Transmisión de mensajes en archivos de imágenes y texto usando esteganografía

Proyecto Terminal. Licenciatura en Ingeniería en Computación

Joel David Villegas Hernández Universidad Autónoma Metropolitana. Azcapotzalco

Matrícula: 204304945

Director de proyecto: Dr. Francisco Javier Zaragoza Martínez

Memoria de Desarrollo Técnico

Transmisión de mensajes en archivos de imágenes y texto usando esteganografía

Transmisión de mensajes en archivos de imágenes y texto usando esteganografía

Proyecto Terminal.
Licenciatura en Ingeniería en Computación

INTRODUCCIÓN

El presente documento es una bitácora de desarrollo y construcción del sistema de transmisión de mensajes en archivos de imágenes y texto usando esteganografía, llevando un registro detallado de la concepción, análisis, diseño y construcción de este sistema.

Esta memoria de desarrollo describe a detalle cada uno de los elementos que componen al sistema, el análisis y el desarrollo de los algoritmos necesarios así como las decisiones de implementación y la estructura lógica del sistema.

Se realiza la descripción del sistema separando sus 2 componentes principales, manipulación de texto e imágenes, y se describe el proceso de ambos.

Otro objetivo que persigue este documento es el de proveer información clara y necesaria para realizar futuras ampliaciones a este sistema como podrían ser la expansión de formatos de imágenes para el transporte de mensajes o el uso de archivos de sonido con el mismo propósito.

Al final del presente documento se anexan los requerimientos de ejecución del sistema y los pasos necesarios para poder utilizarlo de manera correcta, además de presentar el manual de usuario de la aplicación para facilitar el entendimiento y el manejo de la aplicación.

ESTEGANOGRAFÍA EN TEXTO

Análisis.

Los archivos de texto plano son aquellos que están compuestos únicamente por texto sin formato, sólo caracteres. Algunos de los sistemas de codificación más usados para los caracteres de dichos archivos son ASCII y Unicode. Se les conoce también como archivos de texto llano por carecer de información destinada a generar formatos y tipos de letra.

La estructura de un archivo de texto plano consiste en la secuencia de los caracteres que conforman el texto, y concluye mediante el carácter que representa el fin de archivo (EOF por sus siglas en inglés *End Of File*).

Partiendo de éste análisis nos enfrentamos a la necesidad de almacenar información, ajena al archivo de texto, de una manera que no delate la presencia de la misma y que no afecte la información de ninguno de los dos mensajes implicados.

Se procede a definir las entradas que tendrá el sistema: un archivo de texto plano que contendrá el mensaje que queremos ocultar en otro archivo; un archivo de texto plano el cual nos servirá para ocultar el mensaje, se hará referencia a este archivo como archivo de transporte.

Al ser ambos archivos de texto plano, necesitamos encontrar la manera de alterar el archivo transporte sin modificar ninguno de sus caracteres porque esto representaría modificar directamente la información que representa. Por ésta misma razón no es posible insertar los caracteres del archivo de mensaje dentro del texto del archivo de transporte pues resultaría obvio que son ajenos a dicho archivo.

Tampoco es posible insertar bits de manera aleatoria en el contenido del archivo, debido a que modificaría la interpretación por parte de cualquier editor de texto al tratar de abrir el archivo de transporte.

Entonces no podemos cambiar la estructura del archivo, así como tampoco los caracteres que contiene; el único carácter que al ser insertado en un archivo pasa casi desapercibido es el espacio en blanco ( ) representado por el ASCII 20 en hexadecimal.

En un párrafo de texto los espacios en blanco pasan casi desapercibidos durante la lectura de la información; a continuación se insertan dos párrafos de texto, el primero contiene un espacio que separa cada palabra, mientras que el segundo contiene dos espacios entre algunas palabras, distribuidos de manera aleatoria.

Párrafo 1:

El código el carácter espacio, designa al espacio entre palabras, y se produce normalmente por la barra espaciadora de un teclado. Los códigos del 33 al 126 se conocen como caracteres imprimibles, y representan letras, dígitos, signos de puntuación y varios símbolos.

Párrafo 2:

El código el carácter espacio, designa al espacio entre palabras, y se produce normalmente por la barra espaciadora de un teclado. Los códigos del 33 al 126 se conocen como caracteres imprimibles, y representan letras, dígitos, signos de puntuación y varios símbolos.

Ambos párrafos transmiten el mismo mensaje al ser leídos por una persona; el insertar dos espacios en lugar de uno es generalmente ignorado cuando no se tiene una comparación directa con el mismo texto con espaciado sencillo. Es decir, una persona que tiene en su campo visual ambos párrafos puede distinguir diferencias entre ellos; para un usuario que lee el párrafo con espacios dobles, sin tener la referencia del mismo con espaciado sencillo, suele pasar por alto el espaciado en el texto.

En base a esto, se pretende descomponer en binario el ASCII del texto dentro del archivo de mensaje; si el archivo de mensaje es representado en binario, se tiene una secuencia de valores 1 y 0, siendo posible representar el uno como un espacio doble entre dos palabras y el cero como un espacio sencillo.

Podemos concluir que la salida del sistema que realizará dicho proceso será un archivo de texto plano, el cual contendrá la misma información que el archivo de transporte, con la diferencia de que tendrá una estructura de espacios dobles y sencillos distribuidos con respecto al binario del código ASCII del mensaje.

De ésta manera tendremos el archivo de transporte más el mensaje oculto, que se pretende pase desapercibido.

Al ser archivos de texto plano es más difícil esconder información en éste tipo de archivos; sin embargo se desarrolla la aplicación en base a éste análisis y se exponen sus resultados.

El proceso de extracción de mensajes de un archivo de texto plano consiste en leer un archivo (denominado de transporte) y generar el arreglo que simula la representación binaria de un caracter mediante la lectura de espacios sencillos y dobles (0 y 1, respectivamente). Al completar este arreglo se obtendrá el ASCII del caracter que será escrito en el archivo de salida, que representa el mensaje extraído.

Para la inserción y extracción de mensajes se utilizarán dos procesos diferentes e independientes.

Diseño.

En base al análisis del sistema se propone seguir el siguiente diagrama de flujo para el sistema que inserta un mensaje en un archivo de texto plano.

El usuario tiene que interactuar con el sistema, especificando el nombre de los archivos de entrada (transporte y mensaje), así como el nombre del archivo que se generará y contendrá la salida del sistema.

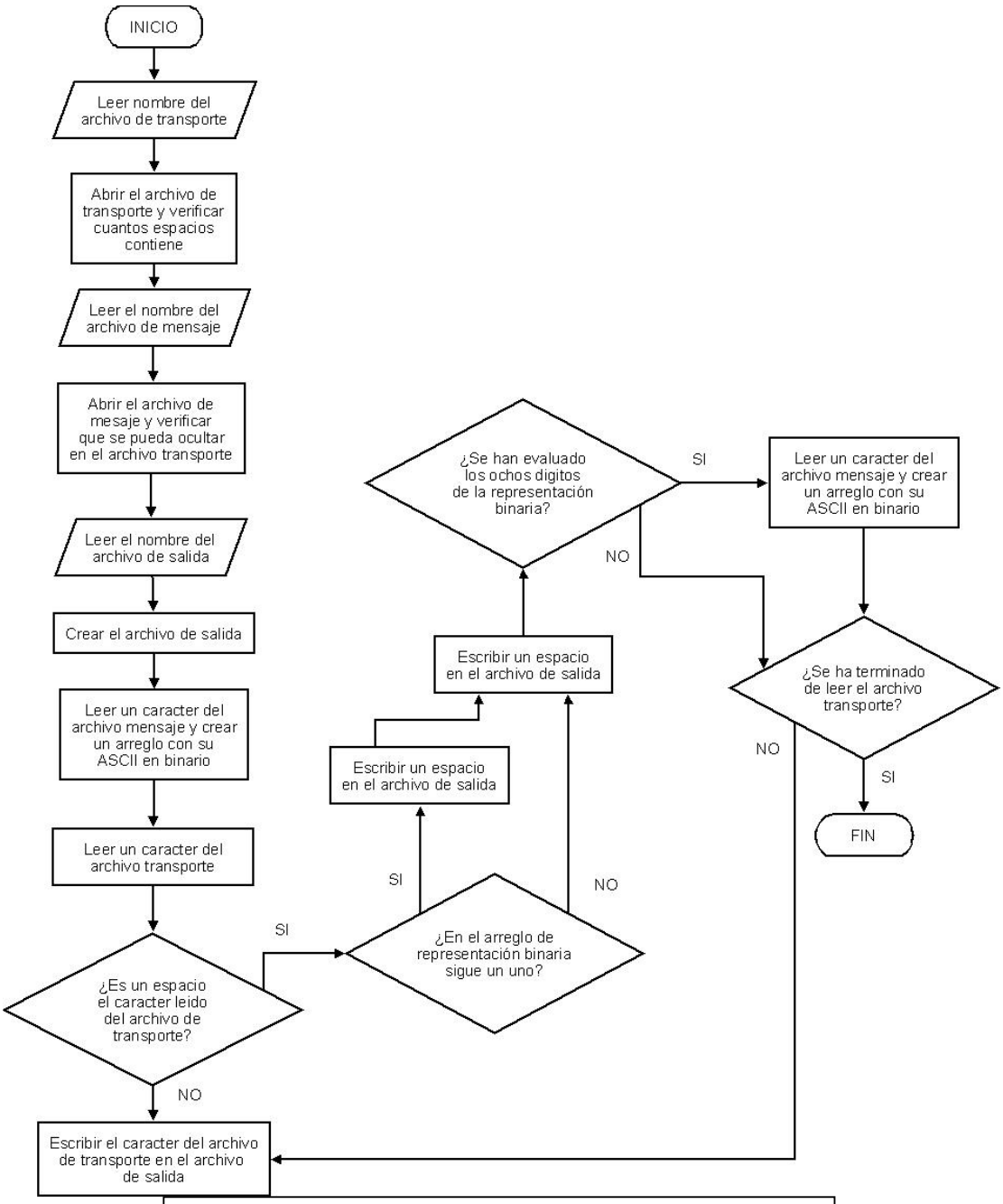


Diagrama de flujo para la inserción de un mensaje en un archivo de texto plano

El flujo mostrado en esta página representa el proceso de extracción de un archivo de mensaje dentro de un archivo de transporte en texto plano.

Utiliza un arreglo de ocho campos donde se almacenan los valores de 1 y 0 (dependiendo de si existe un doble espacio o no), y cuando éste arreglo se llena se puede obtener el ASCII de un caracter y representarlo dentro del archivo de salida, formando de esta manera el mensaje oculto mediante esteganografía en el archivo de transporte y plasmándolo en un archivo de texto plano.

Si un archivo no contiene un mensaje oculto por el algoritmo descrito en este documento, no se podrá extraer un mensaje de manera exitosa.

Es decir que este proceso regresa un intento fallido de extracción de mensajes si encuentra un archivo con un espaciado diferente a espacios sencillos y dobles; además, puede extraer información, que no resulte ser un mensaje oculto, siempre y cuando el archivo de transporte cumpla con las características que se han mencionado.

El mensaje resultado tendrá una codificación ASCII de 8 bits.

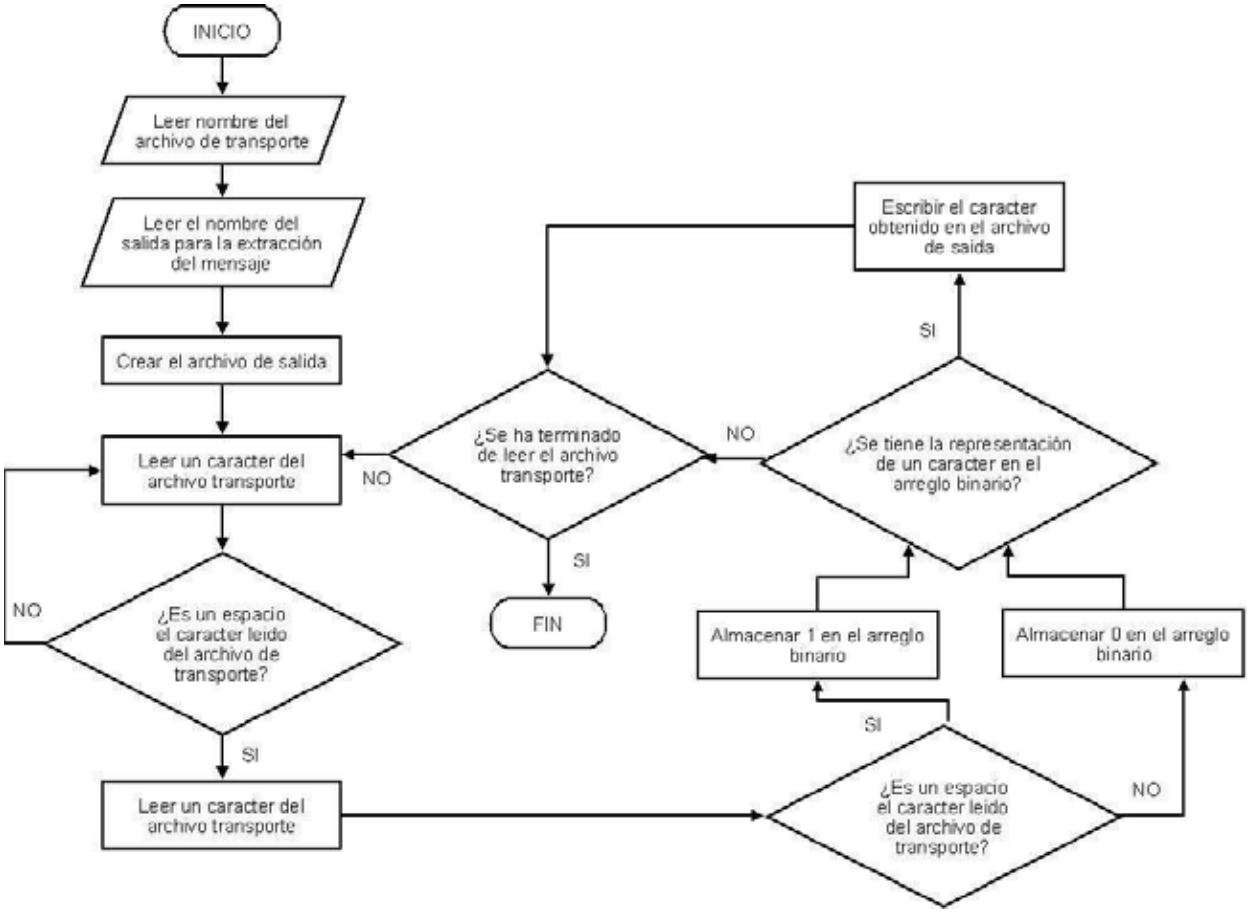


Diagrama de flujo para la extracción de un mensaje en un archivo de texto plano

La estructura del archivo de salida se forma mediante una máquina de estados. Se utiliza la siguiente notación

- q*n* : representa el estado en el que se encuentra el sistema (relación y comportamiento del sistema entre la entrada actual y la anterior).
- s : representa un espacio en blanco.
- ≠s : representa cualquier caracter diferente del espacio en blanco.
- a, b : representa la entrada del sistema, y la respuesta que tendrá (entrada, salida).
- f(a) : es una función que puede regresar 1 ó 0 (que para la salida será considerado un espacio o vacío, respectivamente) que se obtiene del valor binario de la representación ASCII del caracter obtenido del archivo de mensaje.

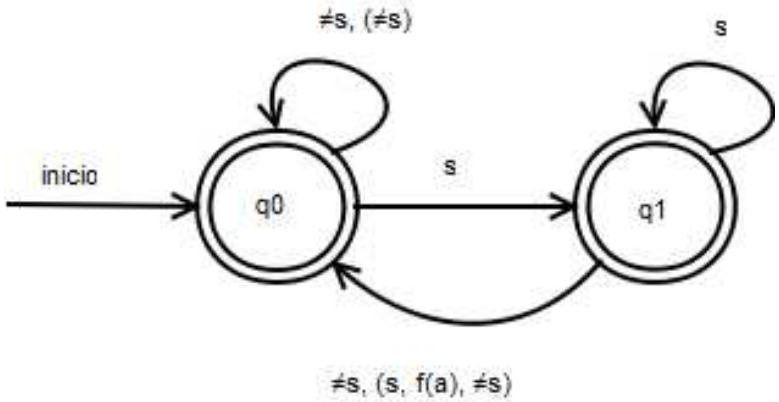
De ésta manera tenemos la siguiente máquina de estados:

La figura indica que tenemos dos estados, al inicio el sistema está esperando entradas; mientras reciba caracteres diferentes al espacio en blanco, manda el mismo a la salida (el archivo de salida) y permanece en el estado q0, si recibe un espacio avanza al siguiente estado (q1) sin mandar respuesta de salida.

En el estado q1 permanecerá el sistema sin responder mientras continúe leyendo espacios en blanco; en cuanto entre un caracter diferente regresa al estado q0, manda a la salida un espacio en blanco (podría representar el mismo espacio que mandó el sistema al estado q1), el valor de retorno de la función f(a) y el valor diferente al espacio que recibió como entrada en este estado.

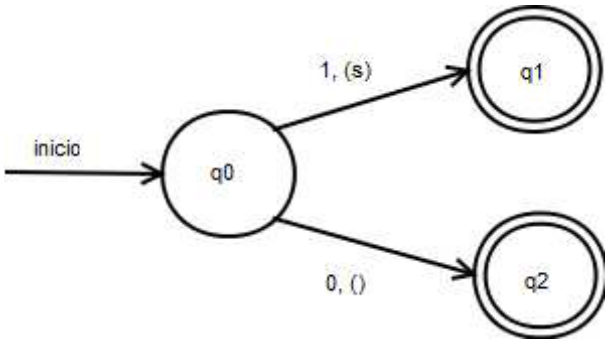
El sistema puede acabar en cualquiera de los dos estados y representa el final de la lectura del archivo de transporte y la escritura del archivo de salida.

Se garantiza que el archivo de salida contendrá la información del archivo de transporte además del espaciado correspondiente al ocultamiento del mensaje.



Máquina de estados para la inserción de mensajes en archivos de texto plano

La figura siguiente representa la función *f(a)*, la cual es utilizada en el proceso de inserción de mensajes en archivos de texto plano.



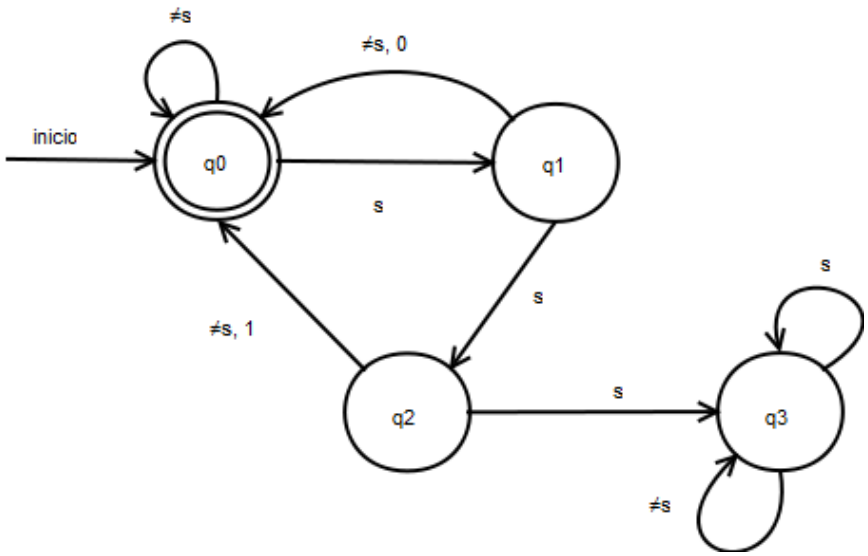
Máquina de estados para la función de inserción de espacios en blanco

Representa una función que, dependiendo de la entrada, regresa un valor específico. La entrada de esta función es la sucesión de valores binarios que representan el ASCII de un caracter (obtenido del archivo de mensaje). En base al valor de retorno de esta función, se decide si insertar un espacio en blanco (representado el valor 1) o ningún valor (simulando un 0).

La figura de la parte inferior describe una máquina de estados, en la cual se basa el proceso de extracción de información insertada en un archivo de texto mediante el algoritmo descrito anteriormente.

Comienza en un estado de aceptación y la entrada es la sucesión de caracteres del archivo transporte; los únicos caracteres que nos importan para la extracción del mensaje son los espacios en blanco, debido a esto, mientras el autómata reciba otro caracter permanecerá en el estado q0; cuando se recibe un espacio en blanco se avanza al estado q1, el cual conduce de nuevo al estado q0 si se encuentra un caracter diferente al espacio y almacena en el arreglo de representación binaria un valor 0. Además el estado q1 conduce al estado q2 cuando recibe un espacio en blanco.

Al estar en el estado q2 (que representa encontrar al menos dos espacios en blanco entre dos palabras) se puede cambiar al sistema al estado q0 si se recibe un



Máquina de estados que describe la extracción de mensajes en archivos de texto plano

caracter diferente al espacio en blanco, pues esto implica forzosamente la existencia de dos espacios en blanco entre dos palabras y esto se procesará en el arreglo de representación binaria con un valor 1.

Sin embargo, si el sistema se encuentra en el estado q2 y se recibe un espacio en blanco (implicaría, al menos, tres espacios en blanco de manera consecutiva), nos conduce al estado q3 (significa un archivo que no contiene un mensaje o que ha sido alterado el contenido del archivo), del cual no se puede salir y terminará en un estado de no aceptación, que para nuestro propósito representa la inexistencia de un mensaje oculto.

Desarrollo.

Se implementó el algoritmo propuesto dentro de una aplicación desarrollada en lenguaje C.

El código fuente tiene como nombre "texto.c", y su funcionamiento se basa en el diagrama de flujo para la inserción de un mensaje en un archivo de texto plano.

Es necesario realizar un proceso de compilación para obtener una aplicación ejecutable mediante el código fuente, el cual fue generado sin utilizar funciones ligadas a un sistema operativo o compilador con el motivo de que pueda funcionar en cualquier equipo que cuente con un compilador de ANSI C.

El programa recibe el archivo de entrada y cuenta la los espacios que se encuentran en el texto (recibe solamente archivos de texto plano), para de esta manera realizar el cálculo de cuantos caracteres podrá ocultar en el archivo propuesto el por el usuario como transporte; posterior a esto recibe el nombre del archivo de entrada y analiza si se podrá ocultar en el archivo transporte o el mensaje resultará truncado.

Posterior a esto, el programa solicita al usuario el nombre deseado para el archivo que contendrá el resultado de insertar el mensaje en el archivo de transporte.

Finalmente el programa realiza el algoritmo ejemplificado en las máquinas de estado para la inserción de mensajes en archivos de texto plano y para la función de inserción de espacios en blanco, dejando la salida del sistema en el archivo indicado por el usuario.

Para el proceso de extracción de mensajes dentro de archivos de texto plano se generó el código "extraetexto.c" en lenguaje C, basando su funcionamiento en el diagrama de flujo para la extracción de mensajes en archivos de texto plano.

Al igual que el proceso de inserción de mensajes, es necesario realizar un proceso de compilación para obtener una aplicación ejecutable a partir del código fuente, generado sin utilizar funciones ligadas a un sistema operativo o compilador con el motivo de que pueda funcionar en cualquier equipo que cuente con un compilador de ANSI C.

El programa recibe el nombre del archivo de transporte del cual se extraerá el mensaje (recibe solamente archivos de texto plano), además del nombre del archivo donde se pretende extraer el mensaje oculto, el cual será una sucesión de caracteres obtenidos en base a la máquina de estados que describe la extracción de mensajes en texto plano.

El programa recorre el archivo transporte en busca de espacios, formando arreglos binarios que, en un conjunto de 8, formaran el código ASCII del mensaje oculto.

El proceso concluye hasta terminar de leer el archivo transporte o hasta formar mediante el arreglo el caracter de fin de archivo (*EOF*).

Pruebas.

Durante la fase de pruebas se obtuvieron resultados satisfactorios y correctos. Sin embargo, en la función de insertar mensaje en texto plano se tenía el siguiente inconveniente: si el archivo transporte contenía más espacios de los necesitados para ocultar el mensaje, el resto del archivo de salida contenía espacios sencillos; esto resultaba contraproducente con el objetivo, debido a que una persona que revisara el archivo resultado, tenía una referencia de texto con espaciado sencillo inmediatamente después del texto con espaciado irregular (sencillo y doble). Esto denotaba una irregularidad en el archivo, por lo que el objetivo deseado de pasar desapercibido el mensaje no se cumplía.

Por esta razón el código de inserción de mensajes tuvo que ser modificado. Se agregó la condición para realizar una inserción de espacios dobles y sencillos, de manera aleatoria, al texto que no era modificado debido a la corta longitud del mensaje a ocultar.

De ésta manera el mensaje concluía con la secuencia que representaba el caracter EOF, pero continuaba con una serie de espacios dobles y sencillos, que no sostenían relación alguna con el contenido del mensaje, sin embargo, lograban disimular la presencia de algún tipo de irregularidad en el texto, puesto que la distribución de espacios en el archivo era constante y disminuía la sospecha de tener un contenido oculto, acercando al sistema al objetivo planteado.

Para el caso de la extracción de mensajes se obtuvieron resultados satisfactorios,

pues la ejecución de dicho programa nos producía como salida un archivo con la copia fiel del mensaje original. Por lo que el módulo no presentó adecuaciones ni defectos inesperados que implicaran un mantenimiento en dicho código.

Para revisar los resultados obtenidos en este desarrollo se debe consultar el Apéndice A.

ESTEGANOGRAFÍA EN IMÁGENES

Análisis.

Para cumplir el objetivo de transmitir información oculta en archivos de imágenes se eligió manipular un tipo de archivos que no utilizara un formato de compresión en su estructura para no alterar el contenido del archivo y mantener constante la información presente.

Para este proyecto se optó por trabajar imágenes en un formato popular para disminuir cualquier sospecha acerca de la irregularidad de utilizar un formato de archivos poco usado. Se decidió analizar el formato *Windows bitmap* (cuya extensión es *.bmp*) para utilizarlo como archivos de transporte (para ocultar y extraer mensajes).

Windows bitmap (*.bmp*) es el formato propio del programa Microsoft Paint, que viene con el sistema operativo Windows. Puede guardar imágenes de 24 bits (16.7 millones de colores), 8 bits (256 colores) y menos.

Los archivos con extensión *.bmp*, en los sistemas operativos Windows, representan las siglas de BitMaP (o también Bit Mapped Picture), que significa mapa de bits. Los archivos de mapas de bits están compuestos por direcciones asociadas a códigos de color, uno para cada cuadro en una matriz de pixeles. Normalmente, se caracterizan por ser muy poco eficientes en su uso de espacio en disco, pero pueden mostrar un buen nivel de calidad. A diferencia de los gráficos vectoriales, al ser reescalados a un tamaño mayor, pierden calidad. Otra desventaja de los archivos BMP es que no son utilizables en páginas

web debido a su gran tamaño en relación a su resolución.

Dependiendo de la profundidad de color que tenga la imagen, cada pixel puede ocupar 1 o varios bytes. Generalmente se suelen transformar en otros formatos, como JPEG (fotografías), GIF o PNG (dibujos y esquemas), los cuales utilizan otros algoritmos para conseguir una mayor compresión (menor tamaño del archivo).

Los archivos comienzan (cabecera o *header*) con las letras 'BM' (0x42 0x4D), que lo identifica con el programa de visualización o edición. En la cabecera también se indica el tamaño de la imagen y con cuántos bytes se representa el color de cada pixel.

El Bitmap de una imagen BMP comienza a leerse desde abajo a arriba, es decir: en una imagen en 24 bits, los primeros 3 bytes corresponden al primer pixel inferior izquierdo.

Para consultar el contenido de la cabecera de cualquier archivo BMP consulte el Apéndice B.

Analizando la información sobre este formato de imágenes, surge el problema de introducir información en las imágenes sin que su presencia sea de fácil detección por la vista de cualquier usuario.

Como acabamos de describir, dentro de la cabecera del archivo se incluye información que describe el tamaño del archivo, la anchura y altura (en pixeles), y el tamaño de la cabecera misma.

Esto nos ayuda a saber con cuántos pixeles cuenta el archivo y la cantidad de bytes que podríamos utilizar a la manera que se considere conveniente.

Se procede a definir las entradas que tendrá el sistema: un archivo de texto plano que contendrá el mensaje que queremos ocultar en otro archivo; y el archivo en formato BMP (que denominaremos archivo de transporte).

La salida del programa consistirá en un archivo en formato BMP, que incluirá el mensaje oculto y la imagen alterada de manera imperceptible a la vista humana.

Para cumplir con el objetivo de mantener el archivo de transporte sin irregularidades en su estructura, el encabezado del archivo se mantendrá igual y no se modificará ancho, alto o calidad del mismo. Como ya se mencionó, cada pixel está formado por una sucesión de bytes y la diferencia de color que se produce en ellos cuando se modifica el bit menos significativo es imperceptible para la vista humana; esto nos permite disponer del bit menos significativo de cada byte, es decir en una imagen en formato BMP de 24 bits, disponemos de 3 bits por pixel para manipularlos de la manera que más nos convenga.

La diferencia de los 2 archivos (original y modificado) es imperceptible, por lo que el usuario no encontrará ninguna diferencia al ver ambos archivos, incluso si los tiene abiertos al mismo tiempo.

En base a esto, se pretende descomponer en binario el ASCII del texto dentro del archivo de mensaje; si el archivo de mensaje es representado en binario, se tiene una secuencia de valores 1 y 0, de manera que este valor será forzado a

aparecer como bit menos significativo en cada byte que se lea.

Podemos concluir que la salida del sistema que realizará dicho proceso será un archivo de imagen en formato BMP, el cual contendrá la misma información del archivo de transporte pero tendrá un conjunto de bits que formaran el código ASCII del mensaje; tendremos el archivo de transporte más el mensaje oculto que, a la vista, pasa desapercibido.

El proceso de extracción de mensajes tendrá como principio recibir un archivo en formato BMP donde previamente se ha insertado un mensaje con el procedimiento anteriormente descrito, para que mediante la lectura del archivo se pueda recuperar el mensaje oculto y generar un archivo de texto plano para vaciar en este el resultado de la extracción.

Se debe considerar que realizar el intento de extracción de un mensaje en algún archivo que no lo contenga regresará un mensaje sin información útil; los mensajes concluyen cuando el archivo sea completamente leído o cuando se encuentre el caracter de fin de archivo (*EOF*).

Podemos concluir que, para tener un resultado satisfactorio, el proceso de extracción de mensajes de un archivo en formato BMP consiste en procesar un archivo (denominado de transporte) y generar el arreglo que simula la representación binaria de un caracter mediante la lectura del bit menos significativo de los bytes que conforman el conjunto de pixeles de la imagen. Al completar este arreglo se obtendrá el ASCII del caracter que será escrito en el archivo de salida que representa el mensaje extraído.

Para la inserción y extracción de mensajes se utilizarán dos procesos diferentes e independientes.

Diseño.

En base al análisis del sistema se propone seguir el siguiente diagrama de flujo para el sistema que inserta un mensaje en un archivo en formato BMP.

El usuario tiene que interactuar con el sistema, especificando el nombre de los archivos de entrada (transporte y mensaje), así como el nombre del archivo que se generará y contendrá la salida del sistema.

El programa se encarga de verificar la capacidad de ocultamiento del archivo de transporte, para evaluar si el mensaje se transmitirá sin problemas. Posteriormente se comienza con la escritura del archivo de salida, manipulando el último bit de cada byte como parte del mensaje, mediante el análisis que ya se ha descrito.

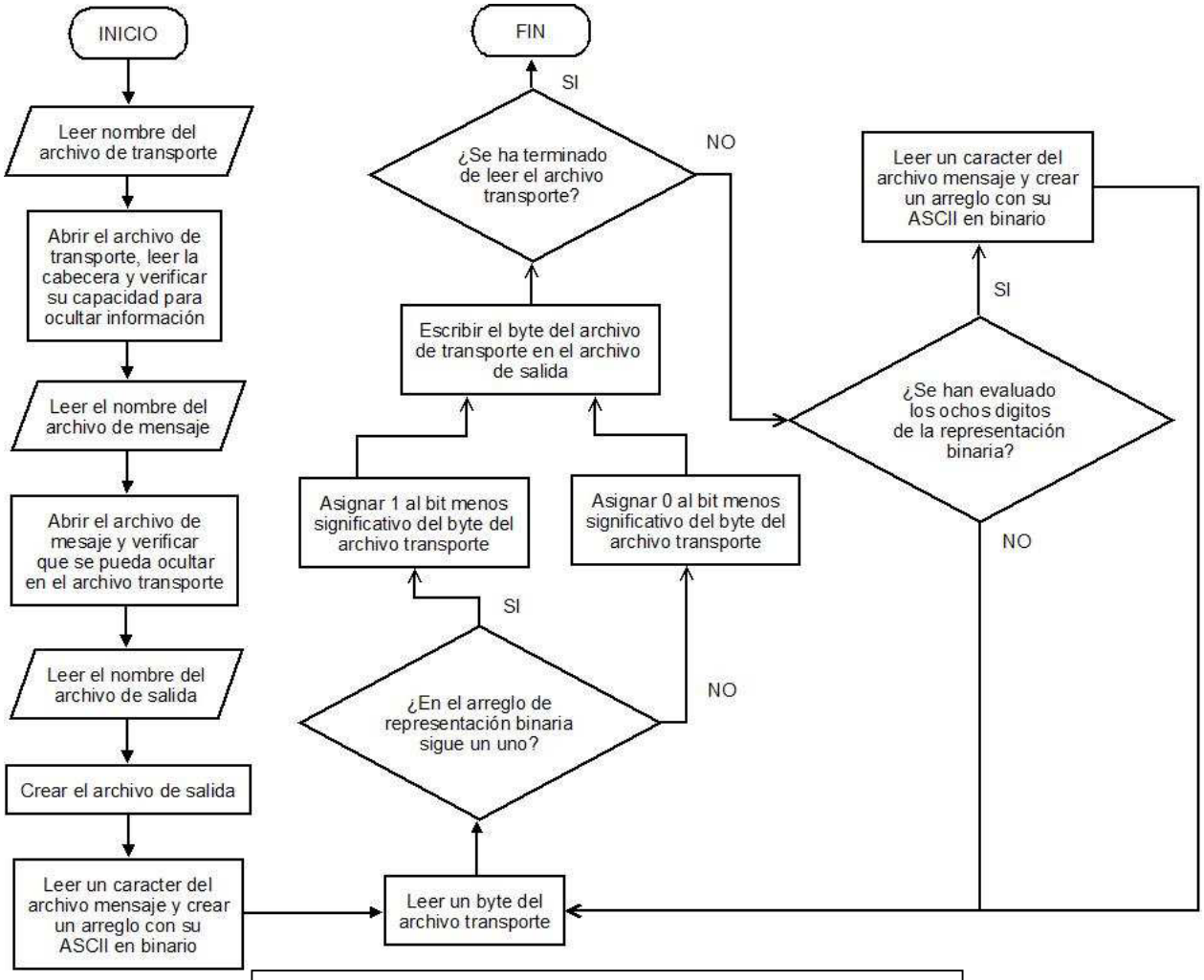


Diagrama de flujo para la inserción de un mensaje en un archivo BMP

El flujo mostrado en esta página representa el proceso de extracción de un mensaje dentro de un archivo de imagen de transporte en formato BMP.

Utiliza un arreglo de ocho campos donde se almacenan los valores de 1 y 0 (dependiendo del valor correspondiente al bit menos significativo de cada byte dentro del conjunto de los pixeles de la imagen), y cuando éste arreglo se llena se puede obtener el ASCII de un caracter y representarlo dentro del archivo de salida, formando de esta manera el mensaje oculto mediante esteganografía en el archivo de transporte y plasmándolo en un archivo de texto plano.

Si un archivo no contiene un mensaje oculto por el algoritmo descrito en este documento, no se podrá extraer un mensaje de manera exitosa.

Es decir, este proceso puede extraer información que no resulte ser un mensaje oculto, siempre y cuando el archivo de transporte cumpla con las características que se han mencionado.

El mensaje oculto tendrá una codificación ASCII de 8 bits.

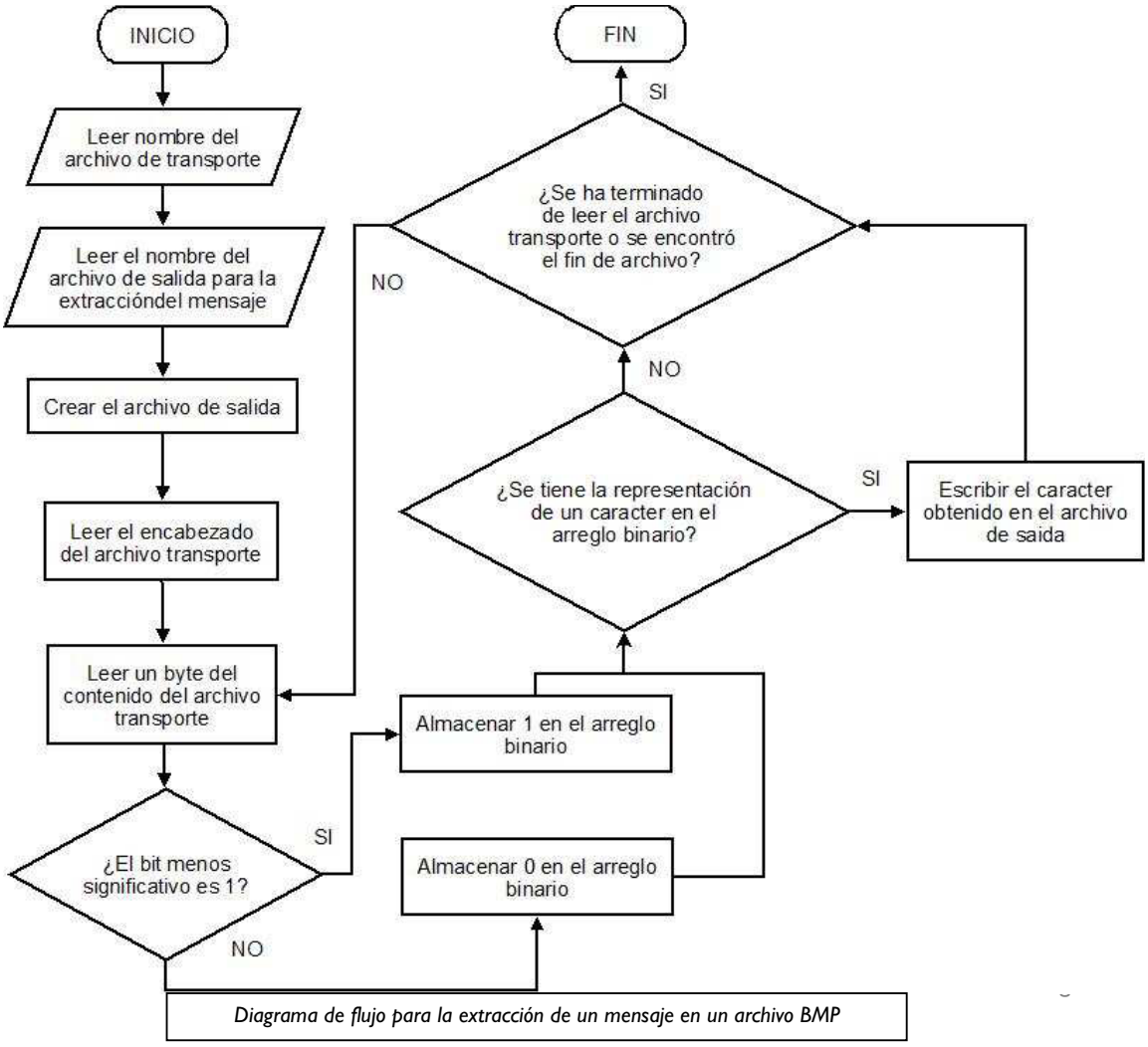


Diagrama de flujo para la extracción de un mensaje en un archivo BMP

Desarrollo.

Se implementó el algoritmo propuesto dentro de una aplicación desarrollada en lenguaje C.

El código fuente correspondiente al proceso de inserción de un mensaje tiene como nombre "imagen.c", y su funcionamiento se basa en el diagrama de flujo para la inserción de un mensaje en un archivo BMP.

Es necesario realizar un proceso de compilación para obtener una aplicación ejecutable mediante el código fuente, el cual fue generado sin utilizar funciones ligadas a un sistema operativo o compilador con el motivo de que pueda funcionar en cualquier equipo que cuente con un compilador de ANSI C.

El programa recibe el archivo de entrada y cuenta la cantidad de bytes que forman los pixeles de la imagen (recibe solamente archivos BMP), para realizar el cálculo de cuantos caracteres podrá ocultar en el archivo propuesto como transporte por el usuario; posterior a esto recibe el nombre del archivo de mensaje y analiza si se puede ocultar en el archivo transporte o resultará truncado.

Posterior a esto, el programa solicita al usuario el nombre deseado para el archivo que contendrá el resultado de insertar el mensaje en el archivo de transporte, es decir, el archivo de salida.

Finalmente el programa realiza el algoritmo ejemplificado en el diagrama de flujo correspondiente a la inserción de mensajes en archivos BMP, dejando la salida del sistema en el archivo indicado por el usuario.

Para el proceso de extracción de mensajes dentro de los archivos en BMP que esconden un mensaje se generó el código "extraeimagen.c" en lenguaje C, basando su funcionamiento en el diagrama de flujo para la extracción de mensajes en archivos BMP.

Al igual que el proceso de inserción de mensajes, es necesario realizar un proceso de compilación para obtener una aplicación ejecutable a partir del código fuente, generado sin utilizar funciones ligadas a un sistema operativo o compilador con el motivo de que pueda funcionar en cualquier equipo que cuente con un compilador de ANSI C.

El programa recibe el nombre del archivo de transporte del cual se extraerá el mensaje (recibe solamente archivos de BMP), además del nombre del archivo donde se pretende extraer el mensaje oculto, el cual será una sucesión de caracteres obtenidos en base al arreglo binario que represente el código ASCII obtenido de los bits menos significativos de cada byte en el archivo de transporte.

El proceso concluye hasta terminar de leer el archivo transporte o hasta formar, mediante el arreglo, el caracter de fin de archivo (*EOF*).

Pruebas.

Durante la primera fase de pruebas se obtuvieron resultados incorrectos para el proceso de inserción del mensaje. La imagen que se obtenía como resultado contenía pixeles que cambiaban su color de manera notoria, a simple vista se notaba la diferencia entre la imagen original y la modificada.

Este inconveniente, obviamente, hacía que el objetivo de que la información pasara inadvertida no se cumpliera y el archivo levantara sospechas de su modificación o de la presencia de algún error en el contenido del mismo.

Por esta razón se tuvo que realizar un proceso de detección y corrección del defecto, el cual se ubicó en el nivel de codificación, pues se tenía un error al momento de tratar de manipular el último bit y, contrario a lo necesitado, se modificaba el byte completo.

El error se modificó en código, utilizando la operación correcta para modificar solamente el último bit mediante un *AND binario* con ~1 (para fijar en cero el último bit) o mediante un *OR binario* con 1 (para forzar el valor uno en el bit menos significativo).

Se procedió a realizar el proceso de compilación del nuevo código y se realizaron pruebas con la versión actualizada del programa para evaluar los nuevos resultados que se obtuvieran. Como resultado a esta nueva adecuación del sistema, se obtuvo un archivo de salida que, ante el ojo humano, eran exactamente iguales; además de que conservaba las mismas propiedades que el archivo original (espacio en disco, cantidad de pixeles).

De ésta manera se cumplía perfectamente el objetivo planteado y este módulo se dio por válido.

Para el caso de la extracción de mensajes se obtuvieron resultados satisfactorios, pues la ejecución de dicho programa nos producía como salida un archivo con la copia fiel del mensaje original. Por lo que el módulo no presentó adecuaciones ni defectos inesperados que implicaran un mantenimiento en dicho código.

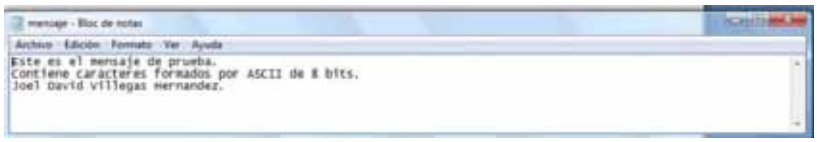
Los resultados obtenidos en esta fase de pruebas se pueden consultar en el Apéndice C.

Apéndice A. Resultados del desarrollo de esteganografía en texto.

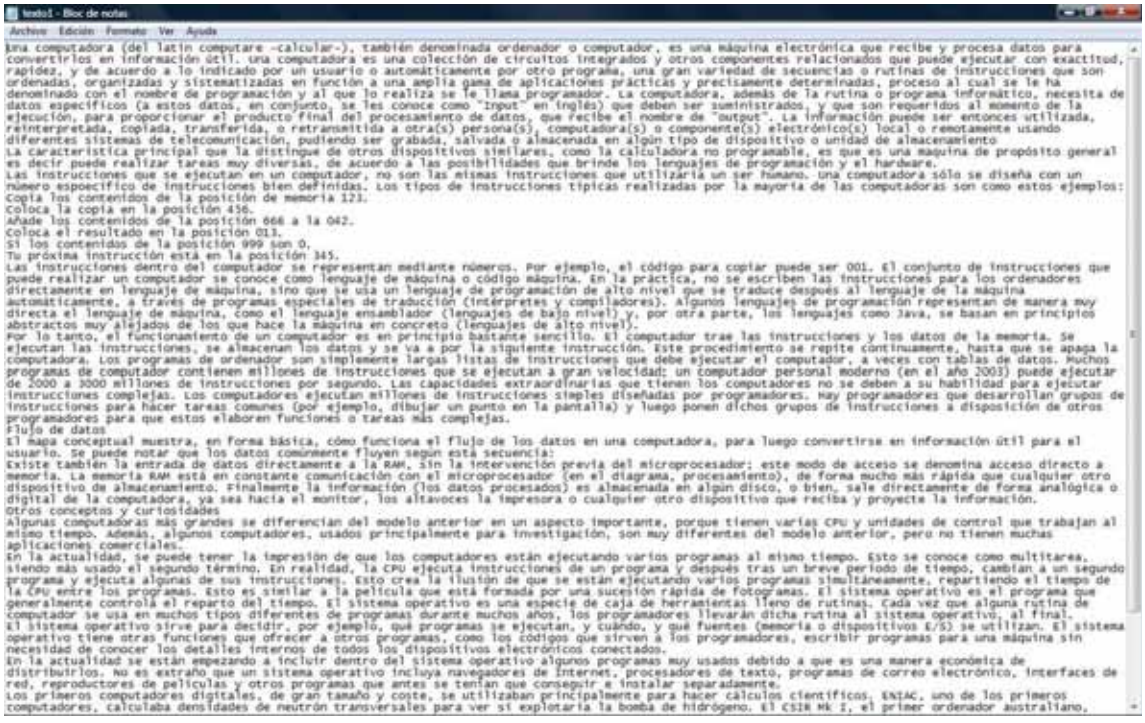
Se presentan los resultados parciales obtenidos de la primera prueba y los resultados finales del sistema generado para este proósito. Además se presenta la justificación de los resultados y la conclusión del proceso.

Dentro del módulo de inserción del mensaje en un archivo de texto plano, para las pruebas se utilizaron los siguientes archivos como entradas:

- Archivo de mensaje, texto plano. Archivo que contiene el mensaje a ocultar, formado en texto plano con caracteres representables en ASCII de 8 bits.



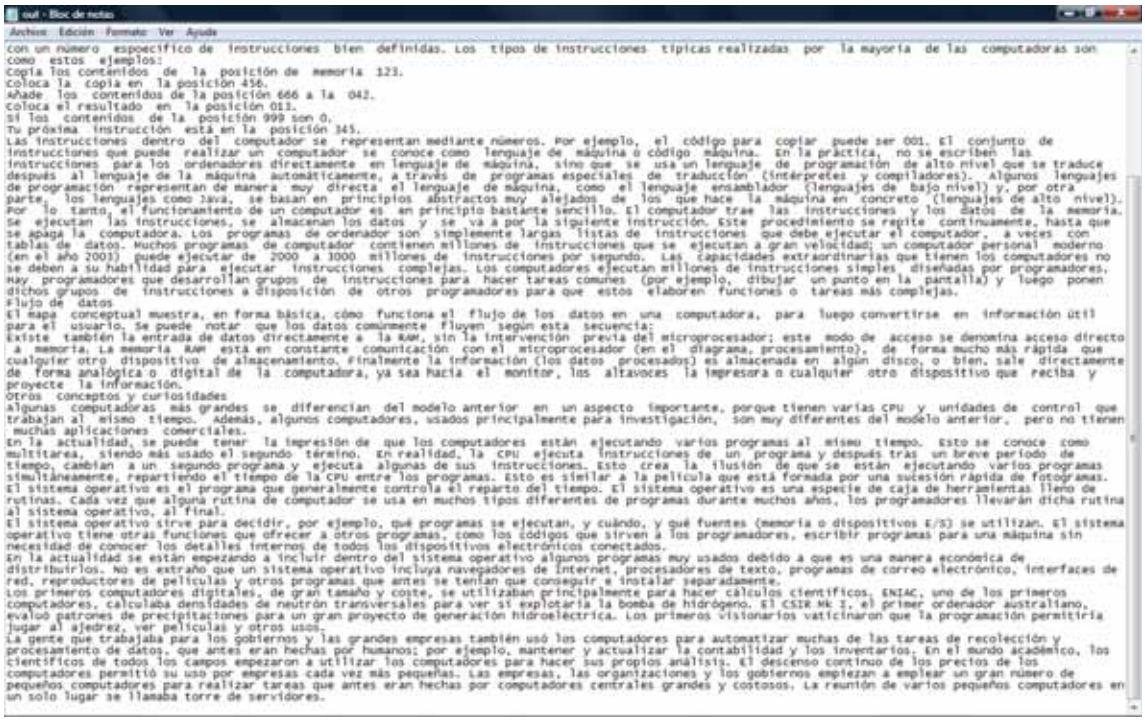
- Archivo de transporte, texto plano. Archivo en texto plano, será el transporte para que nuestro mensaje pase desapercibido; en este ejemplo contiene espacios sencillos, y debe contener al menos 8 espacios por cada caracter del archivo de mensaje.



Se realizó el proceso de inserción del mensaje y se generó un archivo de salida en texto plano, que contiene la información del mensaje de transporte, con la modificación de espacios para mantener el mensaje oculto.

Se obtuvo el siguiente archivo como salida del proceso:

- Archivo de salida, texto plano. Archivo en texto plano, resultado de insertar el mensaje en el archivo de transporte. Contiene espacios sencillos y dobles para la transmisión del mensaje.



En estos resultados parciales, si se tiene todo el archivo dentro del campo visual, podemos observar que el final del archivo contiene solamente espacios sencillos entre las palabras, haciendo notorio que el inicio del mismo contiene espacios sencillos y dobles insertados de manera irregular.

Por esta razón el objetivo de ocultar la información quedaba parcialmente cubierto debido a que, al ver un archivo con un espaciado diferente al final del archivo, podría levantar sospechas sobre la manipulación del mismo. Por medio de ésta justificación, se procedió a realizar un mantenimiento en el código para mejorar los resultados.

Se modificó el código fuente y se agregó una condición para que, después de insertado el mensaje en el archivo transporte, se continúen insertando espacios dobles y sencillos de manera aleatoria para hacer menos visible la presencia del mensaje en el archivo.

Prosiguió la compilación de la nueva versión del código, la ejecución del mismo, utilizando las mismas entradas que en la fase de pruebas anterior, para poder comparar las salidas, y la verificación del impacto en los resultados.

Se obtuvo el siguiente archivo como salida del sistema:

- Archivo de salida, texto plano. Archivo en texto plano, resultado de insertar el mensaje en el archivo de transporte utilizando el sistema después de su mantenimiento. Contiene espacios sencillos y dobles para la transmisión del mensaje.

Como resultado del mantenimiento se obtuvo la salida con un espaciado irregular en todo el contenido del archivo, haciendo más complicado visualizar una modificación intencionada en este archivo.

Para el proceso de extracción de mensaje se utilizó como entrada el archivo de salida en texto plano, y se generó la siguiente salida:

- Archivo de mensaje, texto plano. Archivo que contiene el mensaje a ocultar, formado en texto plano con caracteres representables en ASCII de 8 bits.

De ésta manera se concluyeron los procesos con los códigos texto.c y extraetexto.c, calificando los resultados como satisfactorios.

Apéndice B. Cabecera de un archivo en formato BMP

A continuación se detalla la estructura de la cabecera de un fichero **.BMP**

Bytes	**Información**
0, 1	Tipo de fichero "BM"
2, 3, 4, 5	Tamaño del archivo
6, 7	Reservado
8, 9	Reservado
10, 11, 12, 13	Inicio de los datos de la imagen
14, 15, 16, 17	Tamaño de la cabecera del bitmap
18, 19, 20, 21	Anchura (píxels)
22, 23, 24, 25	Altura (píxels)
26, 27	Número de planos
28, 29	Tamaño de cada punto
30, 31, 32, 33	Compresión (0=no comprimido)
34, 35, 36, 37	Tamaño de la imagen
38, 39, 40, 41	Resolución horizontal
42, 43, 44, 45	Resolución vertical
46, 47, 48, 49	Tamaño de la tabla de color
50, 51, 52, 53	Contador de colores importantes

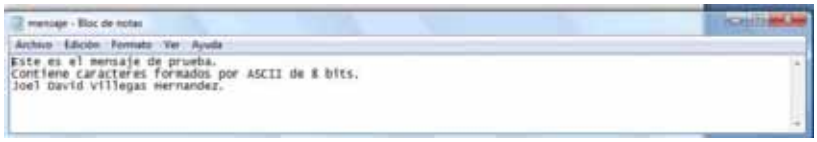
El Bitmap de una imagen **.BMP** comienza a leerse desde abajo a arriba, es decir: en una imagen en 24 bits los primeros 3 bytes corresponden al primer píxel inferior izquierdo.

Apéndice C. Resultados del desarrollo de esteganografía en imágenes.

En este anexo se ilustran los resultados obtenidos durante la fase de pruebas de este sistema.

En el módulo de inserción del mensaje se utilizaron los siguientes archivos como entradas:

- Archivo de mensaje, texto plano. Archivo que contiene el mensaje a ocultar, formado en texto plano con caracteres representables en ASCII de 8 bits.



- Archivo de transporte, formato BMP. Archivo de imagen formato BMP, utilizado como transporte para ocultar nuestro mensaje.

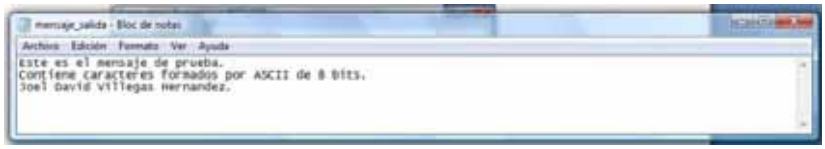
Para la visualización de resultados se ejecutó el sistema realizado para el proceso de inserción de un mensaje de texto en un archivo en formato BMP y se generó un archivo de salida en formato BMP, conteniendo la misma imagen, resultando imperceptible para la vista humana la modificación.

A continuación se presenta el archivo de salida:

- Archivo de salida, formato BMP. Archivo de imagen formato BMP, contiene el mensaje oculto de manera imperceptible para la vista humana.

Para el proceso de extracción de mensaje se utilizó este mismo archivo como entrada y se generó la salida siguiente, que representa el mensaje oculto en la imagen:

- Archivo de mensaje, texto plano. Archivo que contiene el mensaje a ocultar, formado en texto plano con caracteres representables en ASCII de 8 bits.



De ésta manera se concluyeron los procesos con los códigos imagen.c y extraeimagen.c, calificando los resultados como satisfactorios.

Apéndice D. Manual de usuario.

INTRODUCCIÓN.

El sistema de transmisión de mensajes en archivos de imágenes y texto usando esteganografía, es un conjunto de procesos independientes plasmados en archivos de código en lenguaje C, los cuales siguen el objetivo de brindar confidencialidad en la transmisión de información, de manera que, excepto por el emisor y el receptor, nadie conozca la existencia del mensaje.

REQUERIMIENTOS DEL SISTEMA.

Para poder utilizar estas aplicaciones es necesario contar con una computadora con las siguientes características mínimas:

- 128 Mbytes en memoria RAM.
- Procesador Intel x86.
- Almacenamiento en Disco duro de 10 Gbytes disponibles.
- Compilador de ANSI C, compatible con el sistema operativo utilizado (Unix, Windows).

Es necesario compilar los códigos fuente (mediante el compilador ANSI C) para poder ejecutar el código máquina que se genere.

CARACTERÍSTICAS DE LOS ARCHIVOS DE ENTRADA-SALIDA.

Los archivos de texto, necesariamente, tienen que tener un formato de texto plano representable mediante el código ASCII. Los archivos de imagen manipulados por este sistema deben de tener formato BMP (preferentemente formados por 24 bits, debido a su alta variedad de colores).

La salida del sistema será un archivo en el mismo formato que el archivo utilizado como transporte para ambos casos (texto e imágenes).

INSERCIÓN DE MENSAJES EN TEXTO

Es necesario ejecutar el archivo binario generado por el código texto.c.

A continuación el sistema pedirá que se inserte el nombre del archivo que será utilizado como transporte, el nombre del archivo que contiene el mensaje y el nombre del archivo donde se generará la salida. Es necesario entrar cada cadena de la misma manera que el sistema las va solicitando. Si el sistema no logra abrir algún archivo regresa un mensaje de error al usuario y vuelve a solicitar la cadena de nombre del archivo. Posteriormente el

sistema regresa una respuesta informando el intento fallido o exitoso del proceso de inserción del mensaje, y devuelve el control del sistema al usuario.

```
C:\Users\Orlando\Documents\pruebas-pt\EstegaTexto>texto

Nombre del archivo de transporte: texto1.txt
1268 Espacios en el archivo transporte
158 caracteres que se pueden ocultar.

Nombre del archivo de mensaje: mensaje.txt
110 Caracteres en el archivo de mensaje.

Nombre del archivo deonde se generar├í el resultado: salida.txt

Mensaje oculto exitosamente en salida.txt


C:\Users\Orlando\Documents\pruebas-pt\EstegaTexto>
```

EXTRACCIÓN DE MENSAJES EN TEXTO

Es necesario ejecutar el archivo binario generado por el código extraetexto.c.

A continuación el sistema pedirá que se inserte el nombre del archivo del cual se extraerá el mensaje (denominado archivo transporte) y el nombre del archivo donde se generará la salida. Es necesario entrar cada cadena de la misma manera que el sistema las va solicitando. Si el sistema no logra abrir algún archivo regresa un mensaje de error al usuario y vuelve a solicitar la cadena de nombre del archivo. Posteriormente el sistema regresa una respuesta informando el intento fallido o exitoso del proceso de extracción del mensaje, y devuelve el control del sistema al usuario.

```
C:\Users\Orlando\Documents\pruebas-pt\EstegaTexto>extraetexto

Nombre del archivo de transporte: salida.txt

Nombre del archivo donde se extraer├í el mensaje: mensaje_extraccion.txt

Mensaje extraido exitosamente en mensaje_extraccion.txt


C:\Users\Orlando\Documents\pruebas-pt\EstegaTexto>_
```

INSERCIÓN DE MENSAJES EN IMAGEN

Es necesario ejecutar el archivo binario generado por el código imagen.c.

A continuación el sistema pedirá que se inserte el nombre del archivo que será utilizado como transporte, el nombre del archivo que contiene el mensaje y el nombre del archivo donde se generará la salida. Es necesario entrar cada cadena de la misma manera que el sistema las va solicitando. Si el sistema no logra abrir algún archivo regresa un mensaje de error al usuario y vuelve a solicitar la cadena de nombre del archivo. Posteriormente el sistema regresa una respuesta informando el intento fallido o exitoso del proceso de inserción del mensaje, y devuelve el control del sistema al usuario.

```
C:\Users\Orlando\Documents\pruebas-pt\EstegaImagen>imagen

Nombre del archivo de transporte: image_entrada.bmp
Error al tratar de abrir el archivo de transporte.
Nombre del archivo de transporte: Error al tratar de abrir el archivo de transporte.
^C
C:\Users\Orlando\Documents\pruebas-pt\EstegaImagen>imagen

Nombre del archivo de transporte: imagen_entrada.bmp
1440000 Espacios en el archivo transporte
180000 caracteres que se pueden ocultar

Nombre del archivo de mensaje: mensaje.txt
110 Caracteres en el archivo de mensaje.

Nombre del archivo deonde se generar├í el resultado: imagen_salida.bmp

Mensaje oculto exitosamente en imagen_salida.bmp


C:\Users\Orlando\Documents\pruebas-pt\EstegaImagen>
```

EXTRACCIÓN DE MENSAJES EN TEXTO

Es necesario ejecutar el archivo binario generado por el código extraeimagen.c.

A continuación el sistema pedirá que se inserte el nombre del archivo del cual se extraerá el mensaje (denominado archivo transporte) y el nombre del archivo donde se generará la salida. Es necesario entrar cada cadena de la misma manera que el sistema las va solicitando. Si el sistema no logra abrir algún archivo regresa un mensaje de error al usuario y vuelve a solicitar la cadena de nombre del archivo. Posteriormente el sistema regresa una respuesta informando el intento fallido o exitoso del proceso de extracción del mensaje, y devuelve el control del sistema al usuario.

```
C:\Users\Orlando\Documents\pruebas-pt\EstegaImagen>extraeimagen

Nombre del archivo de transporte: imagen_salida.bmp

Nombre del archivo donde se extraer├í el mensaje: mensaje_salida.txt

Mensaje extraido exitosamente en mensaje_salida.txt


C:\Users\Orlando\Documents\pruebas-pt\EstegaImagen>_
```

Bibliografía

- Seguridad en Internet. http://www.eumed.net/cursecon/ecoinet/seguridad/index.htm
- Esteganografía. Dürsteler, Juan. http://www.infovis.net/printMag.php?num=101&lang=1
- Archivo de texto - Wikipedia, la enciclopedia libre http://es.wikipedia.org/wiki/Archivo_de_texto
- Formato BMP http://help.adobe.com/es_ES/Photoshop/10.0/help.html?content=WSfd1234e1c4b69f30ea53e41001031ab64-7751.html
- Windows bitmap - Wikipedia, la enciclopedia libre http://es.wikipedia.org/wiki/BMP