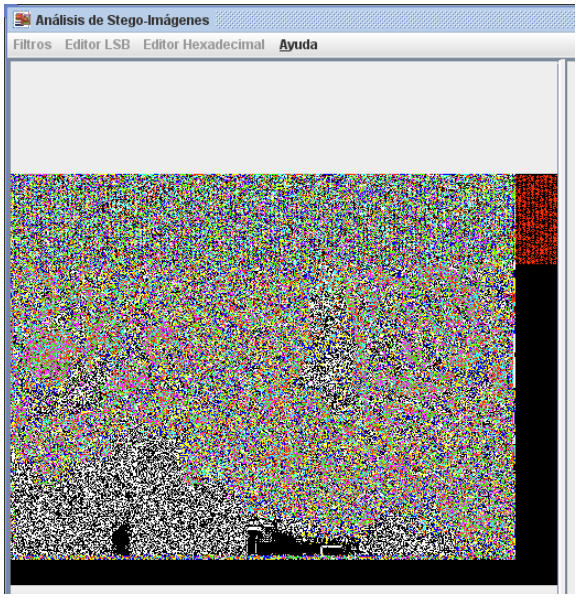
Figura 52 Ocultando 5812 caracteres con el Prototipo II- StegoCrypto.



Fuente: StegoCrypto – Prototipo II.

Al realizar el ataque LSB-0, se obtienen resultados positivos dado a que StegoCrypto se ocultaron los caracteres secuencialmente y utilizó parte del marco que crea para guardar el Estego Objeto. En la figura 53 podemos observar la parte superior como varía los colores de los píxeles en un verde un poco desapercibido; pero en la parte superior derecha nos muestra que hay información oculta, utilizando el canal Red.

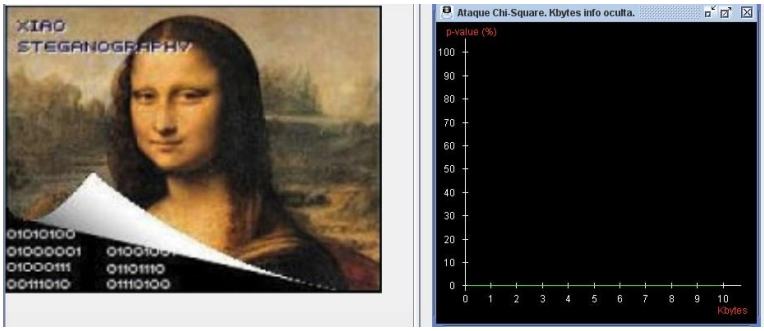
Figura 53 Ataque LSB-0 al archivo MonaLisaBit1red5812caracteres.jpeg.



Fuente: XstegSecret.

El ataque Chi-Cuadrado al objeto contenedor obtenido en la figura 52, reporta que no se encuentra información oculta. La figura 54 visualiza la gráfica cartesiana del ataque Chi-cuadrado.

Figura 54 Ataque Chi_Cuadrado al archivo MonaLisaBit1red5812caracteres.jpeg.



Fuente: XstegSecret.

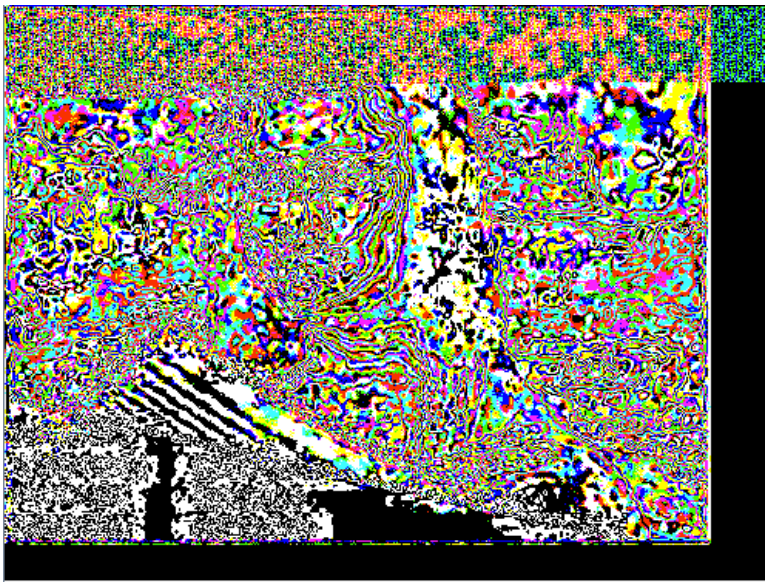
El ataque RS al Estego Objeto de la figura 52, es preciso y contundente, el reporte del ataque informa que la información se encuentra oculta por el canal Rojo en un 15% y que el tamaño aproximado de información oculta es de 4268 bytes.

Figura 55 Ataque RS al archivo MonaLisaBit1red5812caracteres.jpeg.

Fuente: https://es.slideshare.net/erojaso/dlp-stegano-pi

Otra prueba con la herramienta XstegSecret, es un ataque LSB-3 a un Estego Objeto que ocultó información en el bit 4 y los canales GB (Verde-Azul) El resultado es notorio visualmente, la figura 55 muestra los resultados.

Figura 55 Ataque LSB-3 al archivo MonaLisaBit3RG7200caracteres.jpeg.



Fuente: XstegSecret.

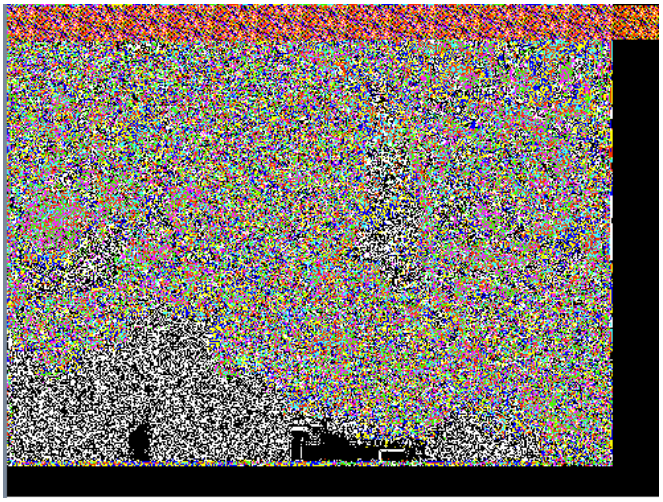
Con el ataque RS, se puede concluir su efectividad al arrojar datos casi perfectos, informando que por los tres canales hay información oculta y dando un tamaño aproximado de información oculta de 5021 bytes

Figura 56 Ataque RS al archivo MonaLisaBit1RG7222caracteres.jpeg.

Fuente: https://es.slideshare.net/erojaso/dlp-stegano-pi.

Otras pruebas que se realizaron fueron con Estego Objetos ocultando información cifrada. Los resultados fueron exitosos con el Ataque LSB-1

Figura 57 Ataque LSB-0 al archivo MonaLisaBit1RGB5200caracterCifrado.jpeg.

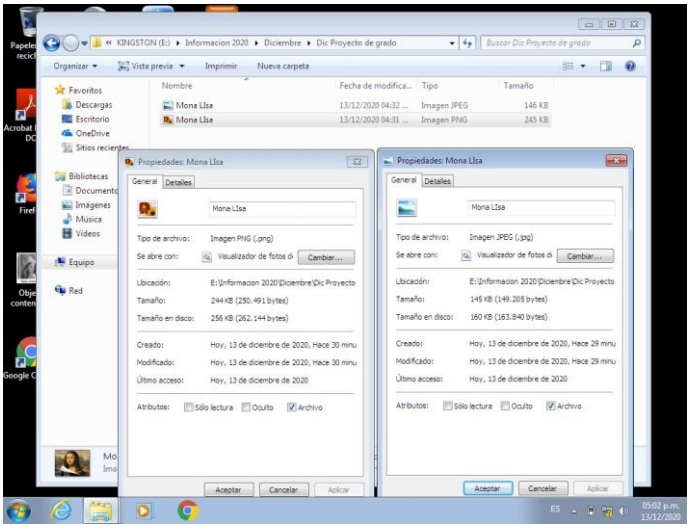


Fuente: XstegSecret.

7.5. DESARROLLO OBJETIVO 5. Generar un informe técnico mencionando las vulnerabilidades encontradas y recomendaciones para reducir el riesgo.

La imagen “mona lisa”, utilizada para las pruebas del cumplimiento del cuarto objetivo nos permite deducir que en una misma imagen varía su tamaño de almacenamiento dependiendo de la extensión asignada. Para este caso específico su tamaño es de 145 KB par extensión Jpeg y de 244 KB para extensión PNG. La figura 58 ilustra los diferentes tamaños de la imagen utilizada en la realización del tercer objetivo de este proyecto de grado.

Figura 58 Comparación del tamaño de una imagen digital con diferente extensión.



Fuente: El Autor.

Con la herramienta MS Paint, abrimos cualquiera de estas dos imágenes (Png o Jpeg) y las guardamos con extensión Gif y mapa de bits (BMP) La extensión Gif es la que de menor tamaño requiere en la imagen con 71 KB y el mapa de bit (BMP) es la de mayor tamaño con 471 KB, como se observa en la siguiente figura 59:

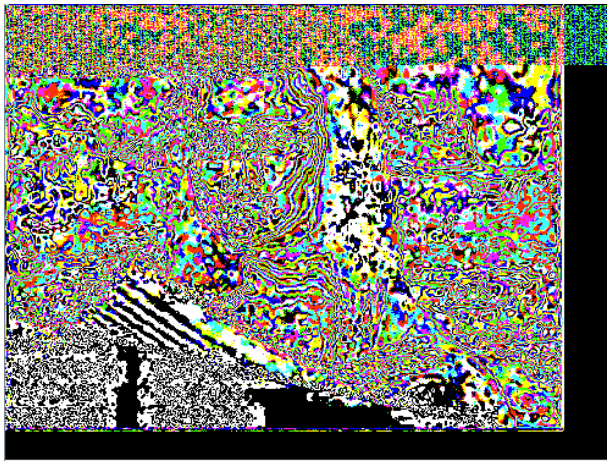
Figura 59 Comparación del tamaño de una imagen digital con diferentes extensiones.

Nombre	Fecha de modifica...	Tipo	Tamaño
Mona Lisa	13/12/2020 04:32 ...	Imagen JPEG	146 KB
Mona Lisa	13/12/2020 04:31 ...	Imagen PNG	245 KB
Mona Lisa	13/12/2020 05:16 ...	Imagen GIF	71 KB
Mona Lisa	13/12/2020 05:21 ...	Imagen de mapa ...	471 KB

Fuente: El Autor.

La herramienta XstegSecret, con los ataques visuales como por ejemplo el ataque LSB-3 a un Estego Objeto que ocultó información en el bit 4 y los canales GB (Verde-Azul) El resultado es notorio visualmente, la figura 60 muestra los resultados.

Figura 60 Ataque LSB-3 al archivo MonaLisaBit3RG7200caracteres.jpeg.



Fuente: XstegSecret.

Los estego Objetos son seguros al utilizar un sólo canal y el prototipo III almacenando la información aleatoriamente.

Con el ataque RS, se puede concluir su efectividad al arrojar datos casi perfectos, informando que por los tres canales hay información oculta y dando un tamaño aproximado de información oculta de 5021 bytes

Figura 61 Ataque RS al archivo MonaLisaBit1RG7222caracteres.jpeg.

Fuente: https://es.slideshare.net/erojaso/dlp-stegano-pi

8. CONCLUSIONES

- Para reducir vulnerabilidades en los estego-objetos (mesajes ocultos) en cuanto a su confidencialidad e integridad, es necesario combinar la esteganografía con la criptografía. Para ello el mensaje a ocultar se debe cifrar y luego introducir en el objeto contenedor. De esta forma, aunque el delincuente informático descubra el patrón esteganográfico, jamás puede llegar a conocer el significado del mensaje encubierto. Adicionalmente, al combinar estás dos áreas en la mayoría de casos pasará inadvertida la información oculta.

- Se lográ establecer tres prototipos de Estegoanálisis, uno secuencial, otro aleatorio y un tercer prototipo semi-aleatorio. La clave de almacenamiento funciona como un Hash que sólo con descifrarlo se podría obtener la secuencia de los bits, lo cual es 99.9% imposible. De esta forma logramos garantizar la protección de los datos en sus tres pilares: Confidencialidad, integridad y disponibilidad.

- Una vez realizado los tres tipos de ataques con XstegSecret a nuestro Estego Objeto, podemos deducir que los ataques LSB-X, no son efectivos dado el número insignificativo de caracteres ocultos. En cuanto al ataque Chi-Cuadrado el método implementado de LSB en los prototipos de StegoCrypto son seguros y finalmente el ataque RS es efectivo y preciso para los prototipos II y IV.

- La técnica LSB con la secuencia aleatoria es la más segura ante ataques de Estegoanálisis (Ataques LSB, Chi-Square y RS), la desventaja es que permite un número muy pequeño de caracteres a ocultar en comparación con los prototipos secuancial y seudo-aleatorio.

- Aunque en algunos tipos de ataques se puede observar que existe información oculta, las herramientas de estegonagrafía aún así son seguras gracias a que la información sólo se puede decodificar si se conociera el valor Hash, el método de descifrado del texto, el bit LSB utilizado y el canal utilizado.

9. RECOMENDACIONES

- Almacenar información con el prototipo III, el cual almacena la cadena de bits que componen el mensaje secreto en forma aleatoria es la técnica más segura ante ataques conocidos como: Ataque LSB, Chi-Square y RS.

- Que el documento sea base para otros futuros trabajos utilizando objetos contenedores como archivos de audio y vídeo con su respectivo estegoanálisis.

10. DIVULGACIÓN

El desarrollo del presente proyecto de grado será dado a conocer en colaboración de la biblioteca de la Universidad Nacional Abierta y a Distancia - UNAD, a través de su aplicativo en línea, en donde se publicará un archivo PDF correspondiente al documento final presentado ante los jurados, posterior a la sustentación del mismo; con el fin de que todos los estudiantes de la Universidad que se encuentren interesados en el tema de Estegoanálisis, puedan acceder al documento.

11. BIBLIOGRAFÍA

Abbas Cheddad, Joan Condell. Kevin Curran, Paul McKevitt. (junio de 2009). Digital image steganography: Survey and analysis of current methods. [En línea]. El Servier (Ed), Journal of signal Processing, 727-752 Recuperado de: http://www.sciencedirect.com/science/article/pi i/S0165168409003648

Apple. (Marzo de 2018). Fundamentos de esteganografía. [En línea] Recuperado de: https://itune.apple.com/co/app/steganographia/id537627497/?platform=ipad&preserveScrollPosition=true#plataform/ipad#

Caballero, Pino (1998). Técnicas Criptográficas. 2ª Edición. Editorial RA-MA.

Chess, B. West, J. (2007). *Secure Programming with static analysis*. 3.ª Edición. Editorial Addison-Wesley Software Security Series.

Cicada,U. (8 de noviembre de 2012). U*ncovering cicada*. [En línea] Recuperado de: http://uncovering-cicada.wikia.com/wiki/OutGuess.html

Díaz, G. Alzórriz. Sancristobal, E. (2014). Procesos y herramientas para la seguridad de redes. Madrid, ES UNED Universidad Nacional de Educación a Distancia. [En línea] Recuperado de: http://bibliotecavirtual.unad.edu.co:2077/lib/unadsp/detail.action?docID=10862475&p00=seguridad+redes

Díaz Orueta, Gabriel. Alzórriz, Ignacio. Sancristóbal R., Elio. Castro Gil, Manuel Alonso (marzo de 2014). Procesos y herramientas para la seguridad de redes. Madrid, ES UNED Universidad Nacional de Educación a Distancia., ISBN 978-84-362-6838-6 Entre Ríos. [En línea] Recuperado de: http://www.uned.es/publicaciones/

Escrivá G., Gema; Romero Serrano,Rosa Mª.; Ramada, David Jorge; Onrubia Pérez, Ramón (España, 2012). *Seguridad Informática*. 5.ª Edición. Editorial

MacMillan Profesional. Área computación, ISBN EDICIÓN ELECTRÓNICA: 978-84-15991-41-0.

FIRSTCLOUDIT. Cuaderno de notas del Observatorio. *Esteganografía el arte de ocultar información*. [www.pgarciab.firstcloudit.com]. Publicación del articulo en firstcloudit, marzo de 2010 [consultado febrero 24 de 2020]. Disponible en: https://pgarciab.firstcloudit.com/documentos/esteganografia.pdf

Frith, David. (2007). Steganography approaches, options, and implications. [En línea]. Ed. Elsevier Ltd, Network Security. Recuperado de: http://www.sciencedirect.com/science/article/pi i/S1353485807700715

GARCÍA, Roberto de Miguel. (España, 2009). *Criptografía clásica y moderna*. 2.ª Edición. Septem Ediciones. Área computación, ISBN 978-84-92536-98-6.

García, Roberto de Miguel. (2009). Criptografía clásica y moderna, España Septem Ediciones. [En línea] Recuperado de: http://bibliotecavirtual.unad.edu.co:2162/openurl?sid=EBSCO:edselb&genre=book&issn=edselb.D3658FCF&ISBN=9788492536986&volume=&issue=&date=&spage=&pages=&title=Criptograf%C3%ADa%20cl%C3%A1sica%20y%20moderna&atitle=Criptograf%C3%ADa%20cl%C3%A1sica%20y%20moderna&aulast=Garc%C3%ADa,%20Roberto%20de%20Miguel&id=DOI

Herramientas de esteganografía (Mayo 2004). [En línea] Recuperado de: http://www.cotse.com/tools/stega.html

Inforense. (13 de mayo de 2015). Blogspot. [En línea] Recuperado de: http://wstj-inforense.blogspot.com/2015/05/programas-para-esteganografia.htm

INTECO. (19 de mayo de 2016).E-Gov Esteganografía. [En línea] Recuperado de: http://www.egov.ufsc.br/portal/sites/default/files/esteganografia1.pdf

Intypedia. (2016). Information Security Encyclopedia, enciclopedia gratuita en vídeo sobre temas de seguridad informática (Criptografía- Esteganografía), con

la garantía de la red CRIPTORED y la universidad Politécnica de Madrid . [En línea] Recuperado de:
http://www.intypedia.com

LUCENA LÓPEZ, Manuel José. (España, 2010). *Criptografía y seguridad en computadores*. Versión 4-0.8.1. Ediciones Universidad de Jaén. Recuperado de: http://creativecommons.org/licenses/by-nc/2.5/es/legalcode.es

McClure, S. Scambray. Kurtz, G. (2010). Hackers 6 secretos y soluciones de seguridad en redes. México McGraw-Hill Interamericana. [En línea] Recuperado de:
http://bibliotecavirtual.unad.edu.co:2077/lib/unadsp/reader.action?ppg=1&docID=10433876&tm=1465509236690

Molina, Mateos. José. María. (mayo de 2000).Seguridad de la información: Criptología. E-libro.net [En línea] Recuperado de:
http://ebookcentral.proquest.com

Moreno, M. (abril de 2010). *Security ar twork*. [En línea] Recuperado de:
https://www.securityartwork.es/2010/04/15/introduccion-a-la-esteganografia-i.pdf

Muñoz M, Alfonso. (México, 2014). *Esteganografía: Privacidad y ocultación de información digital. Protegiendo y atacando redes informáticas*. 2.ª Edición. Editorial RA-MA. Área computación, ISBN 978-607-32-0817-8.

Navas, G. Sergio. Rodríguez Medina, Gustavo. (2008). Esteganografía simulada para análisis de efectos portadores imagen. Ediciones de la Universidad Nacional de San Juan. [En línea] Recuperado de:
http://bibliotecavirtual.unad.edu.co:2139/eds/detail/detail?vid=0&sid=c0a07f6e-e9a3-4eb9-a992-db7b326d7b44@pdc-v-sessmgr01&bdata=Jmxhbmc9ZXMmc2l0ZT1lZHMtbGl2ZSZzY29wZT1zaXRl%23AN=edsbas.F902F40F&db=edsbas

Navas, G. Sergio. Rodríguez Medina, Gustavo. Eterovic, Jorge (2014). Aplicación del filtro de Canny a la esteganografía digital. XVI WICC, ISBN 978-950-34-1084-4 Ushuaia. [En línea] Recuperado de: http://sedici.unlp.edu.ar/handle/10915/40706

Navas, G. Sergio. Rodríguez Medina, Gustavo. Eterovic, Jorge (2015). Selección óptima de métodos de sustitución en aplicaciones esteganográficas. XVII WICC, ISBN 978-987-633-134-0 Salta. [En línea] Recuperado de: http://sedici.unlp.edu.ar/handle/10915/45218

ORTEGA TRIGUERO, Jesús J. (España, 2006). *Introducción a la criptografía. Historia y actualidad*. 2.ª Edición. Ediciones de la Universidad de Castilla-La Mancha. Área computación, ISBN 978-84-8427-946-4.

Pastor, J.; Sarasa, M. A.; Salazar J. L. (2010). *Criptografía digital. Fundamentos y aplicaciones*. 2.ª Edición. Editorial Prensas de la Universidad de Zaragoza.

Rodríguez Medina, Gustavo. Navas, G. Sergio. (2016). Esteganografía: Sustitución LSB 1 bi utilizando MatLabt. XVIII WICC, ISBN 978-950-698-377-2 Entre Ríos. [En línea] Recuperado de: http://sedici.unlp.edu.ar/handle/10915/52766

Rojas, O. Eduardo. (2013). Propiedad intelectual y marcas de agua digitales (pág. 17). Recuperado de: https://es.slideshare.net/erojaso/dlp-stegano-pi

Santillán, M. A. (2003). Revista Facultad de Ciencias Administrativas, sisbib unmsm. Obtenido de La ética Profesional (pag. 69 – 78). Recuperado de: http://sisbib.unmsm.edu.pe/bibvirtualdata/publicaciones/administracion/n12_2003/a08.pdf

Schneier, B. (1996). *Applied Cryptography*. 2.ª Edición. Editorial Wiley.

Sharif, L.; Ahmed, M (2010). IPSec: PracticalApproach: Network Security. Editorial LAP LAMBERT Academic Pblishing, 2010.

Singh, S. (1999).The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography. Ed. Doubleday [En línea] Recuperado de: http://simonsingh.net/cryptography/crypto-cd-rom/

Sourceforge. (11 de octubre de 2003).StegHide. [En línea] Recuperado de: http://steghide.sourceforge.net/documentation.php

Steganografía. Conceptos y fundamentos (Mayo 2004). [En línea] Recuperado de: http://www.jjtc.com/Security/stegtools.html

Tanenbaum, Andrew. Woodhull, A. (2006). *Operating Systems Design and Implementation*. 3.ª Edición. Editorial Prentice Hall.

Tanenbaum, Andrew. Wetherall, David J. (México, 2012). *Redes de Computadoras*. 5.ª Edición. Editorial Pearson. Área computación, ISBN 978-607-32-0817-8.

Triguero, Jesús Ortega Triguero. López Guerrero, Miguel Angel. (Enero de 2006). Introducción a la criptografía: historia y actualidad, Ediciones de la universidad de Castilla-La Manch. [En línea] Recuperado de: http://bibliotecavirtual.unad.edu.co:2162/openurl?sid=EBSCO:edselb&genre=book&issn=edselb.61F234B4&ISBN=9788484279464&volume=&issue=&date=&spage=&pages=&title=Introducci%C3%B3n%20a%20la%20criptograf%C3%ADa:%20historia%20y%20actualidad&atitle=Introducci%C3%B3n%20a%20la%20criptograf%C3%ADa:%20historia%20y%20actualidad&aulast=Ortega%20Triguero,%20Jes%C3%BAs&id=DOI

Vaidy, S. (23 de junio de 2017).OpenStego. [En línea] Recuperado de: https://www.openstego.com

ANEXOS

ANEXO 1. CODIGO FUENTE PROTOTIPO I

```
Sub NewGradient (R1 As Integer, G1 As Integer, B1 As Integer, R2 As Integer,
G2 As Integer, B2 As Integer, obj As Form, WhichWay As Integer, TopOrBottom
As Integer)
 vert = 0
 horz = 1
 If WhichWay = vert Then
  pixels = obj.Height
 End If
 If WhichWay = horz Then
  pixels = obj.Width
 End If
 If R1 < R2 Then
  tempR1 = R1
  tempR2  =  R2
  R1 = tempR2
  R2 = tempR1
 End If
 If G1 < G2 Then
  tempG1 = G1
  tempG2  =  G2
  G1 = tempG2
  G2 = tempG1
 End If
 If B1 < B2 Then
  tempB1 = B1
  tempB2  =  B2
  B1 = tempB2
  B2 = tempB1
 End If
 If (R1 - R2) <> 0 Then
  nRStep = (R1 - R2) / pixels
 End If
 If (G1 - G2) <> 0 Then
  nGStep = (G1 - G2) / pixels
 End If
 If (B1 - B2) <> 0 Then
  nBStep = (B1 - B2) / pixels
 End If
 For X = 1 To pixels
  If TopOrBottom = 0 Then
   nR = nR + nRStep
```

```
   nG = nG + nGStep
   nB = nB + nBStep
   R = R1 - nR
   G = G1 - nG
   B = B1 - nB
  End If
  If TopOrBottom = 1 Then
   R = R + nRStep
   G = G + nGStep
  B = B + nBStep
  End If
  If R < 0 Then R = 0
  If G < 0 Then G = 0
  If B < 0 Then B = 0
  If WhichWay = vert Then obj.Line (1, X)-(obj.Width, X), RGB(R, G, B), BF
  If WhichWay = horz Then obj.Line (X, 1)-(X, obj.Height), RGB(R, G, B),
BF
 Next
End Sub

Private Sub Form_Load()
```

Codificación del Formulario Principal - FrmMain

```
Option Explicit
Private imgWidth As Long
Private imgHeight As Long
Private valor_X As Long
Private valor_Y As Long
Private imgLoaded As Boolean
Private colPositions As Collection
Dim Kx As Integer: Dim Ky As Integer

Private Sub EncodeByte (Value As Byte)
    On Error Resume Next
    Dim i As Integer
    Dim offset As Integer
    Dim bit As Byte
    Dim X As Long
    Dim Y As Long
    Dim C As Integer
    Dim strPosition As String
    Dim pixel As Long
    Dim R As Byte
    Dim G As Byte
    Dim B As Byte
    offset = 1
    For i = 1 To 8
```

```
        Do
            'X = Int(Rnd * imgWidth)
            'Y = Int(Rnd * imgHeight)
            X = Kx
            Y = Ky
            C = 0 'Int(Rnd * 100) Mod 3
            strPosition = "[" & C & "," & X & "," & Y & "]"
            colPositions.Add strPosition, strPosition
            If Err = 0 Then Exit Do
        Loop
        ' Obtener valores Red-Green-Blue en el pixel X Y
        pixel = picImage.Point(X, Y)
        R = pixel And &HFF&
        G = (pixel And &HFF00&) \ &H100&
        B = (pixel And &HFF0000) \ &H10000
        ' Determina si el bit is 0 or 1
        If Value And offset Then bit = 1 Else bit = 0
        ' Adiciona el bit para el cannel cero
        Select Case C
            Case 0, 1, 2
                R = (R And &HFE) Or bit
        End Select
        'Update el pixel en el estego Objeto
        picImage.PSet (X, Y), RGB(R, G, B)
        offset = offset * 2
        Kx = Kx + 15
        If Kx >= valor_X Then
         Kx = 15
         Ky = Ky + 15
        End If
    Next i
End Sub

Private Function DecodeByte() As Integer
    On Error Resume Next
    Dim Value As Integer
    Dim i As Integer
    Dim offset As Integer
    Dim bit As Byte
    Dim X As Long
    Dim Y As Long
    Dim C As Integer
    Dim strPosition As String
    Dim pixel As Long
    Dim R As Byte
    Dim G As Byte
    Dim B As Byte
    offset = 1
```

```
    For i = 1 To 8
        'Obtener posición
        Do
            X = Kx
            Y = Ky
            C = 0
            strPosition = "[" & C & "," & X & "," & Y & "]"
            colPositions.Add strPosition, strPosition
            If Err = 0 Then Exit Do
        Loop
        'Obtener valores Red-Green-Blue en el pixel X,Y
        pixel = picImage.Point(X, Y)
        R = pixel And &HFF&
        G = (pixel And &HFF00&) \ &H100&
        B = (pixel And &HFF0000) \ &H10000
        'Determinar si el bit es 0 or 1
        Select Case C
            Case 0, 1, 2
                bit = (R And &H1)
        End Select
        'Incremento el valor en el byte
        If bit Then
            Value = Value Or offset
        End If
        offset = offset * 2
        Kx = Kx + 15
        If Kx >= valor_X Then
         Kx = 15
         Ky = Ky + 15
        End If
    Next i
    DecodeByte = Value
End Function

Private Function CalculaHash(ByVal password As String) As Long

    ' Calcula el HASH basado en la contraseña para ocultar
    ' el mensaje secreto, retorna un long que es único
    Dim Value As Long
    Dim ch As Long
    Dim shift1 As Long
    Dim shift2 As Long
    Dim i As Integer
    Dim str_len As Integer
    shift1 = 3
    shift2 = 17
    str_len = Len(password)
    For i = 1 To str_len
```

```
        ch = Asc(Mid$(password, i, 1))
        Value = Value Xor (ch * 2 ^ shift1)
        Value = Value Xor (ch * 2 ^ shift2)
        shift1 = (shift1 + 7) Mod 19
        shift2 = (shift2 + 13) Mod 23
    Next i
    CalculaHash = Value
End Function

Private Sub cmdEncode_Click()
    Dim Long_Hash As String
    Dim strMessage As String
    Dim caracter As String
    Dim i As Long
    Dim message_length As Long
    Dim long_msg As Long
    Dim V_Hash As Long
    Kx = 10: Ky = 10
    If imgLoaded = False Then
        MsgBox "Primero debe cargar una imagen!"
        Exit Sub
    End If
    V_Hash = CalculaHash(CStr(txtPassword.Text))
    ' Inicializar randomizer
    Rnd  -1
    Randomize V_Hash
    Set colPositions = New Collection
    strMessage = CStr(V_Hash)
    message_length = Len(strMessage) 'Longitud del HASH
    EncodeByte Asc(message_length) ' Oculta longitud HASH
    'Oculta el HASH
    For i = 1 To message_length
       EncodeByte Asc(Mid(strMessage, i, 1))
    Next i
    ' Asigna TAG al mensaje
    strMessage = "@ ∂" & txtMessage.Text
    message_length = Len(strMessage)
    message_length = Len(CStr(message_length))
    long_msg = Len(strMessage)
    'Ocultar longitud del mensaje ejemplo 9789
    For i = 1 To message_length ' Numero de caracteres de la long del
mensaje
     EncodeByte Asc(Mid(CStr(long_msg), i, 1))
    Next i
    ' Ocultar el mensaje
    For i = 1 To long_msg
        EncodeByte Asc(Mid(strMessage, i, 1))
    Next i
```

```
    While colPositions.Count
        colPositions.Remove 1
    Wend
    Set colPositions = Nothing
    ' Update control picture
    picImage.Picture = picImage.Image
    cmdSave.Enabled = True
    MsgBox "Proceso Exitoso", vbExclamation + vbOKOnly, "StegoCrypto"
End Sub

Private Sub cmdDecode_Click()
    Dim strMessage As String
    Dim caracter As String
    Dim i As Integer
    Dim message_length As Integer
    Dim V_Hash As Long
    If imgLoaded = False Then
        MsgBox "Primero debe cargar una imagen!"
        Exit Sub
    End If
    V_Hash = CalculaHash(CStr(txtPassword.Text))
    ' Inicializa randomizer
    Rnd  -1
    Randomize V_Hash
    Set colPositions = New Collection
    Kx = 10: Ky = 10
    'leer tamaño del HASH
    message_length = CInt(Chr(DecodeByte))
    strMessage = ""
    For i = 1 To message_length 'Ciclo para Leer el HASH
        strMessage = strMessage & Chr(DecodeByte)
    Next
    'Compara si el HASH almacenado corresponde con la clave de descifrado
    If V_Hash = CLng(strMessage) Then
        'Si es la clave Ahora extrae el tamaÒo del texto oculto
        strMessage = ""
        i = 1
        While caracter <> "@"
          caracter = Chr(DecodeByte)
          strMessage = strMessage & caracter
          i = i + 1
        Wend
        'Extraer texto oculto
        i = Len(strMessage)
        strMessage = Left(strMessage, i - 1)
        message_length = CLng(strMessage)
        strMessage = ""
        For i = 2 To message_length
```

```
         strMessage = strMessage & Chr(DecodeByte)
        Next
        txtMessage.Text = Mid(strMessage, 3)
        MsgBox "Proceso Exitoso", vbExclamation + vbOKOnly, "StegoCrypto"
    Else
        MsgBox "Verifique la contraseña", vbQuestion + vbOKOnly,
"StegoCrypto"
    End If
    While colPositions.Count
        colPositions.Remove 1
    Wend
    Set colPositions = Nothing
End Sub

Private Sub cmdLoad_Click()
 Dim sFilename As String
 ' Muestra cuadro de diálogo Abrir Archivo
    With CDialog
        .DialogTitle = "Abrir Imagen"
        '.Filter = "Windows Bitmap (*.bmp)|*.bmp|CompuServe Graphics
Interchange (*.jpeg)|*.jpeg"
        .Filter = "Todos los archivos de imagen|*.bmp;*.gif;*.jpg"
        .ShowOpen
        sFilename = .FileName
    End With
 ' Cargar imagen
    If sFilename <> "" Then
        picImage.Picture = LoadPicture(sFilename)
        imgWidth = picImage.Picture.Width
        imgHeight = picImage.Picture.Height
        If imgWidth > picImage.ScaleWidth Then imgWidth =
picImage.ScaleWidth
        If imgHeight > picImage.ScaleHeight Then imgHeight =
picImage.ScaleHeight
        imgLoaded = True
        cmdEncode.Enabled = True
        cmdDecode.Enabled = True
        Img1.Picture = LoadPicture(sFilename)
        valor_X = imgWidth
        valor_Y = imgHeight
    End If
End Sub

Private Sub cmdQuit_Click()
   Dim rta As VbMsgBoxResult
   rta = MsgBox("Desea salir de la aplicación?", vbYesNo Or vbQuestion,
   "StegoCrypto")
   If rta = vbYes Then End
```

```
End Sub

Private Sub cmdSave_Click()
    Dim sFilename As String
    ' Visualizar el Commondialog
    With CDialog
        .DialogTitle = "Save image"
        '.Filter = "Windows Bitmap (*.bmp)|*.bmp"
        .Filter = "Jpeg (*.jpg)|*.jpg"
        .DefaultExt = ".jpg"
        .ShowSave
        sFilename = .FileName
    End With
    ' Guardar Estego Objeto
    If sFilename <> "" Then
        SavePicture picImage.Picture, sFilename
    End If
End Sub

Private Sub Command1_Click()
FrmAcercaDe.Show  vbModal
End Sub

Private Sub txtMessage_Change()
 Dim long_Text As Long
   long_Text = Len(txtMessage.Text)
   Label3.Caption = "Caracteres a procesar: " & long_Text
End Sub
```

ANEXO 2. CODIGO FUENTE PROTOTIPO II

Codificación del Formulario Principal - FrmMain

```
Option Explicit
Private imgWidth As Long
Private imgHeight As Long
Private valor_X As Long
Private valor_Y As Long
Private imgLoaded As Boolean
Private colPositions As Collection
Private Pos_LSB
Private Color_RGB As Byte
Private Color As Byte
Private valor_LSB As Byte
Dim Kx As Integer: Dim Ky As Integer

Private Sub EncodeByte(Value As Byte)
   On Error Resume Next
   Dim i As Byte
   Dim offset As Integer
   Dim bit1, bit2, bit3, bit4, bit5, bit6, bit7, bit8 As Byte
   Dim X As Long
   Dim Y As Long
   Dim C As Byte
   Dim strPosition As String
   Dim pixel As Long
   Dim R As Byte
   Dim G As Byte
   Dim B As Byte
   offset = 1
   If Color_RGB = 1 Then ' 1 Color
     For i = 1 To 8
       ' Obtener posición aleatoria de acuerdo al HASH
          X = Kx
          Y = Ky
          C = 0
          strPosition = "[" & C & "," & X & "," & Y & "]"
          colPositions.Add strPosition, strPosition
          If Err = 0 Then Exit Do
       Loop
       'Obtener el valor de Red-Green-Blue en el pixel X,Y
       pixel = picImage.Point(X, Y)
       R = pixel And &HFF&
       G = (pixel And &HFF00&) \ &H100&
       B = (pixel And &HFF0000) \ &H10000
       ' Determinar si el bite is 0 or 1
       If Value And offset Then bit1 = valor_LSB Else bit1 = 0
```

```
    ' Add bit de acuerdo al color seleccionado
    Select Case Color
    Case 1
      R = (R And Pos_LSB) Or bit1
    Case 2
      G = (G And Pos_LSB) Or bit1
    Case 3
      B = (B And Pos_LSB) Or bit1
    End Select
    ' Update el pixel en la imagen
    picImage.PSet (X, Y), RGB(R, G, B)
    offset = offset * 2
    Kx = Kx + 15 'Kx = Kx + (Int(Rnd * 10) Mod 10) + 1 'Kx = Kx + 1
    If Kx >= valor_X Then
      Kx = 15
      Ky = Ky + 15 'Ky = Ky + (Int(Rnd * 10) Mod 10) + 1 'Ky = Ky + 1
    End If
  Next i
End If

If Color_RGB = 2 Then ' 2 Colores
 For i = 1 To 4
    'Obtener la posición de acuerdo al HASH
    Do
       'X = Int(Rnd * imgWidth)
       'Y = Int(Rnd * imgHeight)
       X = Kx
       Y = Ky
       C = 0
       strPosition = "[" & C & "," & X & "," & Y & "]"
       colPositions.Add strPosition, strPosition
       If Err = 0 Then Exit Do
    Loop
    'Obtener el valor Red-Green-Blue de el pixel
    pixel = picImage.Point(X, Y)
    R = pixel And &HFF&
    G = (pixel And &HFF00&) \ &H100&
    B = (pixel And &HFF0000) \ &H10000
    ' Determine wether bit is 0 or 1
    If Value And offset Then bit1 = valor_LSB Else bit1 = 0
    'Adicionar bit dentro del color seleccionado
    Select Case Color
       Case 4, 5
            R = (R And Pos_LSB) Or bit1 'R = (R And &H7F) Or Bit
       Case 6
            G = (G And Pos_LSB) Or bit1 'G = (G And &H7F) Or Bit
       'Case 7
            'B = (B And &H7F) Or Bit
    End Select
```

```
    'R = (R And &H7F) Or bit1
    'R = (R And Pos_LSB) Or bit1
    Do
       C = 1
       strPosition = "[" & C & "," & X & "," & Y & "]"
       colPositions.Add strPosition, strPosition
       If Err = 0 Then Exit Do
    Loop
    If Value And offset * 2 Then bit2 = valor_LSB Else bit2 = 0
    'G = (G And &H7F) Or bit2
    Select Case Color
       Case 4
            G = (G And Pos_LSB) Or bit2 'G = (G And &H7F) Or Bit
       Case 5, 6
            B = (B And Pos_LSB) Or bit2 'B = (B And &H7F) Or Bit
    End Select
    'Update el pixel en la imagen
    picImage.PSet (X, Y), RGB(R, G, B)
    offset = offset * 2 * 2
    Kx = Kx + 15 'Kx = Kx + (Int(Rnd * 10) Mod 10) + 1 'Kx = Kx + 1
    If Kx >= valor_X Then
      Kx = 15
      Ky = Ky + 15 'Ky = Ky + (Int(Rnd * 10) Mod 10) + 1 'Ky = Ky + 1
    End If
 Next i
End If ' Color_RGB = 2 Then ' 2 Colores
If Color_RGB = 3 Then ' 3 Colores
 For i = 1 To 2
    'Obtener la posición aleatoria de acuerdo al HASH
    Do
       'X = Int(Rnd * imgWidth)
       'Y = Int(Rnd * imgHeight)
       X = Kx
       Y = Ky
       C = 0
       strPosition = "[" & C & "," & X & "," & Y & "]"
       colPositions.Add strPosition, strPosition
       If Err = 0 Then Exit Do
    Loop
    'Obtener los valores Red-Green-Blue en el pixel X,Y
    pixel = picImage.Point(X, Y)
    R = pixel And &HFF&
    G = (pixel And &HFF00&) \ &H100&
    B = (pixel And &HFF0000) \ &H10000
    ' Determine wether bit is 0 or 1
    If Value And offset Then bit1 = valor_LSB Else bit1 = 0
    ' Adicionar el bit al color Red
    R = (R And Pos_LSB) Or bit1
    Do
```

```
      C = 1
      strPosition = "[" & C & "," & X & "," & Y & "]"
      colPositions.Add strPosition, strPosition
      If Err = 0 Then Exit Do
    Loop
    If Value And offset * 2 Then bit2 = valor_LSB Else bit2 = 0
    G = (G And Pos_LSB) Or bit2
    Do
      C = 2
      strPosition = "[" & C & "," & X & "," & Y & "]"
      colPositions.Add strPosition, strPosition
      If Err = 0 Then Exit Do
    Loop
    If Value And offset * 4 Then bit3 = valor_LSB Else bit3 = 0
    B = (B And Pos_LSB) Or bit3
    ' Update el pixel en la imagen
    picImage.PSet (X, Y), RGB(R, G, B)
    offset = offset * 2 * 2 * 2
    Kx = Kx + 15 'Kx = Kx + (Int(Rnd * 10) Mod 10) + 1 'Kx = Kx + 1
    If Kx >= valor_X Then
      Kx = 15
      Ky = Ky + 15 'Ky = Ky + (Int(Rnd * 10) Mod 10) + 1 'Ky = Ky + 1
    End If
Next i
'**********++++ Ultimos 2 dosssss bits
' Obtener la posición  X,Y
Do
    X = Kx
    Y = Ky
    C = 0
    strPosition = "[" & C & "," & X & "," & Y & "]"
    colPositions.Add strPosition, strPosition
    If Err = 0 Then Exit Do
Loop
pixel = picImage.Point(X, Y)
R = pixel And &HFF&
G = (pixel And &HFF00&) \ &H100&
B = (pixel And &HFF0000) \ &H10000
If Value And offset Then bit1 = valor_LSB Else bit1 = 0
R = (R And Pos_LSB) Or bit1
Do
      C = 1
      strPosition = "[" & C & "," & X & "," & Y & "]"
      colPositions.Add strPosition, strPosition
      If Err = 0 Then Exit Do
Loop
If Value And offset * 2 Then bit2 = valor_LSB Else bit2 = 0
G = (G And Pos_LSB) Or bit2
```

```
    Kx = Kx + 15 'Kx = Kx + (Int(Rnd * 10) Mod 10) + 1 'Kx = Kx + 1
    If Kx >= valor_X Then
       Kx = 15
       Ky = Ky + 15 'Ky = Ky + (Int(Rnd * 10) Mod 10) + 1 'Ky = Ky + 1
    End If
    picImage.PSet (X, Y), RGB(R, G, B)
   End If ' Color_RGB = 3 Then ' 3 Colores
End Sub

Private Function DecodeByte() As Integer
  On Error Resume Next
   Dim Value As Integer
   Dim i As Byte
   Dim offset As Integer
   'Dim bit As Byte
   Dim bit1, bit2, bit3, bit4, bit5, bit6, bit7, bit8 As Byte
   Dim X As Long
   Dim Y As Long
   Dim C As Byte
   Dim strPosition As String
   Dim pixel As Long
   Dim R As Byte
   Dim G As Byte
   Dim B As Byte
   Value = 0
   offset = 1
  If Color_RGB = 1 Then ' 1 Color
    For i = 1 To 8
      Do
         X = Kx
         Y = Ky
         C = 0
         strPosition = "[" & C & "," & X & "," & Y & "]"
         colPositions.Add strPosition, strPosition
         If Err = 0 Then Exit Do
      Loop
      ' Obtener el valor Red-Green-Blue en el pixel
      pixel = picImage.Point(X, Y)
      R = pixel And &HFF&
      G = (pixel And &HFF00&) \ &H100&
      B = (pixel And &HFF0000) \ &H10000
      ' Determinar que color se seleccionó
      Select Case Color
      Case 1
        bit1 = (R And Pos_LSB)
      Case 2
        bit1 = (G And Pos_LSB)
      Case 3
        bit1 = (B And Pos_LSB)
```

```
      End Select
      'Incrementar el valor del byte
      If bit1 Then
         Value = Value Or offset
      End If
      offset = offset * 2
      Kx = Kx + 15 'Kx = Kx + (Int(Rnd * 10) Mod 10) + 1 'Kx = Kx + 1
      If Kx >= valor_X Then
        Kx = 15
        Ky = Ky + 15 'Ky = Ky + (Int(Rnd * 10) Mod 10) + 1 'Ky = Ky + 1
      End If
    Next i
  End If
'*****************************Inicio de 2 colores
If Color_RGB = 2 Then ' 2 Colores
   For i = 1 To 4
      Do
         X = Kx
         Y = Ky
         C = 0
         strPosition = "[" & C & "," & X & "," & Y & "]"
         colPositions.Add strPosition, strPosition
         If Err = 0 Then Exit Do
      Loop
      pixel = picImage.Point(X, Y)
      R = pixel And &HFF&
      G = (pixel And &HFF00&) \ &H100&
      B = (pixel And &HFF0000) \ &H10000
      Select Case Color
      Case 4, 5
        bit1 = (R And Pos_LSB)
      Case 6
        bit1 = (G And Pos_LSB)
      End Select
      If bit1 Then
         Value = Value Or offset
      End If
      Do
         C = 1
         strPosition = "[" & C & "," & X & "," & Y & "]"
         colPositions.Add strPosition, strPosition
         If Err = 0 Then Exit Do
      Loop
      offset = offset * 2
      Select Case Color
      Case 4
        bit2 = (G And Pos_LSB) 'G = (G And &H7F) Or Bit
      Case 5, 6
        bit2 = (B And Pos_LSB) 'B = (B And &H7F) Or Bit
```

```
      End Select
      If bit2 Then
         Value = Value Or offset
      End If
      offset = offset * 2
      Kx = Kx + 15 'Kx = Kx + (Int(Rnd * 10) Mod 10) + 1 'Kx = Kx + 1
      If Kx >= valor_X Then
        Kx = 15
        Ky = Ky + 15 'Ky = Ky + (Int(Rnd * 10) Mod 10) + 1 'Ky = Ky + 1
      End If
    Next i
  End If ' Color_RGB = 2 Then ' 2 Colores
'''''''''**********++++++++++++++++++++++++++ Fin de 2 colores

If Color_RGB = 3 Then ' 3 Colores
  For i = 1 To 2
      Do
         X = Kx
         Y = Ky
         C = 0
         strPosition = "[" & C & "," & X & "," & Y & "]"
         colPositions.Add strPosition, strPosition
         If Err = 0 Then Exit Do
      Loop
      pixel = picImage.Point(X, Y)
      R = pixel And &HFF&
      G = (pixel And &HFF00&) \ &H100&
      B = (pixel And &HFF0000) \ &H10000
      bit1 = (R And Pos_LSB)
      If bit1 Then
         Value = Value Or offset
      End If
      Do
         C = 1
         strPosition = "[" & C & "," & X & "," & Y & "]"
         colPositions.Add strPosition, strPosition
         If Err = 0 Then Exit Do
      Loop
      offset = offset * 2
      bit2 = (G And Pos_LSB)
      If bit2 Then
         Value = Value Or offset
      End If
      Do
         C = 2
         strPosition = "[" & C & "," & X & "," & Y & "]"
         colPositions.Add strPosition, strPosition
         If Err = 0 Then Exit Do
      Loop
```

```
      offset = offset * 2
      bit3 = (B And Pos_LSB)
      If bit3 Then
         Value = Value Or offset
      End If
      Kx = Kx + 15 'Kx = Kx + (Int(Rnd * 10) Mod 10) + 1 'Kx = Kx + 1
      If Kx >= valor_X Then
       Kx = 15
       Ky = Ky + 15 'Ky = Ky + (Int(Rnd * 10) Mod 10) + 1 'Ky = Ky + 1
      End If
      offset = offset * 2
   Next i
   'Ahora bit 7 y 8 **********************
   '********************++++ Ultimos 2 dosssss bits
   Do
      X = Kx
      Y = Ky
      C = 0
      strPosition = "[" & C & "," & X & "," & Y & "]"
      colPositions.Add strPosition, strPosition
      If Err = 0 Then Exit Do
   Loop
   pixel = picImage.Point(X, Y)
   R = pixel And &HFF&
   G = (pixel And &HFF00&) \ &H100&
   B = (pixel And &HFF0000) \ &H10000
   bit7 = (R And Pos_LSB)
   If bit7 Then
         Value = Value Or offset
   End If
   Do
         C = 1
         strPosition = "[" & C & "," & X & "," & Y & "]"
         colPositions.Add strPosition, strPosition
         If Err = 0 Then Exit Do
   Loop
   offset = offset * 2
   bit8 = (G And Pos_LSB)
   If bit8 Then
         Value = Value Or offset
   End If
   Kx = Kx + 15 'Kx = Kx + (Int(Rnd * 10) Mod 10) + 1 'Kx = Kx + 1
   If Kx >= valor_X Then
      Kx = 15
      Ky = Ky + 15 'Ky = Ky + (Int(Rnd * 10) Mod 10) + 1 'Ky = Ky + 1
   End If
   'Terminacion bit 7 y 8 ************************
End If 'Color_RGB = 3 Then ' 3 Colores
DecodeByte = Value
```

```
End Function

Private Function CalculaHash(ByVal password As String) As Long

    'Calcular el valor HASH de acuerdo al password del mensaje cifrado

    Dim Value As Long
    Dim ch As Long
    Dim shift1 As Long
    Dim shift2 As Long
    Dim i As Integer
    Dim str_len As Integer
    shift1 = 3
    shift2 = 17
    str_len = Len(password)
    For i = 1 To str_len
        ch = Asc(Mid$(password, i, 1))
        Value = Value Xor (ch * 2 ^ shift1)
        Value = Value Xor (ch * 2 ^ shift2)
        shift1 = (shift1 + 7) Mod 19
        shift2 = (shift2 + 13) Mod 23
    Next i
    CalculaHash = Value
End Function

Private Sub Check4_Click()
 If Check4.Value = 1 Then
    Encrypt txtMessage.Text, txtMessage
 End If
End Sub

Private Sub Check5_Click()
 If Check5.Value = 1 Then
    Decrypt txtMessage.Text, txtMessage
 End If
End Sub

Private Sub cmdEncode_Click()
    'On Error Resume Next
    Dim Long_Hash As String
    Dim strMessage As String
    Dim caracter As String
    Dim i As Long
    Dim message_length As Long
    Dim long_msg As Long
    Dim V_Hash As Long
    Kx = 10: Ky = 10
    If imgLoaded = False Then
        MsgBox "Primero debe cargar una imagen!"
```

```
    Exit Sub
End If
If Option1.Value Then
  Pos_LSB = &H7F ' 8 bit
  valor_LSB = 128
ElseIf Option2.Value Then
  Pos_LSB = &HBF ' 7 bit
  valor_LSB = 64
ElseIf Option3.Value Then
  Pos_LSB = &HDF ' 6 bit
  valor_LSB = 32
ElseIf Option4.Value Then
  Pos_LSB = &HEF ' 5 bit
  valor_LSB = 16
ElseIf Option5.Value Then
  Pos_LSB = &HF7 ' 4 bit
  valor_LSB = 8
ElseIf Option6.Value Then
  Pos_LSB = &HFB ' 3 bit
  valor_LSB = 4
ElseIf Option7.Value Then
  Pos_LSB = &HFD ' 2 bit
  valor_LSB = 2
ElseIf Option8.Value Then
  Pos_LSB = &HFE ' 1 bit
  valor_LSB = 1
End If
If Check1.Value And Check2.Value = False And Check3.Value = False Then
  Color_RGB = 1
  Color = 1 ' Red
ElseIf Check1.Value = False And Check2.Value And Check3.Value = False Then
  Color_RGB = 1
  Color = 2 'Green
ElseIf Check1.Value = False And Check2.Value = False And Check3.Value Then
  Color_RGB = 1
  Color = 3 'Blue
ElseIf Check1.Value And Check2.Value And Check3.Value = False Then
  Color_RGB = 2
  Color = 4 'RG
ElseIf Check1.Value And Check2.Value = False And Check3.Value Then
  Color_RGB = 2
  Color = 5 'RB
ElseIf Check1.Value = False And Check2.Value And Check3.Value Then
  Color_RGB = 2
  Color = 6 'GB
ElseIf Check1.Value And Check2.Value And Check3.Value Then
  Color_RGB = 3
  Color = 8 'RGB
ElseIf Check1.Value = False And Check2.Value = False And Check3.Value = False Then
```

```
        MsgBox "Debe seleccionar un Color (RGB)!"
        Exit Sub
    End If
    V_Hash = CalculaHash(CStr(txtPassword.Text))
    ' Initializar randomizer
    Rnd -1
    Randomize V_Hash
    Set colPositions = New Collection
    strMessage = CStr(V_Hash)
    message_length = Len(strMessage) 'Longitud del HASH
    EncodeByte Asc(message_length) ' Oculta longitud HASH
    'Oculta el HASH
    For i = 1 To message_length
      EncodeByte Asc(Mid(strMessage, i, 1))
    Next i
    ' Asigna TAG al mensaje
    strMessage = "@ ∂" & txtMessage.Text
    message_length = Len(strMessage)
    message_length = Len(CStr(message_length))
    long_msg = Len(strMessage)
    'Ocultar longitud del mensaje ejemplo 9789
    For i = 1 To message_length ' Numero de caracteres de la long del mensaje
     EncodeByte Asc(Mid(CStr(long_msg), i, 1))
    Next i
    ' Ocultar el mensaje
    Timer1.Enabled = True
    Label3.Caption = "Caracteres a procesar: " & long_msg
    For i = 1 To long_msg
        EncodeByte Asc(Mid(strMessage, i, 1))
        If (i Mod 500) = 0 Then
          Label4.Caption = "Bit procesados: " & i
          OpenForms = DoEvents
        End If
    Next i
    While colPositions.Count
        colPositions.Remove 1
    Wend
    Set colPositions = Nothing
    ' Update control picture
    picImage.Picture = picImage.Image
    cmdSave.Enabled = True
    MsgBox "Proceso Exitoso", vbExclamation + vbOKOnly, "StegoCrypto"
    Timer1.Enabled = False
End Sub

Private Sub cmdDecode_Click()
    Dim strMessage As String
    Dim caracter As String
    Dim i As Integer
```

```
Dim message_length As Integer
Dim V_Hash As Long
If imgLoaded = False Then
    MsgBox "Primero debe cargar una imagen!"
    Exit Sub
End If
If Option1.Value Then
   Pos_LSB = &H80 ' 8 bit
ElseIf Option2.Value Then
  Pos_LSB = &H40 ' 7 bit
ElseIf Option3.Value Then
  Pos_LSB = &H20 ' 6 bit
ElseIf Option4.Value Then
  Pos_LSB = &H10 ' 5 bit
ElseIf Option5.Value Then
   Pos_LSB = &H8 ' 4 bit
ElseIf Option6.Value Then
   Pos_LSB = &H4 ' 3 bit
ElseIf Option7.Value Then
   Pos_LSB = &H2 ' 2 bit
ElseIf Option8.Value Then
   Pos_LSB = &H1 ' 1 bit
End If
If Check1.Value And Check2.Value = False And Check3.Value = False Then
   Color_RGB = 1
   Color = 1 ' Red
ElseIf Check1.Value = False And Check2.Value And Check3.Value = False Then
   Color_RGB = 1
   Color = 2 'Green
ElseIf Check1.Value = False And Check2.Value = False And Check3.Value Then
   Color_RGB = 1
   Color = 3 'Blue
ElseIf Check1.Value And Check2.Value And Check3.Value = False Then
   Color_RGB = 2
   Color = 4 'RG
ElseIf Check1.Value And Check2.Value = False And Check3.Value Then
   Color_RGB = 2
   Color = 5 'RB
ElseIf Check1.Value = False And Check2.Value And Check3.Value Then
   Color_RGB = 2
   Color = 6 'GB
ElseIf Check1.Value And Check2.Value And Check3.Value Then
   Color_RGB = 3
   Color = 8 'RGB
ElseIf Check1.Value = False And Check2.Value = False And Check3.Value = False Then
    MsgBox "Debe seleccionar un Color (RGB)!"
    Exit Sub
End If
V_Hash = CalculaHash(CStr(txtPassword.Text))
```

```
    ' Initializar randomizer
    Rnd -1
    Randomize V_Hash
    Set colPositions = New Collection
    Kx = 10: Ky = 10
    'leer tamaÒo del HASH
    caracter = Chr(DecodeByte)
    Select Case caracter
    Case "1", "2", "3", "4", "5", "6", "7", "8", "9":
      message_length = CInt(caracter)
    Case Else:
     MsgBox "No existe Mensaje Secreto", vbQuestion + vbOKOnly, "StegoCrypto"
     Exit Sub
    End Select
    strMessage = ""
    Label3.Caption = "": Label4.Caption = ""
    For i = 1 To message_length 'Ciclo para Leer el HASH
      strMessage = strMessage & Chr(DecodeByte)
    Next
    'Compara si el HASH almacenado corresponde con la clave de descifrado
    If V_Hash = CLng(strMessage) Then
       'Si es la clave Ahora extrae el tamaÒo del texto oculto
       strMessage = ""
       i = 1
       While caracter <> "@"
         caracter = Chr(DecodeByte)
         strMessage = strMessage & caracter
         i = i + 1
       Wend
       'Extraer texto oculto
       i = Len(strMessage)
       strMessage = Left(strMessage, i - 1)
       message_length = CLng(strMessage)
       Label3.Caption = "Caracteres a procesar Func HASH: " & message_length
       strMessage = ""
       For i = 2 To message_length
        strMessage = strMessage & Chr(DecodeByte)
       Next
       txtMessage.Text = Mid(strMessage, 3)
       MsgBox "Proceso Exitoso", vbExclamation + vbOKOnly, "StegoCrypto"
    Else
       MsgBox "Verifique la contraseña", vbQuestion + vbOKOnly, "StegoCrypto"
    End If
    While colPositions.Count
       colPositions.Remove 1
    Wend
    Set colPositions = Nothing
End Sub
```

```
Private Sub cmdLoad_Click()
    Dim sFilename As String
    ' Show dialog
    With CDialog
        .DialogTitle = "Abrir Imagen"
        '.Filter = "Windows Bitmap (*.bmp)|*.bmp|CompuServe Graphics Interchange
(*.jpeg)|*.jpeg"
        .Filter = "Todos los archivos de imagen|*.bmp;*.gif;*.jpg"
        .ShowOpen
        sFilename = .FileName
    End With
    ' Cargar Imagen
    If sFilename <> "" Then
        picImage.Picture = LoadPicture(sFilename)
        imgWidth = picImage.Picture.Width
        imgHeight = picImage.Picture.Height
        If imgWidth > picImage.ScaleWidth Then imgWidth = picImage.ScaleWidth
        If imgHeight > picImage.ScaleHeight Then imgHeight = picImage.ScaleHeight
        imgLoaded = True
        cmdEncode.Enabled = True
        cmdDecode.Enabled = True
        Img1.Picture = LoadPicture(sFilename)
        valor_X = imgWidth
         valor_Y = imgHeight
    End If
End Sub

Private Sub cmdQuit_Click()
  Dim rta As VbMsgBoxResult
   rta = MsgBox("Desea salir de la aplicaciÛn?", vbYesNo Or vbQuestion, "StegoCrypto")
   If rta = vbYes Then End

End Sub

Private Sub cmdSave_Click()
  Dim sFilename As String
    ' Visualizar el Commondialog
    With CDialog
        .DialogTitle = "Save image"
        '.Filter = "Windows Bitmap (*.bmp)|*.bmp"
        .Filter = "Jpeg (*.jpg)|*.jpg"
        .DefaultExt = ".jpg"
        .ShowSave
        sFilename = .FileName
    End With
    ' Guardar Estego Objeto
    If sFilename <> "" Then
        SavePicture picImage.Picture, sFilename
    End If
```

```
End Sub


Private Sub Command1_Click()
   FrmAcercaDe.Show vbModal
End Sub

Private Sub Timer1_Timer()
    'If frmMain.WindowState = vbNormal Then
    'frmMain.picImage.Picture = picImage.Image
    'MsgBox "Procesando!", vbOKOnly, "StegoCrypto"
    'End If
    picImage.Refresh
End Sub

Private Sub txtMessage_Change()
  Dim long_Text As Long
   long_Text = Len(txtMessage.Text)
   Label3.Caption = "Caracteres a procesar: " & long_Text
End Sub
```

Módulo de Encriptación y Desencriptación

```
Public Sub Decrypt(Text As String, Output As TextBox)
    On Error GoTo Break
    Dim Char() As String
    Dim i As Long
    Dim Out As String
    Dim iLen As Long
    Dim Rand As Integer
    Dim RLen As Single
    RLen = Right(Text, 1) - 1
    RLen = RLen / 2
    Rand = StrReverse(Mid(Text, Len(Text) - RLen, RLen))
    Text = Mid(Text, 2, Len(Text) - RLen - 2)
    'Text = StrReverse(Text)
    iLen = Len(Text): ReDim Char(1 To iLen)
    For i = 1 To iLen
        Char(i) = Chr(Asc(Mid(Text, i, 1)) Xor Rand)
        Out = Out & Char(i)
    Next
    Output.Text = StrReverse(Out)
Exit Sub
Break:
MsgBox Err.Description, vbOKOnly + vbExclamation, "Error " & Err.Number

End Sub
```

```
Public Sub Encrypt(Text As String, Output As TextBox)
   On Error GoTo Break
   Randomize
   Dim Char() As String
   Dim si As Long
   Dim Out As String
   Dim iLen As Long
   Dim Rand
   Dim v1 As Byte
    Text = StrReverse(Text)
    iLen = Len(Text)
    Rand = Int(50 * Rnd) + 1
    ReDim Char(1 To iLen)
    Out = "¯"
    For i = 1 To iLen
        v1 = Asc(Mid(Text, i, 1))
        v1 = v1 Xor Rand
        Char(i) = Chr(v1)
        Out = Out & Char(i)
    Next
    Output.Text = Out & StrReverse(Rand) & Len(Rand) * 2 + 1
Exit Sub
Break:
MsgBox Err.Description, vbOKOnly + vbExclamation, "Error " & Err.Number

End Sub
```

ANEXO 3. CODIGO FUENTE PROTOTIPO III

```
Option Explicit
Private imgWidth As Long
Private imgHeight As Long
Private valor_X As Long
Private valor_Y As Long
Private imgLoaded As Boolean
Private colPositions As Collection
Private Pos_LSB
Private Color_RGB As Byte
Private Color As Byte
Private valor_LSB As Byte
Dim Kx As Integer: Dim Ky As Integer


Private Sub EncodeByte(Value As Byte)
    On Error Resume Next
    Dim i As Byte
    Dim offset As Integer
    Dim bit1, bit2, bit3, bit4, bit5, bit6, bit7, bit8 As Byte
    Dim X As Long
    Dim Y As Long
    Dim C As Byte
    Dim strPosition As String
    Dim pixel As Long
    Dim R As Byte
    Dim G As Byte
    Dim B As Byte
    offset = 1
    If Color_RGB = 1 Then ' 1 Color
      For i = 1 To 8
        Do
            X = Int(Rnd * imgWidth)
            Y = Int(Rnd * imgHeight)
            C = 0
            strPosition = "[" & C & "," & X & "," & Y & "]"
            colPositions.Add strPosition, strPosition
            If Err = 0 Then Exit Do
        Loop
        pixel = picImage.Point(X, Y)
        R = pixel And &HFF&
        G = (pixel And &HFF00&) \ &H100&
        B = (pixel And &HFF0000) \ &H10000
        If Value And offset Then bit1 = valor_LSB Else bit1 = 0
        ' Add bit to the selected channel
        Select Case Color
```

```
    Case 1
       R = (R And Pos_LSB) Or bit1
    Case 2
       G = (G And Pos_LSB) Or bit1
    Case 3
       B = (B And Pos_LSB) Or bit1
    End Select
    picImage.PSet (X, Y), RGB(R, G, B)
    offset = offset * 2
  Next i
End If

If Color_RGB = 2 Then ' 2 Colores
 For i = 1 To 4
    Do
        X = Int(Rnd * imgWidth)
        Y = Int(Rnd * imgHeight)
        C = 0
        strPosition = "[" & C & "," & X & "," & Y & "]"
        colPositions.Add strPosition, strPosition
        If Err = 0 Then Exit Do
    Loop
    pixel = picImage.Point(X, Y)
    R = pixel And &HFF&
    G = (pixel And &HFF00&) \ &H100&
    B = (pixel And &HFF0000) \ &H10000
    If Value And offset Then bit1 = valor_LSB Else bit1 = 0
    Select Case Color
        Case 4, 5
              R = (R And Pos_LSB) Or bit1 'R = (R And &H7F) Or Bit
        Case 6
              G = (G And Pos_LSB) Or bit1 'G = (G And &H7F) Or Bit
    End Select
    Do
        C = 1
        strPosition = "[" & C & "," & X & "," & Y & "]"
        colPositions.Add strPosition, strPosition
        If Err = 0 Then Exit Do
    Loop
    If Value And offset * 2 Then bit2 = valor_LSB Else bit2 = 0
    Select Case Color
        Case 4
              G = (G And Pos_LSB) Or bit2 'G = (G And &H7F) Or Bit
        Case 5, 6
              B = (B And Pos_LSB) Or bit2 'B = (B And &H7F) Or Bit
    End Select
    picImage.PSet (X, Y), RGB(R, G, B)
    offset = offset * 2 * 2
```

```
 Next i
End If ' Color_RGB = 2 Then ' 2 Colores

If Color_RGB = 3 Then ' 3 Colores
 For i = 1 To 2
    Do
        X = Int(Rnd * imgWidth)
        Y = Int(Rnd * imgHeight)
        C = 0
        strPosition = "[" & C & "," & X & "," & Y & "]"
        colPositions.Add strPosition, strPosition
        If Err = 0 Then Exit Do
    Loop
    pixel = picImage.Point(X, Y)
    R = pixel And &HFF&
    G = (pixel And &HFF00&) \ &H100&
    B = (pixel And &HFF0000) \ &H10000
    If Value And offset Then bit1 = valor_LSB Else bit1 = 0
    R = (R And Pos_LSB) Or bit1
    Do
        C = 1
        strPosition = "[" & C & "," & X & "," & Y & "]"
        colPositions.Add strPosition, strPosition
        If Err = 0 Then Exit Do
    Loop
    If Value And offset * 2 Then bit2 = valor_LSB Else bit2 = 0
    G = (G And Pos_LSB) Or bit2
    Do
        C = 2
        strPosition = "[" & C & "," & X & "," & Y & "]"
        colPositions.Add strPosition, strPosition
        If Err = 0 Then Exit Do
    Loop
    If Value And offset * 4 Then bit3 = valor_LSB Else bit3 = 0
    B = (B And Pos_LSB) Or bit3
    picImage.PSet (X, Y), RGB(R, G, B)
    offset = offset * 2 * 2 * 2
 Next i
 '********************++++ Ultimos 2 dosssss bits
 Do
    X = Int(Rnd * imgWidth)
    Y = Int(Rnd * imgHeight)
    C = 0
    strPosition = "[" & C & "," & X & "," & Y & "]"
    colPositions.Add strPosition, strPosition
    If Err = 0 Then Exit Do
 Loop
 pixel = picImage.Point(X, Y)
```

```
     R = pixel And &HFF&
     G = (pixel And &HFF00&) \ &H100&
     B = (pixel And &HFF0000) \ &H10000
     If Value And offset Then bit1 = valor_LSB Else bit1 = 0
     R = (R And Pos_LSB) Or bit1
     Do
            C = 1
            strPosition = "[" & C & "," & X & "," & Y & "]"
            colPositions.Add strPosition, strPosition
            If Err = 0 Then Exit Do
     Loop
     If Value And offset * 2 Then bit2 = valor_LSB Else bit2 = 0
     G = (G And Pos_LSB) Or bit2
     picImage.PSet (X, Y), RGB(R, G, B)
    End If ' Color_RGB = 3 Then ' 3 Colores
End Sub

Private Function DecodeByte() As Integer
   On Error Resume Next
    Dim Value As Integer
    Dim i As Byte
    Dim offset As Integer
    Dim bit1, bit2, bit3, bit4, bit5, bit6, bit7, bit8 As Byte
    Dim X As Long
    Dim Y As Long
    Dim C As Byte
    Dim strPosition As String
    Dim pixel As Long
    Dim R As Byte
    Dim G As Byte
    Dim B As Byte
    Value = 0
    offset = 1
   If Color_RGB = 1 Then ' 1 Color
      For i = 1 To 8
        Do
            X = Int(Rnd * imgWidth)
            Y = Int(Rnd * imgHeight)
            C = 0
            strPosition = "[" & C & "," & X & "," & Y & "]"
            colPositions.Add strPosition, strPosition
            If Err = 0 Then Exit Do
        Loop
        pixel = picImage.Point(X, Y)
        R = pixel And &HFF&
        G = (pixel And &HFF00&) \ &H100&
        B = (pixel And &HFF0000) \ &H10000
        Select Case Color
```

```
       Case 1
          bit1 = (R And Pos_LSB)
       Case 2
          bit1 = (G And Pos_LSB)
       Case 3
          bit1 = (B And Pos_LSB)
       End Select
       If bit1 Then
           Value = Value Or offset
       End If
       offset = offset * 2
     Next i
  End If
'*****************************Inicio de 2 colores
If Color_RGB = 2 Then ' 2 Colores
    For i = 1 To 4
       Do
           X = Int(Rnd * imgWidth)
           Y = Int(Rnd * imgHeight)
           C = 0
           strPosition = "[" & C & "," & X & "," & Y & "]"
           colPositions.Add strPosition, strPosition
           If Err = 0 Then Exit Do
       Loop
       pixel = picImage.Point(X, Y)
       R = pixel And &HFF&
       G = (pixel And &HFF00&) \ &H100&
       B = (pixel And &HFF0000) \ &H10000
       Select Case Color
       Case 4, 5
          bit1 = (R And Pos_LSB)
       Case 6
          bit1 = (G And Pos_LSB)
       End Select
       If bit1 Then
           Value = Value Or offset
       End If
       Do
           C = 1
           strPosition = "[" & C & "," & X & "," & Y & "]"
           colPositions.Add strPosition, strPosition
           If Err = 0 Then Exit Do
       Loop
       offset = offset * 2
       Select Case Color
       Case 4
          bit2 = (G And Pos_LSB) 'G = (G And &H7F) Or Bit
       Case 5, 6
```

```
            bit2 = (B And Pos_LSB) 'B = (B And &H7F) Or Bit
        End Select
        If bit2 Then
            Value = Value Or offset
        End If
        offset = offset * 2
    Next i
   End If ' Color_RGB = 2 Then ' 2 Colores
''''''''''''*****************++++++++++++++++++++++++++ Fin de 2 colores

If Color_RGB = 3 Then ' 3 Colores
   For i = 1 To 2
        Do
            X = Int(Rnd * imgWidth)
            Y = Int(Rnd * imgHeight)
            C = 0
            strPosition = "[" & C & "," & X & "," & Y & "]"
            colPositions.Add strPosition, strPosition
            If Err = 0 Then Exit Do
        Loop
        pixel = picImage.Point(X, Y)
        R = pixel And &HFF&
        G = (pixel And &HFF00&) \ &H100&
        B = (pixel And &HFF0000) \ &H10000
        bit1 = (R And Pos_LSB)
        If bit1 Then
            Value = Value Or offset
        End If
        Do
            C = 1
            strPosition = "[" & C & "," & X & "," & Y & "]"
            colPositions.Add strPosition, strPosition
            If Err = 0 Then Exit Do
        Loop
        offset = offset * 2
        bit2 = (G And Pos_LSB)
        If bit2 Then
            Value = Value Or offset
        End If
        Do
            C = 2
            strPosition = "[" & C & "," & X & "," & Y & "]"
            colPositions.Add strPosition, strPosition
            If Err = 0 Then Exit Do
        Loop
        offset = offset * 2
        bit3 = (B And Pos_LSB)
        If bit3 Then
```

```
            Value = Value Or offset
        End If
        offset = offset * 2
    Next i
    'Ahora bit 7 y 8 **********************
    '********************++++ Ultimos 2 dosssss bits
    Do
        X = Int(Rnd * imgWidth)
        Y = Int(Rnd * imgHeight)
        C = 0
        strPosition = "[" & C & "," & X & "," & Y & "]"
        colPositions.Add strPosition, strPosition
        If Err = 0 Then Exit Do
     Loop
     pixel = picImage.Point(X, Y)
     R = pixel And &HFF&
     G = (pixel And &HFF00&) \ &H100&
     B = (pixel And &HFF0000) \ &H10000
     bit7 = (R And Pos_LSB)
     If bit7 Then
            Value = Value Or offset
     End If
     Do
            C = 1
            strPosition = "[" & C & "," & X & "," & Y & "]"
            colPositions.Add strPosition, strPosition
            If Err = 0 Then Exit Do
     Loop
     offset = offset * 2
     bit8 = (G And Pos_LSB)
     If bit8 Then
            Value = Value Or offset
     End If
     'Terminacion bit 7 y 8 ************************
 End If 'Color_RGB = 3 Then ' 3 Colores
 DecodeByte = Value
End Function

Private Function CalculaHash(ByVal password As String) As Long

    Dim Value As Long
    Dim ch As Long
    Dim shift1 As Long
    Dim shift2 As Long
    Dim i As Integer
    Dim str_len As Integer

    shift1 = 3
```

```
    shift2 = 17
    str_len = Len(password)
    For i = 1 To str_len
        ch = Asc(Mid$(password, i, 1))
        Value = Value Xor (ch * 2 ^ shift1)
        Value = Value Xor (ch * 2 ^ shift2)
        shift1 = (shift1 + 7) Mod 19
        shift2 = (shift2 + 13) Mod 23
    Next i
    CalculaHash = Value
End Function

Private Sub Check4_Click()
 If Check4.Value = 1 Then
    Encrypt txtMessage.Text, txtMessage
 End If
End Sub

Private Sub Check5_Click()
 If Check5.Value = 1 Then
    Decrypt txtMessage.Text, txtMessage
 End If
End Sub

Private Sub cmdEncode_Click()
    'On Error  Resume  Next
    Dim Long_Hash As String
    Dim strMessage As String
    Dim caracter As String
    Dim i As Long
    Dim message_length As Long
    Dim long_msg As Long
    Dim V_Hash As Long
    Dim OpenForms
    'Kx = 10: Ky = 10
    If imgLoaded = False Then
        MsgBox "Primero debe cargar una imagen!"
        Exit Sub
    End If
    If Option1.Value Then
       Pos_LSB = &H7F ' 8 bit
       valor_LSB = 128
    ElseIf Option2.Value Then
       Pos_LSB = &HBF ' 7 bit
       valor_LSB = 64
    ElseIf Option3.Value Then
       Pos_LSB = &HDF ' 6 bit
       valor_LSB = 32
```

```
    ElseIf Option4.Value Then
       Pos_LSB = &HEF ' 5 bit
       valor_LSB = 16
    ElseIf Option5.Value Then
       Pos_LSB = &HF7 ' 4 bit
       valor_LSB = 8
    ElseIf Option6.Value Then
       Pos_LSB = &HFB ' 3 bit
       valor_LSB = 4
    ElseIf Option7.Value Then
       Pos_LSB = &HFD ' 2 bit
       valor_LSB = 2
    ElseIf Option8.Value Then
       Pos_LSB = &HFE ' 1 bit
       valor_LSB = 1
    End If
    If Check1.Value And Check2.Value = False And Check3.Value = False Then
       Color_RGB = 1
       Color = 1 ' Red
    ElseIf Check1.Value = False And Check2.Value And Check3.Value = False
Then
       Color_RGB = 1
       Color = 2 'Green
    ElseIf Check1.Value = False And Check2.Value = False And Check3.Value
Then
       Color_RGB = 1
       Color = 3 'Blue
    ElseIf Check1.Value And Check2.Value And Check3.Value = False Then
       Color_RGB = 2
       Color = 4 'RG
    ElseIf Check1.Value And Check2.Value = False And Check3.Value Then
       Color_RGB = 2
       Color = 5 'RB
    ElseIf Check1.Value = False And Check2.Value And Check3.Value Then
       Color_RGB = 2
       Color = 6 'GB
    ElseIf Check1.Value And Check2.Value And Check3.Value Then
       Color_RGB = 3
       Color = 8 'RGB
    ElseIf Check1.Value = False And Check2.Value = False And Check3.Value
= False Then
        MsgBox "Debe seleccionar un Color (RGB)!"
        Exit Sub
    End If
    V_Hash = CalculaHash(CStr(txtPassword.Text))
    ' Initializa randomizer
    Rnd  -1
    Randomize V_Hash
```

```
    Set colPositions = New Collection
    strMessage = CStr(V_Hash)
    message_length = Len(strMessage) 'Longitud del HASH
    EncodeByte Asc(message_length) ' Oculta longitud HASH
    'Oculta el HASH
    For i = 1 To message_length
       EncodeByte Asc(Mid(strMessage, i, 1))
    Next i
    ' Asigna TAG al mensaje
    strMessage = "@ ∂" & txtMessage.Text
    message_length = Len(strMessage)
    message_length = Len(CStr(message_length))
    long_msg = Len(strMessage)
    'Ocultar longitud del mensaje ejemplo 9789
    For i = 1 To message_length ' Numero de caracteres de la long del
mensaje
     EncodeByte Asc(Mid(CStr(long_msg), i, 1))
    Next i
    ' Ocultar el mensaje
    Timer1.Enabled = True
    Label3.Caption = "Caracteres a procesar: " & long_msg
    For i = 1 To long_msg
        EncodeByte Asc(Mid(strMessage, i, 1))
        If (i Mod 500) = 0 Then
          Label4.Caption = "Bytes procesados: " & i
          OpenForms = DoEvents
        End If
    Next i
    While colPositions.Count
        colPositions.Remove 1
    Wend
    Set colPositions = Nothing
    ' Update control picture
    picImage.Picture = picImage.Image
    cmdSave.Enabled = True
    MsgBox "Proceso Exitoso", vbExclamation + vbOKOnly, "StegoCrypto"
    Timer1.Enabled = False
End Sub

Private Sub cmdDecode_Click()
    Dim strMessage As String
    Dim caracter As String
    Dim i As Integer
    Dim message_length As Integer
    Dim V_Hash As Long
    If imgLoaded = False Then
        MsgBox "Primero debe cargar una imagen!"
        Exit Sub
```

```
    End If
    If Option1.Value Then
       Pos_LSB = &H80 ' 8 bit
    ElseIf Option2.Value Then
       Pos_LSB = &H40 ' 7 bit
    ElseIf Option3.Value Then
       Pos_LSB = &H20 ' 6 bit
    ElseIf Option4.Value Then
       Pos_LSB = &H10 ' 5 bit
    ElseIf Option5.Value Then
       Pos_LSB = &H8 ' 4 bit
    ElseIf Option6.Value Then
       Pos_LSB = &H4 ' 3 bit
    ElseIf Option7.Value Then
       Pos_LSB = &H2 ' 2 bit
    ElseIf Option8.Value Then
       Pos_LSB = &H1 ' 1 bit
    End If
    If Check1.Value And Check2.Value = False And Check3.Value = False Then
       Color_RGB = 1
       Color = 1 ' Red
    ElseIf Check1.Value = False And Check2.Value And Check3.Value = False
Then
       Color_RGB = 1
       Color = 2 'Green
    ElseIf Check1.Value = False And Check2.Value = False And Check3.Value
Then
       Color_RGB = 1
       Color = 3 'Blue
    ElseIf Check1.Value And Check2.Value And Check3.Value = False Then
       Color_RGB = 2
       Color = 4 'RG
    ElseIf Check1.Value And Check2.Value = False And Check3.Value Then
       Color_RGB = 2
       Color = 5 'RB
    ElseIf Check1.Value = False And Check2.Value And Check3.Value Then
       Color_RGB = 2
       Color = 6 'GB
    ElseIf Check1.Value And Check2.Value And Check3.Value Then
       Color_RGB = 3
       Color = 8 'RGB
    ElseIf Check1.Value = False And Check2.Value = False And Check3.Value
= False Then
        MsgBox "Debe seleccionar un Color (RGB)!"
        Exit Sub
    End If
    V_Hash = CalculaHash(CStr(txtPassword.Text))
    'Initializa randomizer
```

```
    Rnd  -1
    Randomize V_Hash
    Set colPositions = New Collection
    'leer tamaÒo del HASH
    caracter = Chr(DecodeByte)
    Select Case caracter
    Case "1", "2", "3", "4", "5", "6", "7", "8", "9":
       message_length = CInt(caracter)
    Case Else:
      MsgBox "No existe Mensaje Secreto", vbQuestion + vbOKOnly,
"StegoCrypto"
      Exit Sub
    End Select
    strMessage = ""
    Label3.Caption = "": Label4.Caption = ""
    For i = 1 To message_length 'Ciclo para Leer el HASH
       strMessage = strMessage & Chr(DecodeByte)
    Next
    'Compara si el HASH almacenado corresponde con la clave de descifrado
    If V_Hash = CLng(strMessage) Then
        'Si es la clave Ahora extrae el tamaÒo del texto oculto
        strMessage = ""
        i = 1
        While caracter <> "@"
          caracter = Chr(DecodeByte)
          strMessage = strMessage & caracter
          i = i + 1
        Wend
        'Extraer texto oculto
        i = Len(strMessage)
        strMessage = Left(strMessage, i - 1)
        message_length = CLng(strMessage)
        Label3.Caption = "Caracteres a procesar Func HASH: " &
message_length
        strMessage = ""
        For i = 2 To message_length
         strMessage = strMessage & Chr(DecodeByte)
        Next
        txtMessage.Text = Mid(strMessage, 3)
        MsgBox "Proceso Exitoso", vbExclamation + vbOKOnly, "StegoCrypto"
    Else
        MsgBox "Verifique la contraseÒa", vbQuestion + vbOKOnly,
"StegoCrypto"
    End If
    While colPositions.Count
        colPositions.Remove 1
    Wend
    Set colPositions = Nothing
```

```
End Sub

Private Sub cmdLoad_Click()
    Dim sFilename As String
    ' Show dialog
    With CDialog
        .DialogTitle = "Abrir Imagen"
        '.Filter = "Windows Bitmap (*.bmp)|*.bmp|CompuServe Graphics
Interchange (*.jpeg)|*.jpeg"
        .Filter = "Todos los archivos de imagen|*.bmp;*.gif;*.jpg"
        .ShowOpen
        sFilename = .FileName
    End With
    ' Cargar Imagen
    If sFilename <> "" Then
        picImage.Picture = LoadPicture(sFilename)
        imgWidth = picImage.Picture.Width
        imgHeight = picImage.Picture.Height
        If    imgWidth    >    picImage.ScaleWidth    Then    imgWidth    =
picImage.ScaleWidth
        If   imgHeight   >   picImage.ScaleHeight   Then   imgHeight   =
picImage.ScaleHeight
        imgLoaded = True
        cmdEncode.Enabled = True
        cmdDecode.Enabled = True
        Img1.Picture = LoadPicture(sFilename)
         valor_X = imgWidth
         valor_Y = imgHeight
    End If
End Sub

Private Sub cmdQuit_Click()
  Dim rta As VbMsgBoxResult
   rta = MsgBox("Desea salir de la aplicaciÛn?", vbYesNo Or vbQuestion,
"StegoCrypto")
   If rta = vbYes Then End

End Sub

Private Sub cmdSave_Click()
  Dim sFilename As String
    ' Visualizar el Commondialog
    With CDialog
        .DialogTitle = "Save image"
        '.Filter = "Windows Bitmap (*.bmp)|*.bmp"
        .Filter = "Jpeg (*.jpg)|*.jpg"
        .DefaultExt = ".jpg"
        .ShowSave
```

```
        sFilename = .FileName
    End With
    ' Guardar Estego Objeto
    If sFilename <> "" Then
        SavePicture picImage.Picture, sFilename
    End If
End Sub


Private Sub Command1_Click()
   FrmAcercaDe.Show vbModal
End Sub

Private Sub Timer1_Timer()

    'If frmMain.WindowState = vbNormal Then
    'frmMain.picImage.Picture = picImage.Image
    'MsgBox "Procesando!", vbOKOnly, "StegoCrypto"
    'End If
    picImage.Refresh
End Sub

Private Sub txtMessage_Change()
 Dim long_Text As Long
   long_Text = Len(txtMessage.Text)
   Label3.Caption = "Caracteres a procesar: " & long_Text
End Sub
```

ANEXO 4. CODIGO FUENTE PROTOTIPO IV

```
Option Explicit
Private imgWidth As Long
Private imgHeight As Long
Private valor_X As Long
Private valor_Y As Long
Private imgLoaded As Boolean
Private colPositions As Collection
Private Pos_LSB
Private Color_RGB As Byte
Private Color As Byte
Private valor_LSB As Byte
Dim Kx As Integer: Dim Ky As Integer
Private Const Steganography_Tag = "TAG"

Private Sub EncodeByte(Value As Byte)
    On Error Resume Next
    Dim i As Byte
    Dim offset As Integer
    Dim bit1, bit2, bit3, bit4, bit5, bit6, bit7, bit8 As Byte
    Dim X As Long
    Dim Y As Long
    Dim C As Byte
    Dim strPosition As String
    Dim pixel As Long
    Dim R As Byte
    Dim G As Byte
    Dim B As Byte
    offset = 1
    If Color_RGB = 1 Then ' 1 Color
      For i = 1 To 8
        Do
            X = Kx
            Y = Ky
            C = 0
            strPosition = "[" & C & "," & X & "," & Y & "]"
            colPositions.Add strPosition, strPosition
            If Err = 0 Then Exit Do
        Loop
        pixel = picImage.Point(X, Y)
        R = pixel And &HFF&
        G = (pixel And &HFF00&) \ &H100&
        B = (pixel And &HFF0000) \ &H10000
        If Value And offset Then bit1 = valor_LSB Else bit1 = 0
        Select Case Color
```

```
    Case 1
       R = (R And Pos_LSB) Or bit1
    Case 2
       G = (G And Pos_LSB) Or bit1
    Case 3
       B = (B And Pos_LSB) Or bit1
    End Select
    picImage.PSet (X, Y), RGB(R, G, B)
    offset = offset * 2
    Kx = Kx + (Int(Rnd * 15)) + 15 'Kx = Kx + 1
    If Kx >= valor_X Then
       Kx = 15
       Ky = Ky + (Int(Rnd * 15)) + 15 'Ky = Ky + 1
    End If
  Next i
End If

If Color_RGB = 2 Then ' 2 Colores
 For i = 1 To 4
    Do
        X = Kx
        Y = Ky
        C = 0
        strPosition = "[" & C & "," & X & "," & Y & "]"
        colPositions.Add strPosition, strPosition
        If Err = 0 Then Exit Do
    Loop
    pixel = picImage.Point(X, Y)
    R = pixel And &HFF&
    G = (pixel And &HFF00&) \ &H100&
    B = (pixel And &HFF0000) \ &H10000
    ' Determine wether bit is 0 or 1
    If Value And offset Then bit1 = valor_LSB Else bit1 = 0
    Select Case Color
        Case 4, 5
              R = (R And Pos_LSB) Or bit1 'R = (R And &H7F) Or Bit
        Case 6
              G = (G And Pos_LSB) Or bit1 'G = (G And &H7F) Or Bit
    End Select
    Do
        C = 1
        strPosition = "[" & C & "," & X & "," & Y & "]"
        colPositions.Add strPosition, strPosition
        If Err = 0 Then Exit Do
    Loop
    If Value And offset * 2 Then bit2 = valor_LSB Else bit2 = 0
    Select Case Color
        Case 4
```

```
            G = (G And Pos_LSB) Or bit2 'G = (G And &H7F) Or Bit
        Case 5, 6
            B = (B And Pos_LSB) Or bit2 'B = (B And &H7F) Or Bit
    End Select
    picImage.PSet (X, Y), RGB(R, G, B)
    offset = offset * 2 * 2
    Kx = Kx + (Int(Rnd * 15)) + 15 'Kx = Kx + 1
    If Kx >= valor_X Then
     Kx = 15
     Ky = Ky + (Int(Rnd * 15)) + 15 'Ky = Ky + 1
    End If
 Next i
End If ' Color_RGB = 2 Then ' 2 Colores

If Color_RGB = 3 Then ' 3 Colores
 For i = 1 To 2
    Do
        X = Kx
        Y = Ky
        C = 0
        strPosition = "[" & C & "," & X & "," & Y & "]"
        colPositions.Add strPosition, strPosition
        If Err = 0 Then Exit Do
    Loop
    pixel = picImage.Point(X, Y)
    R = pixel And &HFF&
    G = (pixel And &HFF00&) \ &H100&
    B = (pixel And &HFF0000) \ &H10000
    If Value And offset Then bit1 = valor_LSB Else bit1 = 0
    R = (R And Pos_LSB) Or bit1
    Do
        C = 1
        strPosition = "[" & C & "," & X & "," & Y & "]"
        colPositions.Add strPosition, strPosition
        If Err = 0 Then Exit Do
    Loop
    If Value And offset * 2 Then bit2 = valor_LSB Else bit2 = 0
    G = (G And Pos_LSB) Or bit2
    Do
        C = 2
        strPosition = "[" & C & "," & X & "," & Y & "]"
        colPositions.Add strPosition, strPosition
        If Err = 0 Then Exit Do
    Loop
    If Value And offset * 4 Then bit3 = valor_LSB Else bit3 = 0
    B = (B And Pos_LSB) Or bit3
    picImage.PSet (X, Y), RGB(R, G, B)
    offset = offset * 2 * 2 * 2
```

```
         Kx = Kx + (Int(Rnd * 15)) + 15 'Kx = Kx + 1
         If Kx >= valor_X Then
          Kx = 15
          Ky = Ky + (Int(Rnd * 15)) + 15 'Ky = Ky + 1
         End If
      Next i
      '********************++++ Ultimos 2 dosssss bits
      Do
         X = Kx
         Y = Ky
         C = 0
         strPosition = "[" & C & "," & X & "," & Y & "]"
         colPositions.Add strPosition, strPosition
         If Err = 0 Then Exit Do
      Loop
      pixel = picImage.Point(X, Y)
      R = pixel And &HFF&
      G = (pixel And &HFF00&) \ &H100&
      B = (pixel And &HFF0000) \ &H10000
      If Value And offset Then bit1 = valor_LSB Else bit1 = 0
      R = (R And Pos_LSB) Or bit1
      Do
             C = 1
             strPosition = "[" & C & "," & X & "," & Y & "]"
             colPositions.Add strPosition, strPosition
             If Err = 0 Then Exit Do
      Loop
      If Value And offset * 2 Then bit2 = valor_LSB Else bit2 = 0
      G = (G And Pos_LSB) Or bit2
      Kx = Kx + (Int(Rnd * 15)) + 15 'Kx = Kx + 1
      If Kx >= valor_X Then
          Kx = 15
          Ky = Ky + (Int(Rnd * 15)) + 15 'Ky = Ky + 1
      End If
      picImage.PSet (X, Y), RGB(R, G, B)
     End If ' Color_RGB = 3 Then ' 3 Colores
End Sub

Private Function DecodeByte() As Integer
   On Error Resume Next
    Dim Value As Integer
    Dim i As Byte
    Dim offset As Integer
    Dim bit1, bit2, bit3, bit4, bit5, bit6, bit7, bit8 As Byte
    Dim X As Long
    Dim Y As Long
    Dim C As Byte
    Dim strPosition As String
```

```
   Dim pixel As Long
   Dim R As Byte
   Dim G As Byte
   Dim B As Byte
   Value = 0
   offset = 1
  If Color_RGB = 1 Then ' 1 Color
     For i = 1 To 8
       Do
           X = Kx
           Y = Ky
           C = 0
           strPosition = "[" & C & "," & X & "," & Y & "]"
           colPositions.Add strPosition, strPosition
           If Err = 0 Then Exit Do
       Loop
       pixel = picImage.Point(X, Y)
       R = pixel And &HFF&
       G = (pixel And &HFF00&) \ &H100&
       B = (pixel And &HFF0000) \ &H10000
       Select Case Color
       Case 1
          bit1 = (R And Pos_LSB)
       Case 2
          bit1 = (G And Pos_LSB)
       Case 3
          bit1 = (B And Pos_LSB)
       End Select
       If bit1 Then
           Value = Value Or offset
       End If
       offset = offset * 2
       Kx = Kx + (Int(Rnd * 15)) + 15 'Kx = Kx + 1
       If Kx >= valor_X Then
          Kx = 15
          Ky = Ky + (Int(Rnd * 15)) + 15 'Ky = Ky + 1
       End If
     Next i
  End If
'*****************************Inicio de 2 colores
If Color_RGB = 2 Then ' 2 Colores
    For i = 1 To 4
       Do
           X = Kx
           Y = Ky
           C = 0
           strPosition = "[" & C & "," & X & "," & Y & "]"
           colPositions.Add strPosition, strPosition
```

```
            If Err = 0 Then Exit Do
        Loop
        pixel = picImage.Point(X, Y)
        R = pixel And &HFF&
        G = (pixel And &HFF00&) \ &H100&
        B = (pixel And &HFF0000) \ &H10000
        Select Case Color
        Case 4, 5
           bit1 = (R And Pos_LSB)
        Case 6
           bit1 = (G And Pos_LSB)
        End Select
        If bit1 Then
            Value = Value Or offset
        End If
        Do
            C = 1
            strPosition = "[" & C & "," & X & "," & Y & "]"
            colPositions.Add strPosition, strPosition
            If Err = 0 Then Exit Do
        Loop
        offset = offset * 2
        Select Case Color
        Case 4
           bit2 = (G And Pos_LSB) 'G = (G And &H7F) Or Bit
        Case 5, 6
           bit2 = (B And Pos_LSB) 'B = (B And &H7F) Or Bit
        End Select
        If bit2 Then
            Value = Value Or offset
        End If
        offset = offset * 2
        Kx = Kx + (Int(Rnd * 15)) + 15 'Kx = Kx + 1
        If Kx >= valor_X Then
           Kx = 15
           Ky = Ky + (Int(Rnd * 15)) + 15 'Ky = Ky + 1
        End If
    Next i
   End If ' Color_RGB = 2 Then ' 2 Colores
''''''''''''**********++++++++++++++++++++++++++ Fin de 2 colores

If Color_RGB = 3 Then ' 3 Colores
   For i = 1 To 2
        Do
            X = Kx
            Y = Ky
            C = 0
            strPosition = "[" & C & "," & X & "," & Y & "]"
```

```
        colPositions.Add strPosition, strPosition
        If Err = 0 Then Exit Do
    Loop
    pixel = picImage.Point(X, Y)
    R = pixel And &HFF&
    G = (pixel And &HFF00&) \ &H100&
    B = (pixel And &HFF0000) \ &H10000
    bit1 = (R And Pos_LSB)
    If bit1 Then
        Value = Value Or offset
    End If
    Do
        C = 1
        strPosition = "[" & C & "," & X & "," & Y & "]"
        colPositions.Add strPosition, strPosition
        If Err = 0 Then Exit Do
    Loop
    offset = offset * 2
    bit2 = (G And Pos_LSB)
    If bit2 Then
        Value = Value Or offset
    End If
    Do
        C = 2
        strPosition = "[" & C & "," & X & "," & Y & "]"
        colPositions.Add strPosition, strPosition
        If Err = 0 Then Exit Do
    Loop
    offset = offset * 2
    bit3 = (B And Pos_LSB)
    If bit3 Then
        Value = Value Or offset
    End If
    Kx = Kx + (Int(Rnd * 15)) + 15 'Kx = Kx + 1
    If Kx >= valor_X Then
     Kx = 15
     Ky = Ky + (Int(Rnd * 15)) + 15 'Ky = Ky + 1
    End If
    offset = offset * 2
Next i
'Ahora bit 7 y 8 **********************
'********************++++ Ultimos 2 dosssss bits
Do
    X = Kx
    Y = Ky
    C = 0
    strPosition = "[" & C & "," & X & "," & Y & "]"
    colPositions.Add strPosition, strPosition
```

```
        If Err = 0 Then Exit Do
    Loop
    pixel = picImage.Point(X, Y)
    R = pixel And &HFF&
    G = (pixel And &HFF00&) \ &H100&
    B = (pixel And &HFF0000) \ &H10000
    bit7 = (R And Pos_LSB)
    If bit7 Then
           Value = Value Or offset
    End If
    Do
           C = 1
           strPosition = "[" & C & "," & X & "," & Y & "]"
           colPositions.Add strPosition, strPosition
           If Err = 0 Then Exit Do
    Loop
    offset = offset * 2
    bit8 = (G And Pos_LSB)
    If bit8 Then
           Value = Value Or offset
    End If
    Kx = Kx + (Int(Rnd * 15)) + 15 'Kx = Kx + 1
    If Kx >= valor_X Then
        Kx = 15
        Ky = Ky + (Int(Rnd * 15)) + 15 'Ky = Ky + 1
    End If
    'Terminacion bit 7 y 8 ************************
 End If 'Color_RGB = 3 Then ' 3 Colores
 DecodeByte = Value
End Function

Private Function CalculaHash(ByVal password As String) As Long

    Dim Value As Long
    Dim ch As Long
    Dim shift1 As Long
    Dim shift2 As Long
    Dim i As Integer
    Dim str_len As Integer
    shift1 = 3
    shift2 = 17
    str_len = Len(password)
    For i = 1 To str_len
        ch = Asc(Mid$(password, i, 1))
        Value = Value Xor (ch * 2 ^ shift1)
        Value = Value Xor (ch * 2 ^ shift2)
        shift1 = (shift1 + 7) Mod 19
        shift2 = (shift2 + 13) Mod 23
```

```
    Next i
    CalculaHash = Value
End Function

Private Sub Check4_Click()
 If Check4.Value = 1 Then
    Encrypt txtMessage.Text, txtMessage
 End If
End Sub

Private Sub Check5_Click()
 If Check5.Value = 1 Then
    Decrypt txtMessage.Text, txtMessage
 End If
End Sub

Private Sub cmdEncode_Click()
    'On Error  Resume  Next
    Dim Long_Hash As String
    Dim strMessage As String
    Dim caracter As String
    Dim i As Long
    Dim message_length As Long
    Dim long_msg As Long
    Dim V_Hash As Long
    Dim OpenForms
    Kx = 10: Ky = 10
    If imgLoaded = False Then
        MsgBox "Primero debe cargar una imagen!"
        Exit Sub
    End If
    If Option1.Value Then
       Pos_LSB = &H7F ' 8 bit
       valor_LSB = 128
    ElseIf Option2.Value Then
       Pos_LSB = &HBF ' 7 bit
       valor_LSB = 64
    ElseIf Option3.Value Then
       Pos_LSB = &HDF ' 6 bit
       valor_LSB = 32
    ElseIf Option4.Value Then
       Pos_LSB = &HEF ' 5 bit
       valor_LSB = 16
    ElseIf Option5.Value Then
       Pos_LSB = &HF7 ' 4 bit
       valor_LSB = 8
    ElseIf Option6.Value Then
       Pos_LSB = &HFB ' 3 bit
```

```
       valor_LSB = 4
    ElseIf Option7.Value Then
       Pos_LSB = &HFD ' 2 bit
       valor_LSB = 2
    ElseIf Option8.Value Then
       Pos_LSB = &HFE ' 1 bit
       valor_LSB = 1
    End If
    If Check1.Value And Check2.Value = False And Check3.Value = False Then
       Color_RGB = 1
       Color = 1 ' Red
    ElseIf Check1.Value = False And Check2.Value And Check3.Value = False
Then
       Color_RGB = 1
       Color = 2 'Green
    ElseIf Check1.Value = False And Check2.Value = False And Check3.Value
Then
       Color_RGB = 1
       Color = 3 'Blue
    ElseIf Check1.Value And Check2.Value And Check3.Value = False Then
       Color_RGB = 2
       Color = 4 'RG
    ElseIf Check1.Value And Check2.Value = False And Check3.Value Then
       Color_RGB = 2
       Color = 5 'RB
    ElseIf Check1.Value = False And Check2.Value And Check3.Value Then
       Color_RGB = 2
       Color = 6 'GB
    ElseIf Check1.Value And Check2.Value And Check3.Value Then
       Color_RGB = 3
       Color = 8 'RGB
    ElseIf Check1.Value = False And Check2.Value = False And Check3.Value
= False Then
        MsgBox "Debe seleccionar un Color (RGB)!"
        Exit Sub
    End If
    V_Hash = CalculaHash(CStr(txtPassword.Text))
    ' Initialize randomizer
    Rnd  -1
    Randomize V_Hash
    Set colPositions = New Collection
    strMessage = CStr(V_Hash)
    message_length = Len(strMessage) 'Longitud del HASH
    EncodeByte Asc(message_length) ' Oculta longitud HASH
    'Oculta el HASH
    For i = 1 To message_length
       EncodeByte Asc(Mid(strMessage, i, 1))
    Next i
```

```
    ' Asigna TAG al mensaje
    strMessage = "@ ∂" & txtMessage.Text
    message_length = Len(strMessage)
    message_length = Len(CStr(message_length))
    long_msg = Len(strMessage)
    'Ocultar longitud del mensaje ejemplo 9789
    For i = 1 To message_length ' Numero de caracteres de la long del
mensaje
     EncodeByte Asc(Mid(CStr(long_msg), i, 1))
    Next i
    ' Ocultar el mensaje
    Timer1.Enabled = True
    Label3.Caption = "Caracteres a procesar: " & long_msg
    For i = 1 To long_msg
        EncodeByte Asc(Mid(strMessage, i, 1))
        If (i Mod 500) = 0 Then
          Label4.Caption = "Bytes procesados: " & i
          OpenForms = DoEvents
        End If
    Next i
    While colPositions.Count
        colPositions.Remove 1
    Wend
    Set colPositions = Nothing
    ' Update control picture
    picImage.Picture = picImage.Image
    cmdSave.Enabled = True
    MsgBox "Proceso Exitoso", vbExclamation + vbOKOnly, "StegoCrypto"
    Timer1.Enabled = False
End Sub

Private Sub cmdDecode_Click()
    Dim strMessage As String
    Dim caracter As String
    Dim i As Integer
    Dim message_length As Integer
    Dim V_Hash As Long
    If imgLoaded = False Then
        MsgBox "Primero debe cargar una imagen!"
        Exit Sub
    End If
    If Option1.Value Then
       Pos_LSB = &H80 ' 8 bit
    ElseIf Option2.Value Then
       Pos_LSB = &H40 ' 7 bit
    ElseIf Option3.Value Then
       Pos_LSB = &H20 ' 6 bit
    ElseIf Option4.Value Then
```

```
       Pos_LSB = &H10 ' 5 bit
    ElseIf Option5.Value Then
       Pos_LSB = &H8 ' 4 bit
    ElseIf Option6.Value Then
       Pos_LSB = &H4 ' 3 bit
    ElseIf Option7.Value Then
       Pos_LSB = &H2 ' 2 bit
    ElseIf Option8.Value Then
       Pos_LSB = &H1 ' 1 bit
    End If
    If Check1.Value And Check2.Value = False And Check3.Value = False Then
       Color_RGB = 1
       Color = 1 ' Red
    ElseIf Check1.Value = False And Check2.Value And Check3.Value = False
Then
       Color_RGB = 1
       Color = 2 'Green
    ElseIf Check1.Value = False And Check2.Value = False And Check3.Value
Then
       Color_RGB = 1
       Color = 3 'Blue
    ElseIf Check1.Value And Check2.Value And Check3.Value = False Then
       Color_RGB = 2
       Color = 4 'RG
    ElseIf Check1.Value And Check2.Value = False And Check3.Value Then
       Color_RGB = 2
       Color = 5 'RB
    ElseIf Check1.Value = False And Check2.Value And Check3.Value Then
       Color_RGB = 2
       Color = 6 'GB
    ElseIf Check1.Value And Check2.Value And Check3.Value Then
       Color_RGB = 3
       Color = 8 'RGB
    ElseIf Check1.Value = False And Check2.Value = False And Check3.Value
= False Then
        MsgBox "Debe seleccionar un Color (RGB)!"
        Exit Sub
    End If
    V_Hash = CalculaHash(CStr(txtPassword.Text))
    ' Initialize randomizer
    Rnd  -1
    Randomize V_Hash
    Set colPositions = New Collection
    Kx = 10: Ky = 10
    'leer tamaÒo del HASH
    caracter = Chr(DecodeByte)
    Select Case caracter
    Case "1", "2", "3", "4", "5", "6", "7", "8", "9":
```

```
       message_length = CInt(caracter)
    Case Else:
      MsgBox "No existe Mensaje Secreto", vbQuestion + vbOKOnly,
"StegoCrypto"
      Exit Sub
    End Select
    strMessage = ""
    Label3.Caption = "": Label4.Caption = ""
    For i = 1 To message_length 'Ciclo para Leer el HASH
       strMessage = strMessage & Chr(DecodeByte)
    Next
    'Compara si el HASH almacenado corresponde con la clave de descifrado
    If V_Hash = CLng(strMessage) Then
        'Si es la clave Ahora extrae el tamaÒo del texto oculto
        strMessage = ""
        i = 1
        While caracter <> "@"
          caracter = Chr(DecodeByte)
          strMessage = strMessage & caracter
          i = i + 1
        Wend
        'Extraer texto oculto
        i = Len(strMessage)
        strMessage = Left(strMessage, i - 1)
        message_length = CLng(strMessage)
        Label3.Caption = "Caracteres a procesar Func HASH: " &
message_length
        strMessage = ""
        For i = 2 To message_length
         strMessage = strMessage & Chr(DecodeByte)
        Next
        txtMessage.Text = Mid(strMessage, 3)
        MsgBox "Proceso Exitoso", vbExclamation + vbOKOnly, "StegoCrypto"
    Else
        MsgBox "Verifique la contraseÒa", vbQuestion + vbOKOnly,
"StegoCrypto"
    End If
    While colPositions.Count
        colPositions.Remove 1
    Wend
    Set colPositions = Nothing
End Sub

Private Sub cmdLoad_Click()
    Dim sFilename As String
    ' Show dialog
    With CDialog
        .DialogTitle = "Abrir Imagen"
```

```
        '.Filter = "Windows Bitmap (*.bmp)|*.bmp|CompuServe Graphics
Interchange (*.jpeg)|*.jpeg"
        .Filter = "Todos los archivos de imagen|*.bmp;*.gif;*.jpg"
        .ShowOpen
        sFilename = .FileName
    End With
    ' Cargar Imagen
    If sFilename <> "" Then
        picImage.Picture = LoadPicture(sFilename)
        imgWidth = picImage.Picture.Width
        imgHeight = picImage.Picture.Height
        If   imgWidth   >   picImage.ScaleWidth   Then   imgWidth   =
picImage.ScaleWidth
        If   imgHeight   >   picImage.ScaleHeight   Then   imgHeight   =
picImage.ScaleHeight
        imgLoaded = True
        cmdEncode.Enabled = True
        cmdDecode.Enabled = True
        Img1.Picture = LoadPicture(sFilename)
         valor_X = imgWidth
         valor_Y = imgHeight
    End If
End Sub

Private Sub cmdQuit_Click()
  Dim rta As VbMsgBoxResult
   rta = MsgBox("Desea salir de la aplicaciÛn?", vbYesNo Or vbQuestion,
"StegoCrypto")
   If rta = vbYes Then End

End Sub

Private Sub cmdSave_Click()
  Dim sFilename As String
    ' Visualizar el Commondialog
    With CDialog
        .DialogTitle = "Save image"
        '.Filter = "Windows Bitmap (*.bmp)|*.bmp"
        .Filter = "Jpeg (*.jpg)|*.jpg"
        .DefaultExt = ".jpg"
        .ShowSave
        sFilename = .FileName
    End With
    ' Guardar Estego Objeto
    If sFilename <> "" Then
        SavePicture picImage.Picture, sFilename
    End If
End Sub
```

```
Private Sub Command1_Click()
   FrmAcercaDe.Show vbModal
End Sub

Private Sub Timer1_Timer()

    'If frmMain.WindowState = vbNormal Then
    'frmMain.picImage.Picture = picImage.Image
    'MsgBox "Procesando!", vbOKOnly, "StegoCrypto"
    'End If
    picImage.Refresh
End Sub

Private Sub txtMessage_Change()
 Dim long_Text As Long
   long_Text = Len(txtMessage.Text)
   Label3.Caption = "Caracteres a procesar: " & long_Text
End Sub
```