ESTUDIO Y ANÁLISIS DE LOS MÉTODOS DE ESTENOGRAFÍA PARA APLICACIONES TELEMÁTICAS

Ing. JAVIER ENRIQUE QUINTERO ROJAS

UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA
FACULTAD DE INGENIERÍA DE SISTEMAS
MAESTRÍA EN TELEMÁTICA
BUCARAMANGA
2015

ESTUDIO Y ANÁLISIS DE LOS MÉTODOS DE ESTENOGRAFÍA PARA APLICACIONES TELEMÁTICAS

Ing. JAVIER ENRIQUE QUINTERO ROJAS

Trabajo de Investigación para optar el título de maestría en telemática

Director:
Mag. JAURI LEÓN TÉLLEZ
Grupo de Investigación en Ciencias Aplicadas – GINCAP
Universidad Autónoma de Bucaramanga

UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA
FACULTAD DE INGENIERÍA DE SISTEMAS
MAESTRÍA EN TELEMÁTICA
BUCARAMANGA
2015

DEDICATORIA

Dedico esta tesis a mis a mis padres Anny Graciela y Javier Antonio por el amor que me brindan y por su apoyo incondicional.

A mi esposa Laura y mis Hijos Christian, Anny, Gaby y María Paula por el tiempo que les robé mientras escribía esta tesis.

AGRADECIMIENTOS

A las Unidades Tecnológicas de Santander por el apoyo económico y vinculación laboral que hizo posible este logro.

A la Universidad Autónoma de Bucaramanga por encontrar en ella los espacios de colaboración y asesoría correspondiente a la investigación y gestión académica.

A mi director de proyecto Jauri León que nunca desistió en apoyarme en la culminación de ésta tesis.

Al ingeniero Francisco Javier Dietes por su colaboración y orientación durante el desarrollo de la maestría.

A todos los que me acompañaron en este proceso por su apoyo incondicional

CONTENIDO

Pág.

LISTA DE FIGURAS

Pág.

RESUMEN

A nivel mundial las empresas y organizaciones de distintos sectores gubernamentales, sociales, académicos y económicos han incrementado su interés y atención en la protección de lo que hoy en día es un activo de vital importancia la “INFORMACIÓN”, todo esto consecuencia de que los avances tecnológicos han derivado en nuevas amenazas, y los problemas que afectan a la seguridad informática avanzan a la par que la tecnología. Si no se disponen de estrategias, políticas y soluciones de hardware y software sofisticadas cualquier organización Grande, Mediana o Pequeña puede ser fácilmente víctima de un hacker o pirata informático que haya decidido perjudicarlas.

La infraestructura de redes de una organización está amenazada constantemente en que sea filtrada y atacada en sus sistemas de almacenamiento de información para arrebatar datos e información privilegiada, ya que ésta puede tener recovecos o espacios donde infiltrarse.

¿Qué hacer ante un escenario de constante amenaza? Dos acciones: Protegerse constantemente y Actualizarse "No bajar la Guardia" ante las diferentes vulnerabilidades que pueden surgir a diario.

En este trabajo se describe una forma de transmitir información de forma segura que usa un método esteganográfico de ocultamiento y se proporciona una herramienta que facilita la implementación de dicho método.

El desarrollo del proyecto se encuentra divido en tres momentos, un primer momento de contextualización donde se realizó la respectiva revisión bibliográfica y revisión de antecedentes investigativos para dar soporte teórico a esta tesis. Un segundo momento que incluye además de la fase de desarrollo de la herramienta en su primera versión, una segunda versión que es la incluida en este trabajo. En esta segunda versión se corrigieron algunos errores que fueron detectados durante la primera etapa de desarrollo. Por último se encuentran los resultados y análisis de datos de los cuales se obtuvieron las conclusiones y recomendaciones finales.

Palabras Claves: Esteganografía, Telemática, Información, Señal de Audio, Criptografía, Matlab

ABSTRACT

Companies and organizations from government, social , academic and economic sectors have increased their interest and attention on protecting what today is a vital asset "information" , all of that as a result of what technologic advances have led to new threats; and challenges affecting the Informatics security go hand to hand with technology . If not disposed of strategies, policies and hardware solutions and sophisticated software any organization large, medium or small can easily be the victim of a hacker that has decided to harm them.

The network infrastructure of an organization is constantly threatened as it is filtered and attacked in their storage systems to snatch data and privileged information , as this may have recesses or spaces where can be infiltrated.

What to do in a scenario of constant threat? Two actions: Protect and updated constantly “remain vigilant” to the different vulnerabilities that may arise daily.

In this paper is described a way to securely transmit information using a stenographic method of concealment and provides a tool that facilitates its implementation.

The project is divided into three stages, the first stage is contextualization where the respective bibliographical review and review of background research was conducted to give theoretical support to this thesis. A second stage that in addition to the development phase of the tool in its first version, include a second version in this work. In this second version some errors that were detected during the first stage of development were corrected. Finally the results and analysis of data from which the final conclusions and recommendations were obtained are shown.

Keywords: Steganography, Telematics, hide information, Audio Signal, Cryptography, Matlab.

INTRODUCCIÓN

Desde ya hace varios años, es muy frecuente el envió de información por medios magnéticos, los avances tecnológicos nos han facilitado tanto en tiempo como distancia la forma en que nos comunicamos; es posible realizar el envío de información a cualquier parte del mundo en cuestión de segundos. Teniendo en cuenta que esta información puede ser de carácter público como privado, existe el riesgo que pueda ser manipulada por personas no deseadas. Para ello, se han generado distintas técnicas de protección, y evitar cualquier tipo de alteración de la información que se transmite.

Una de estas técnicas es *la esteganografía*, una herramienta que tiende a aplicarse en situaciones y casos específicos, es decir, si se desea ocultar algún mensaje basta con que se aplique alguna de las técnicas para adelantar el ocultamiento, generando un objeto con el mensaje encubierto.

En la actualidad, la revolución digital sigue tomando posicionamiento, ayudada de excelentes técnicas de transmisión de alta velocidad, redes, tecnologías satelitales, etc., los cuales constituyen una verdadera infraestructura del ciberespacio. Encontramos en la Internet la máxima expresión de esta revolución, la información publicada en la Web ha ocasionado grandes problemas por su uso indiscriminado y el intercambio de información ha generado situaciones difíciles de controlar.

La presencia de fraudes en las redes públicas, la falsa representación y la reproducción ilegal van en aumento haciendo que sus propietarios requieran de sistemas eficientes de detección de datos ocultos en canales encubiertos,

imágenes y archivos de sonido, cuyos formatos permiten camuflar todo tipo de información (Spencer, 2004).

Existes varias herramientas para el ocultamiento de información, los grupos terroristas por ejemplo utilizan la e*stenografía* para enviar mensajes clandestinos, cometer delitos informáticos que continuamente son ocasionados a través de la red, o simplemente aficionados que juegan a filtrar mensajes secretos que realmente llegan a ser menos ofensivos que los mismos archivos en los que estos son camuflados. Es de vital importancia para todas las organizaciones gubernamentales y privadas, entidades sociales y académicas mantener la confidencialidad en sus comunicaciones y protegerse de cualquier intruso, así como detectar cualquier tipo de amenaza o información oculta que pueda poner en riesgo estructuras físicas, económicas, políticas o sociales, lo que puede lograrse con la técnica del estegoanálisis.

En este proyecto, se pretende detallar los elementos de *la esteganografía*, sus principales estrategias, algoritmos y limitaciones, con el fin de orientar a la comunidad educativa en una nueva estrategia para el ocultamiento de la información.

1. GENERALIDADES DEL PROYECTO

1.1 FORMULACIÓN DEL PROBLEMA

El tema de esta investigación está enmarcado en el análisis de las principales técnicas esteganográficas aplicadas en archivos de audio, estableciendo para ello un método computacional como herramienta para la evaluación del desempeño de los algoritmos empleados.

Parte del entendimiento de estas técnicas requiere un análisis cualitativo y cuantitativo de cómo se procesan, modifican y codifican las señales de audio, identificar cuáles son los elementos que se pueden excluir (para ser luego reemplazados por "información oculta") y cuáles no; también se puede incluir el detalle del sistema auditivo humano, principal elemento diferenciador de rangos de frecuencia perceptibles.

Actualmente el perfeccionamiento de las técnicas de procesamiento digital de video (audio + imágenes) representa un gran interés, pues el almacenamiento y las formas de compartir archivos en medios magnéticos o en la Web, son catalogados como las actividades de mayor demanda. Por esta razón se requiere modelar las técnicas que permiten comprimir y aumentar la calidad de archivos fuente, tanto como para los de tipo audio como los de imagen. Este análisis debe incluir mejoras que pueden efectuar en los niveles de seguridad tanto en la transmisión como en la recepción sobre los canales de comunicación empleados.

Los métodos estaganográficos empleados para ocultar información poseen características como: invisibilidad estadística o algorítmica, seguridad, capacidad, formas de detección, complejidad y coste computacional. Entre las técnicas

desarrolladas se destacan: modificación de la paleta de coles de una imagen, métodos de sustitución (imagen y audio), ocultación en el dominio transformado (imagen y audio), spread spectrum (imagen y audio), esteganografía estadística (imagen y audio), esteganografía sobre texto, entre otros.

El objeto de esta investigación permitirá crear una interfaz gráfica de usuario (GUI) bajo algoritmo para cualificar una técnica esteganográfica en particular: LSB (Bit menos significativo), empleada sobre archivos de audio con extensión tipo WAV.

Lo anterior deriva en la pregunta de investigación que da lugar a la presente propuesta: Cumpliendo las condiciones básicas de un estegosistema, la técnica esteganográfica LSB aplicada a una archivo de audio WAV representa la mejor alternativa para la transmisión en un entorno hostil?

1.2 JUSTIFICACIÓN

En la actualidad el flujo constante de información multimedia en equipos personales y redes corporativas ha suscitado el interés de las empresas dedicadas a la seguridad informática y activos de networking. Surgen nuevas técnicas (aunque no es del todo nueva esta ciencia) como la esteganografía que aprovechan las tecnologías de la información para “camuflar” información relevante en textos, archivos de audio, fotografías, videos, algunas veces de apariencia inofensiva. La esteganografía ha sido tradicionalmente usada por agencia militares y de inteligencia, policía y criminales, así como por civiles que buscan evadir restricciones.

La esteganografía clásica se basa en el desconocimiento del canal encubierto bajo uso, en la moderna se emplean canales digitales para alcanzar el objetivo. Es común que el objeto contenedor sea conocido, pero lo que se ignora es el

algoritmo de inserción de datos en dicho objeto. 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 (conocido como algoritmo de ocultación), lo que no ocurre en caso del criptograma. La criptografía, propende por asegurar la confidencialidad de la información ante la presencia de un interceptor que es capaz de ver el criptograma, aun cuando éste conoce el algoritmo que lo genera.

Las áreas de ciberdefensa y ciberseguridad surgen como contramedidas para el uso equivoco de la esteganografía, pues se han detectado actividades terroristas en muchos mensajes y videos circulantes por la Web, algunos de tipo subliminal imperceptibles al oído u ojo humano.

Identificar, analizar y diferenciar las técnicas esteganográficas y las características de los estegosistemas, representa una base fundamental para crear modelos de aseguramiento de la información y canales de comunicación, permitiendo a su vez, la combinación con técnicas existentes como la criptografía.

1.3 OBJETIVOS

1.3.1 Objetivo General Realizar un estudio y análisis de métodos de Esteganografía en el dominio del espacio y de la frecuencia para aplicaciones telemáticas.

1.3.2 Objetivos específicos

- Recopilar y analizar la información bibliográfica sobre esta técnica de ocultamiento y sus características principales como base de estudio y generar un esquema de comparación.

- Describir algunas técnicas esteganográficas en el dominio espacial y de frecuencia, en donde se especifiquen parámetros y características técnicas utilizadas para la transmisión de la información.

- Analizar las aplicaciones en telemática de las técnicas mencionadas anteriormente enfatizando en los protocolos de comunicación y realizar una simulación en Matlab para demostrar los resultados obtenidos en la aplicación de la técnica LSB.

- Establecer las conclusiones y recomendaciones de los resultados obtenidos en la realización del estudio.

2. MARCO REFERENCIAL

2.1 ANTECEDENTES

En la actualidad el uso de *la esteganografía* y la criptografía ha fortalecido el desarrollo de sistemas de información seguros. De la mano con el desarrollo tecnológico y creación de dispositivos computacional de gran poder, se crean nuevas técnicas y algoritmos para el "ocultamiento" de información, haciendo uso incluso, del cluoud computing aplicado al esteganoanálisis. Los modelos matemáticos y estadísticos son el pilar fundamental para el entendimiento de los procesos que se llevan a cabo en la manipulación de información en los estegosistemas.

A continuación se resaltan investigaciones realizadas en esta área, que permiten establecer las condiciones iniciales del proyecto e identificar fortalezas y debilidades entre los modelos empleados.

Esteganografía en audio por el criterio de la transformada Discreta del coseno (TDC) (Morales, Delgado Guitierrez , & Vazquez Medina)***.*** En esta investigación se presenta una metodología para resolver parte del problema de los prisioneros propuesto por Simmons. La alternativa planteada permite establecer una comunicación "encubierta" segura por medio de la TDC. Los valores obtenidos para el coeficiente de correlación y la divergencia de Kullbacko-Leibler muestran que este estaganograma propuesto es muy seguro pues es muy parecido a la portadora. Se destaca de este trabajo, el uso como portadoras archivos no conocidos o muy utilizados, si alguien requiere información sin permiso, no puede obtenerla, pues ni siquiera sabrá que la información está oculta.

Sistema de Watermarking basado en alteraciones de color. (Galcerán, 2013) En este proyecto se emplea un método en el dominio espacial que permite insertar información binaria en las imágenes manipulando su componente de amarillo. El proceso permite dividir en bloques y los bloques y los bits se codificaron en la imagen utilizando la diferencia de amarillo entre dos bloques contiguos, por tanto cada dos bloques se podrá codificar un bit, si el primer bloque contiene más amarillo que el segundo se codificará un '0' y si es al revés se codificará un "1". Esta técnica se implementó en Matlab y permitió determinar soluciones adaptadas donde la marca por la inserción de datos imagen, era invisible y aunque podría ser camuflada, degradaba la imagen original. La evaluación de la técnica determinó acorde a la visibilidad y resistencia de la marca no resulto se excesivamente robusto.

Técnica de inserción de información en video aprovechando el mismo ancho de banda (Gamez, 2009) Esta investigación analiza el uso de la Transformada Wavelet discreta (TWD), en el desarrollo de un estegosistema para video, capaz de albergar cualquier clase de archivos aprovechando el mismo ancho en un medio guiado para la transmisión de CATV. Se llevaron a cabo pruebas de aplicación de ruido gaussiano, analizando el efecto en la varianza para cada histograma del video base. Se determinó que al conservar la frecuencia de colores en cada una de las matrices de una imagen RGB, no se altera su ancho de banda, siendo ya insertada la información deseada.

En el caso del ancho de banda establecido de 6Mhz, se estableció que el algoritmo esteganográfico se puede aplicar a la transmisión de video, permitiendo agregar más de dos archivos a un solo archivo de video, siendo esto un elemento muy importante en las telecomunicaciones puesto que para muchas aplicaciones de hoy en día se puede ahorrar tiempo y dinero en las transmisiones de cualquier índole. Finalmente el autor propone, mejoras como la integración con un

acelerador de video y realizar un algoritmo esteganográfico sin la separación del flujo de audio.

Esteganografía y estegoanálisis: ocultación de datos en streams de audio vorbis (Vico, 2010)***.*** Esta investigación permite un acercamiento al estudio de los codecs perceptivos que utilizan modelos psicoacústicos para eliminar componentes prescindibles de las señales de audio originales, entre ellos. Mp3, (MPEG-1 Audio layer 3), AAC (Advanced Audio Coding), WMA (Windows Media Audio) y en especial el codec Vorbis conocido como Ogg Vorbis, o simplemente Ogg. Se desarrolla entonces un modelo de modificación de los vectores residuales en un stream de audio, determinando los cambios máximos que podrán introducirse en los vectores residuales. El modelo del códec Vorbis tiene un elemento central, el vector floor, que se puede entender como la superposición de las máscaras tonales y el ruido. Este vector floor se utiliza a modo de cuantizador, restándolo de los coeficientes frecuenciales en escala logarítmica. Se emplea luego el método de paridad de bit para insertar los masajes "subliminales" en el archivo de audio. Esta técnica permite, independientemente de cómo lleguen los datos subliminales (cifrados o no), borrar cualquier rastro estadístico de la distribución original, obteniendo una distribución uniforme, que siempre es más fácil de manejar a la hora de mantener las propiedades estadísticas del objeto portador. Se ha determinado entre las conclusiones que este es un método delicado, ya que las propiedades del sistema auditivo humano hacen que el volumen se perciba de forma subjetiva.

Técnicas de auto escalado de cloud computing aplicadas al esteganoanálisis. (Obregozo, 2011) Este trabajo presenta el desarrollo de un algoritmo esteganográfico evaluado con técnicas que optimizan los recursos de cómputo como: CPU, RAM, etc. El mensaje anfitrión que se escogió es un foto con extensión bmp y el mensaje huésped seleccionado es una longitud variable de caracteres que hacen parte del primer capítulo del libro "El Quijote" La técnica

empleada es la Transformada Discreta de Fourier (DFT) agregada a un sistema de sustitución de los coeficientes obtenidos en el dominio de la frecuencia. Esta técnica permite acoger la imagen del mensaje anfitrión, dividirlo por bloques y aplicar la DFT para obtener las componentes frecuenciales de cada uno de los bloques. Luego se codifican usando el ultimo bit que no se ve alterado por el canal (se guarda la imagen en números enteros de 0-255), parte del mensaje huésped codificado en ASCII y se deshace la DFT para obtener otra vez el bloque. Se repite este proceso hasta obtener todo el mensaje oculto en la imagen. Se encontró que la computación en la nube (cloud computing) usando la tipo maestro –esclavo de varios niveles, es la mejor solución.

El proceso esteganográfico descrito se implementó usando Matlab 2011a y debido al gran coste computacional la herramienta Octave 3.0 fue la solución para ejecutar este programa en Cloud, usando sistemas operativos libres como Linux. Como resultados encontrados en el estegoanálisis se determina que la propiedad de auto-escalado en el proceso permite reducir considerablemente el tiempo en el que una imagen puede ser analizada.

2.2 MARCO TEÓRICO

2.2.1 Qué es la esteganografía Del griego στεγανοζ (steganos, encubierto) y γραπτοζ (graphos, escritura) nace el término esteganografía, el arte de escribir de forma oculta.

Aunque κρυπτός (criptos, oculto) y στεγανοζ (steganos, encubierto) puedan parecer en un principio términos equivalentes, o al menos similares, son cosas completamente distintas. La criptografía es el arte de escribir de forma enigmática (según la Real Academia Española), mientras que la esteganografía es el arte de escribir de forma oculta. Puede que sigan pareciendo similares, pero las

connotaciones toman mucho valor al analizarlo detenidamente: la criptografía tiene su fuerza en la imposibilidad de comprender el mensaje, mientras que la esteganografía la tiene en el desconocimiento de que el mensaje siquiera existe.

Aplicado al campo informático, se pueden dar los siguientes ejemplos: se podría robar un mensaje cifrado con relativa facilidad, pero aun sabiendo que contiene información importante seríamos incapaces de obtener información alguna de él (si la criptografía ha cumplido con su cometido). Respecto a la esteganografía, se podría capturar el tráfico completo de un individuo y tratar de analizarlo completamente sin tener la certeza de que haya o no un mensaje oculto. (Death-Master, 2004)

2.2.2 Historia esteganografía Heródoto, en sus Historias, ya describe cómo las tabletas recubiertas de cera que se utilizaban para escribir se habían usado para advertir del peligro de invasión de Esparta por parte de Jerjes. La estratagema consistía en retirar la cera, escribir el mensaje en el soporte de madera y recubrir de nuevo con la cera la tableta. De este modo, el mensaje escrito en el soporte de madera pasaba totalmente desapercibido.

Figura 1. Primeros usos de la esteganografía (Cano, 2004)

En la Grecia antigua, usaban un método mediante el cual, una persona elegida al azar y con la cabeza afeitada, le tatuaban el mensaje secreto en la cabeza, tras lo cual le dejaban crecer el pelo a su longitud normal. El mensajero procedería a llevar a su destino el mensaje ya que pasaría cualquier control de seguridad al no

percibir nada sospechoso por parte del enemigo. Una vez presentado al receptor de la información, este le afeitaba la cabeza para leer el texto secreto. Una desventaja importante a este método era el estado latente en conseguir el mensaje al receptor. Se tuvo que esperar a que el pelo creciera suficientemente para cubrir el texto antes de que el mensaje pudiera ser entregado. Otra desventaja a este método es que el mensajero quedaba marcado con el tatuaje de por vida sin poder ser destruido.

Durante la guerra mundial II, las tintas invisibles fueron utilizadas para encubrir la información en notas o letras aparentemente estándares e inofensivas. Las fuentes comunes para las tintas invisibles son leche, vinagre, zumos de fruta.

Uno de los métodos más ingeniosos está desarrollado por Gaspar Schott y se detalla en su libro Schola Steganographica. El método implicaba el codificar la información emparejando letras a las notas musicales especıficas sobre una hoja. Aparentemente pareciera como una partitura musical normal. Si uno tocara el trozo de dicha partitura musical en un instrumento, el resultado no sería todo lo agradable que esperaríamos. (Clemente, 2015)

En otros tiempos, la información que rodeaba a una persona era transportada por unos canales muy definidos (correo, teléfono...) y de forma limitada; pero eso es algo que en las últimas décadas ha cambiado de forma radical. El actual desarrollo de la tecnología computacional y las telecomunicaciones, cuyas culminaciones son el ordenador personal e Internet respectivamente, han rodeado completamente nuestras vidas de torrentes de información. Vivimos, de hecho, rodeados de un continuo “ruido de fondo”. Si se piensa en la información que se extrae al visitar una web y la cantidad real de información que contiene, se puede ver que el vivir en un medio ruidoso nos hace, en cierto modo, impermeables a la información no deseada, actuando como “cribas humanas” que separan el grano de la paja en la Gran Telaraña Mundial. Ese enorme ruido de fondo, ese enorme

remanso de información que supone Internet, es el perfecto cultivo para las técnicas estenográficas. Cuanta más información exista y cuanto más fácilmente sea ésta ignorada, más fácil será que la información pase completamente desapercibida al resto del mundo. En particular, las técnicas conocidas como "marcas de agua" (Watermarking) para ocultar mensajes de copyright y la inclusión de "huellas dactilares" (fingerprinting) para identificar números de serie o distinguir objetos concretos entre otros similares, han tenido un notable auge. Su importancia radica en que la protección de los derechos de copyright de imágenes, bandas sonoras y documentos escritos se ha hecho cada vez más difícil en un mundo en el que bajarse de la web una imagen o una banda sonora original está a tan sólo un clic de distancia. Es importante recordar los problemas de la industria discográfica con Napster o el software pirateado. Los sistemas de Watermarking modifican sutilmente los bits que constituyen el documento, la banda sonora o la imagen de forma que el resultado es sensiblemente igual al original, pero mediante un algoritmo apropiado se puede reconstruir la "firma digital" embebida en el mismo. (Cano, 2004)

2.2.3 Terminología

Información embebida.

Se conoce por información embebida o información oculta a la información que se envía en forma secreta (oculta).

Carrier.

La pista de audio, video, texto, imagen o en resumen los datos entre los cuales se requiere ocultar la información embebida, recibe el nombre de portador (en la literatura inglesa se encuentran los términos de "carrier" y "cover").

Estegano-objeto.

El resultado de introducir la información a embeber en el portador es el estegano-objeto. En casos concretos de audio, texto, imagen, etc. También se llama estegano-audio, estegano-imagen, estegano-texto respectivamente.

Estegano-clave.

La clave utilizada en el proceso se conoce como estegano-clave. En la actualidad, existen tres tendencias de diseño de sistema esteganográfico. (Vico, 2010)

Estegano-sistemas de clave simétrica

Es el esquema de estegano-sistema más común. Emisor y receptor comparten una clave secreta (estegano-clave) y toda la seguridad del sistema se basa en ella.

Estegano-sistemas de clave pública.

Son aquellos sistemas que requieren el uso de dos claves. Una clave pública para el proceso de ocultación y una clave privada para obtener el mensaje oculto.

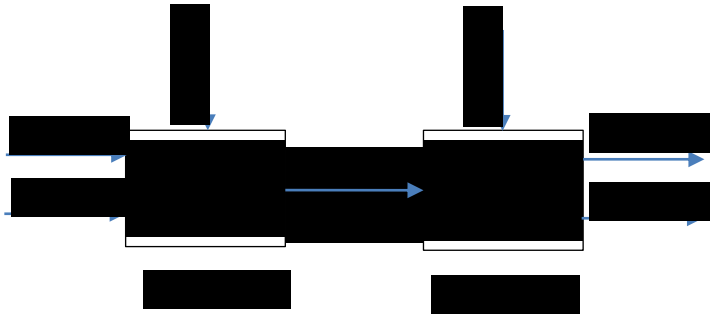
Estegano-sistemas cuánticos.

Estos sistemas aprovechan los conocimientos sobre física cuántica para diseñar sistemas que faciliten la ocultación de información. Existen varias formas de realizar esto, por ejemplo, aprovechándose del ruido cuántico o de los códigos correctores cuánticos. Ideas parecidas se han utilizado en las últimas décadas para establecer comunicaciones enmascaradas utilizando diversas características

de la naturaleza (comunicaciones atmosféricas con ciertas peculiaridades, ruido ambiente, etc.). (Muñoz, 2014)

2.2.4 Cómo funciona la esteganografía Para esconder un mensaje mediante esteganografía, en primer lugar se escoge un fichero cualquiera, un documento Word, un documento PDF de Adobe, un fichero de imagen BMP o uno de sonido .WAV o .MP3, y se escoge el mensaje que se quiere ocultar, un mensaje de texto u otro fichero. El programa modifica el portador de varias formas posibles: alterando los valores de algunos de los puntos de la imagen, sumándoles o restándoles "1" de forma que sea imperceptible al usuario, pero con alguien que sepa que en esa imagen hay un mensaje el cual pueda recuperar. La siguiente figura muestra cómo funciona, a grandes rasgos, la esteganografía: (Clemente, 2015)

Figura 2. Funcionamiento de un algoritmo de esteganografía (Clemente, 2015)



f_E = Función para embeber

f_E^{-1} = Función para extraer

Cover = Objeto donde embeber el mensaje (foto, audio o video)

Emb = Mensaje embebido

Emb* = Mensaje recuperado

Llave = Parámetro de f_E

Estegano objeto = Objeto con el mensaje embebido

Emisor = Realiza el proceso de inserción de archivos

Receptor = Realiza el proceso de recuperación de archivos.

2.2.5 Métodos de audio esteganográficos Hay muchas técnicas para ocultar información o mensajes de forma que las alteraciones hechas al archivo de audio sean imperceptibles. Las aproximaciones comunes incluyen:

2.2.6 Codificación LSB Un método muy popular es el algoritmo LSB (Least Significant Bit), que reemplaza al bit menos significativo en algunos bytes del archivo de portadora para ocultar una secuencia de bytes que contienen el conjunto de datos ocultos. Es una técnica común y efectiva en casos donde la substitución LSB no cause una degradación significativa en la calidad, como en un mapa de bits de 24-bits.

En informática, el bit menos significativo (LSB) es la posición de bit en un entero binario que da el valor de las unidades, eso es, determinar si el número es par o impar. El LSB a menudo es referido como el bit que está más a la derecha. (Jayaram, 2011)

Figura 3. Representación binaria del decimal 149 (Jayaram, 2011)

El MSB en un número binario de 8 dígitos representa el valor decimal 128. El LSB representa un valor de 1. Por ejemplo, para ocultar la letra “a” (Código ASCII 97, la cual es 01100001) dentro de una portadora de 8 bytes, se puede por el LSB de cada byte así:

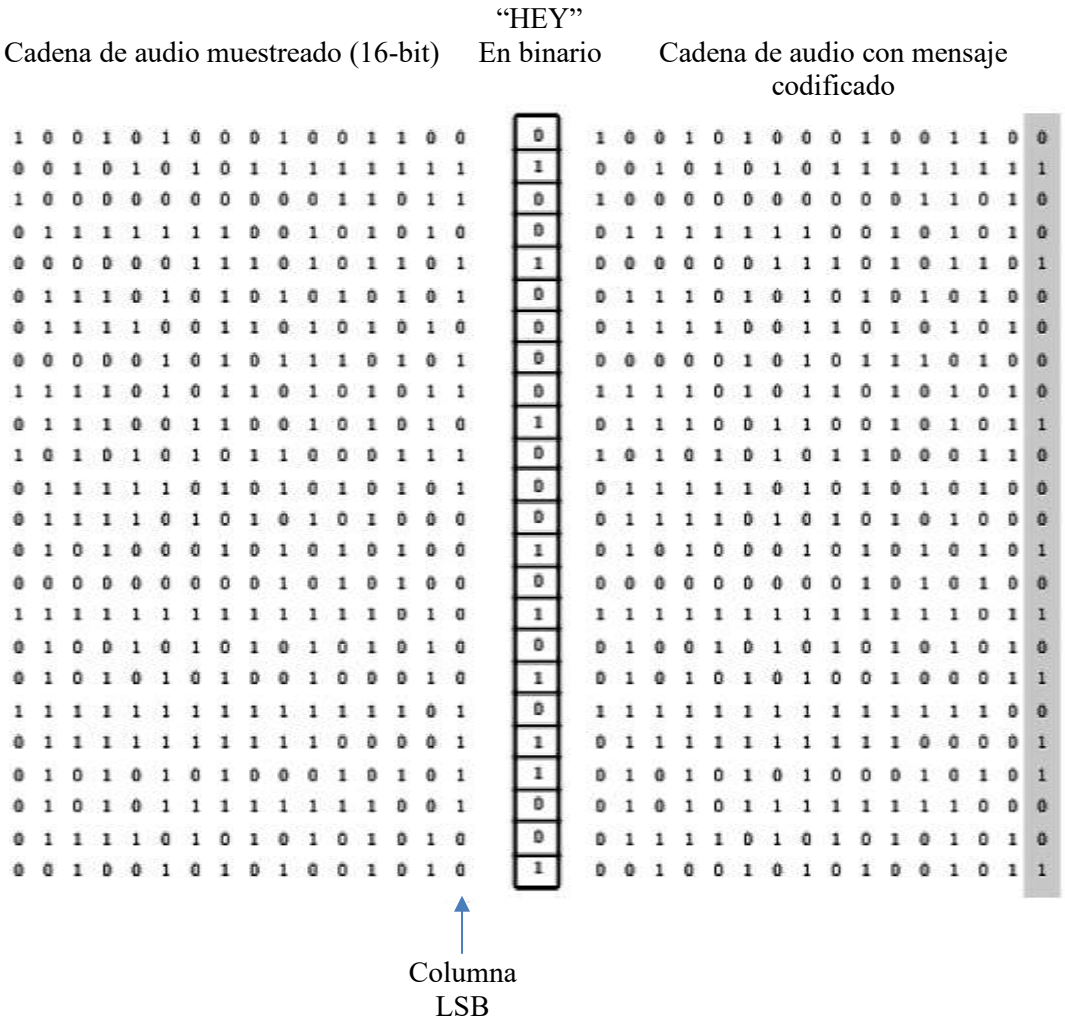
1001001**0**
0101001**1**
1001101**1**

1101001**0**
1000101**0**
0000001**0**
0111001**0**
0010101**1**

Decodificando la portadora la aplicación lee los ocho bits menos significativos de aquellos bytes para recrear el byte oculto que es 0110001 la letra “a”. Como se puede observar, el uso de la técnica permite ocultar un byte cada 8 bytes de la portadora. Nótese que hay un cincuenta por ciento de posibilidades de que el bit que se está reemplazando sea el mismo que su reemplazo, en otras palabras, la mitad del tiempo, el bit no cambia, lo que ayuda a minimizar la degradación de calidad.

La siguiente tabla muestra como el mensaje “HEY” es codificado en una muestra de 16 bit usando el método LSB. Aquí la información secreta es “HEY” y el archivo de portadora es un archivo de audio. HEY es incrustado dentro del archivo de audio. Primero la información secreta “HEY” y el archivo de audio son convertidos en una cadena de bits. La columna de los menos significativos es reemplazada por la cadena de la información secreta “HEY”. El archivo resultante después del embebido de la información secreta “HEY” es un archivo estegano. (Jayaram, 2011)

Figura 4. Ejemplo de codificación LSB (Jayaram, 2011)



2.2.7 Codificación por paridad Este método consiste en dividir el objeto portador en bloques o segmentos de un determinado tamaño estableciendo un orden de los mismos, se hará el cálculo de cada bloque en dicho orden y si la paridad del bloque b1 coincide con el i-ésimo del mensaje a ocultar no se hará nada; si no coincide se modificará el bit menos significativo de algún elemento del bloque. El receptor solamente tendrá que calcular la paridad de los bloques usando la misma ordenación del emisor. (Vico, 2010)

2.2.8 Codificación por fase La técnica de codificación por fase funciona reemplazando la fase de un segmento de audio inicial con una fase de referencia que representa la información secreta. Las fases de los segmentos restantes se ajustan para preservar la fase relativa entre segmentos. En términos de relación señal a ruido, la codificación por fase es uno de los métodos más efectivos. Cuando hay un cambio drástico de la relación entre cada uno de los componentes de frecuencia, ocurrirá una notable dispersión de fase. Sin embargo, mientras la modificación sea suficientemente pequeña, se puede lograr una codificación inaudible. Este método se basa en el hecho de que los componentes de fase del sonido no son tan perceptibles al oído humano como lo es el ruido. (Vico, 2010).

La codificación por fase se explica en el siguiente procedimiento.

a. Se divide la señal del sonido original en segmentos más pequeños de tal forma que las longitudes sean del mismo tamaño del mensaje que va a ser codificado.
b. Se crea una matriz de fase al aplicar la transformada discreta de Fourier (DFT).
c. Se calculan las diferencias de fase entre los segmentos adyacentes.
d. Los cambios de fase entre los segmentos adyacentes son fácilmente detectables. Lo que significa que podemos cambiar las fases absolutas de los segmentos pero las diferencias de fase relativas entre segmentos adyacentes deben preservarse. Entonces la información secreta se inserta solo en el vector de fase del segmento de la primera señal como a continuación:

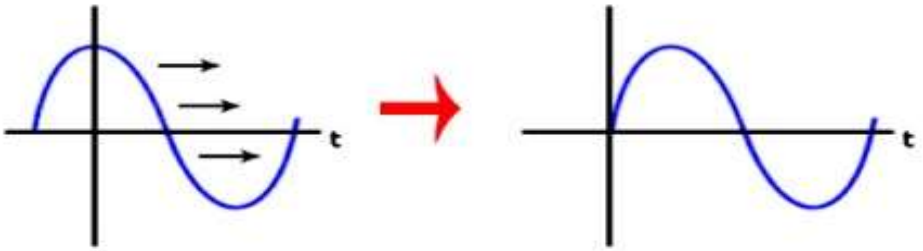
$$nuevafase = \begin{cases} \frac{\pi}{2}\ si\ el\ bit\ de\ mensaje = 0 \\ -\frac{\pi}{2}\ si\ el\ bit\ de\ mensaje = 1 \end{cases} \quad (1)$$

e. Al usar la nueva fase del primer segmento se crea una nueva matriz de fase y las diferencias de fase originales.

f. La señal de sonido es reconstruida al aplicar la transformada discreta de Fourier usando la nueva matriz de fase, la matriz de magnitud original y después volviendo a concatenar los segmentos de sonido.

El receptor debe conocer la longitud del segmento para extraer la información secreta desde el archivo de audio. Luego el receptor puede usar la DFT para obtener las fases y extraer la información secreta. (Jayaram, 2011)

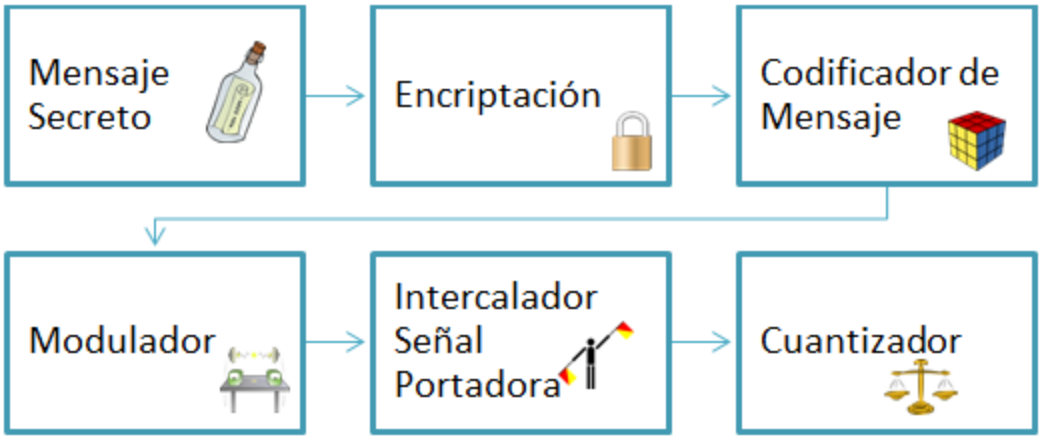
Figura 5. Ejemplo Fase Codificada (Jayaram, 2011)



Señal Original Señal Codificada

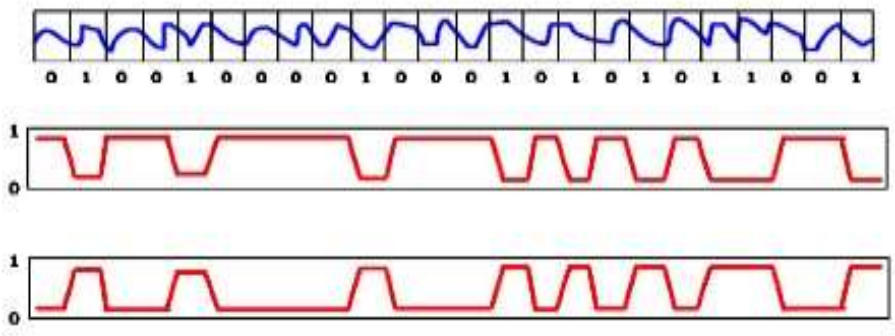
2.2.9 Espectro expandido El método de espectro expandido básico (SS) intenta expandir la información secreta a través del espectro de frecuencia de la señal de audio. Esto es similar a un sistema que usa una implementación del LSB que propaga los mensajes de bit aleatoriamente sobre el archivo entero de sonido. Sin embargo, a diferencia de la codificación LSB, el método del espectro expandido propaga la información secreta sobre el espectro en frecuencia del archivo de audio usando un código que es independiente de la señal actual. Como resultado, la señal final ocupa un ancho de banda que es más de lo que realmente requiere para la transmisión. (Vico, 2010)

Figura 6. Diagrama de bloques método espectro expandido.



2.2.10 Ocultación de eco La ocultación de eco tiene la ventaja de proveer una alta tasa de transmisión y una robustez superior en comparación con otros métodos. Solo un bit de la información secreta puede ser codificado si solo un eco fue producido de la señal original. Por lo tanto, antes que comience el proceso de codificación la señal original es desglosada en bloques. Una vez termina el proceso de codificación, los bloques son concatenados juntos para crear la señal final. La figura 7 muestra la ocultación de eco. (Jayaram, 2011)

Figura 7. Ejemplo de ocultación de eco (Jayaram, 2011)



2.2.11 Transformada discreta del coseno En general, cualquier trasformación a la que se someta una imagen, pretende encontrar una función matemática que se "adapte a las imágenes" y por tanto debe presentar las siguientes características:

a) Permite pasar de valores de brillos o componentes a unos elementos matemáticos denominados coeficientes.

b) La transformación es reversible.

c) Debe concentrarse la energía en pocos coeficientes, para así poder desechar otros muchos con valores bajos sin cometer apenas error.

d) Los coeficientes deben ser lo más independientes posibles. Con esto se consigue que la información se encuentre concentrada sobre cada coeficiente y no sobre el conjunto de ellos.

e) La transformación debe ser sencilla de implementar.

Fórmula Matemática del DCT. La DCT se define como:

$$C[u,v] = \frac{2}{\sqrt{MN}} K(u)K(v) \sum_{m=0}^{M\_1} \sum_{n=0}^{N-1} x[m,n] \cos\frac{(2m+1)\pi u}{2M} \cos\frac{(2n+1)\pi v}{2N} \qquad (2)$$

Donde,

x[m, n] es una matriz de muestras (imagen)

$$x[m,n] = \begin{bmatrix} x_{0,0} & x_{1,0} & \dots & x_{M-1,0} \\ x_{0,1} & \cdots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ x_{0,N-1} & \dots & \dots & x_{M-1,N-1} \end{bmatrix} \quad \begin{bmatrix} C_{0,0} & C_{1,0} & \dots & C_{M-1,0} \\ C_{0,1} & \cdots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ C_{0,N-1} & \dots & \dots & C_{M-1,N-1} \end{bmatrix} \quad (3)$$

$$= C[u,v]$$

m, n u, v son posiciones en las matrices

M, N son las dimensiones de la matrices

$$K(u) = \frac{1}{\sqrt{2}} \qquad si\ u = 0 \qquad K(u) = 1 \qquad si\ u \neq 0 \qquad (4)$$

$$K(u) = \frac{1}{\sqrt{2}} \qquad si\ v = 0 \qquad K(u) = 1 \qquad si\ v \neq 0 \qquad (5)$$

La IDCT se define como:

$$x\,[m,n]$$

$$= \frac{2}{\sqrt{MN}} \sum_{u=0}^{M-1} K(u) \sum_{v=0}^{N-1} K(v) C[u,v]\, cos \frac{(2m+1)\pi u}{2M} cos\ \ \frac{(2n+1)\pi v}{2N} \qquad (6)$$

Los valores de los coeficientes C[u,v] formarán una matriz de M filas y N columnas donde:

$$0 \leq u \leq M \qquad y \qquad 0 \leq v \leq N$$

Funciones base para la DCT.

Si implementáramos estas fórmulas con cualquier lenguaje, no cabe duda que obtendríamos estos mismos resultados de forma más rápida y menos tediosa que de forma manual. No obstante, esto resultaría bastante poco eficiente

dado el alto grado de anidamiento del código (varios “for” anidados). Con objeto de mejorar esto, podemos introducir una serie de constantes que van a mejorar la eficiencia del código.

Para la DCT particularizando para M = N =8 aparecerán 64 funciones base:

f [m , n] u =0 v=0
f [m , n] u =0 v=1
....
f [m , n] u =7 v=0
....
f[m, n] u=7 v=7

Una función base genérica f [m, n] se obtendría, particularizando “u” y “v”, sobre la expresión:

$$cos\frac{(2m+1)\pi u}{2M}cos\ \frac{(2n+1)\pi v}{2N} \qquad (7)$$

Particularizando además para N = M = 8, cada función base será una matriz de 8x8 valores hasta un total de M x N funciones base.

Ejemplo:

$$f[m,n]_{u=0\ v=0} = cos\frac{(2m+1)\pi 0}{16}cos\frac{(2n+1)\pi 0}{16} = 1 \qquad (8)$$

Independientemente de “m” y “n”. Se obtiene una matriz de 8x8 unos.

Análogamente, si se calcula la función base para el caso de u = 3 v = 5 se obtiene la siguiente matriz:

$$f[m,n]_{u=3\ v=5}$$
$$= cos\frac{(2m+1)\pi 3}{16} cos\frac{(2n+1)\pi 5}{16} \quad dependiendo\ de\ \text{"m"}\quad y\ \text{"}n\text{"} \tag{9}$$

$$f[m,n]_{u=3\ v=5}$$
$$= \begin{bmatrix} 0.46 & -081 & 0.16 & 0.69 & 0.69 & -0.16 & 0.81 & -0.46 \\ -0.10 & 0.19 & 0.03 & -0.16 & 0.16 & 0.03 & -0.19 & 0.10 \\ -0.54 & 0.96 & -0.19 & -0.81 & 0.81 & 0.19 & -0.96 & 0.54 \\ -0.30 & 0.54 & -0.10 & -0.46 & 0.46 & 0.10 & -0.54 & 0.34 \\ 0.30 & -0.54 & 0.10 & 0.46 & -0.46 & -0.10 & 0.54 & -0.30 \\ 0.54 & -0.96 & 0.19 & 0.81 & -0.81 & -0.19 & 0.96 & -0.54 \\ 0.10 & -0.19 & 0.03 & 0.16 & -0.16 & -0.03 & 0.19 & -0.10 \\ -0.46 & 0.81 & -0.16 & -0.69 & 0.69 & 0.16 & -0.81 & 0.46 \end{bmatrix} \tag{10}$$

Una representación gráfica de gran utilidad vendría dada al asignar distintas tonalidades de brillo a cada uno de estos valores comprendidos entre 1 y -1 (por tratarse de un coseno) asignando como extremos el color negro al -1 y el blanco al +1. Mientras tanto los valores intermedios tomarían brillos entre 0 y 255 (8 bits para el brillo). La representación más sencilla es la correspondiente a la función

f [m ,n] u=0 v=0 para la que se obtiene una matriz de 8 x 8 unos y en consecuencia una representación de 8 x 8 blancos.

Existe otra forma para obtener la DCT e IDCT que resulta más eficiente. En su aplicación es necesario conocer de antemano el tamaño de la muestra a codificar.

Para el caso de nuestro ejemplo M = N = 2 y por tanto:

$$C[u,v] = K(u)K(v)\sum_{m=0}^{1}\sum_{n=0}^{1} x[m,n]\cos\frac{(2m+1)\pi u}{2M}\cos\frac{(2n+1)\pi v}{2N} \tag{11}$$

Ecuación que podemos expresar:

$$C[u,v] = \sum_{m=0}^{1} K(u)cos\frac{(2m+1)\pi u}{4} \sum_{n=0}^{1} x[m,n]\cos\frac{(2n+1)\pi v}{4} \qquad (12)$$

Matricialmente esta ecuación puede escribirse como:

$$[C] = [A]^{\gamma}.[X].[A]\ donde\ [A]\ tiene\ la\ expresión: \qquad (13)$$

$a_{ij} = kcos\frac{(2j+1)\pi i}{4}$ con k=1 cuando i≠ 0

Con k= $\frac{1}{\sqrt{2}}$ cuando i= 0

La expresión para IDCT se obtiene despejado [x]:

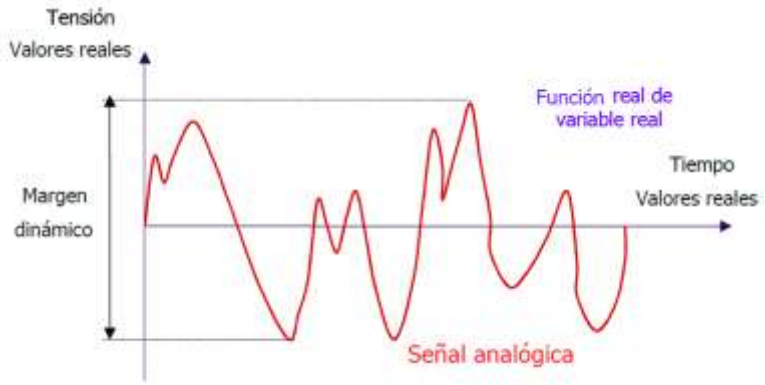
$$[X] = [A].[C][A]^{T} \qquad (14)$$

Se trata en este punto de asociar brillos (escala de grises) a los valores de los coeficientes, para poder interpretar mejor las variaciones relativas entre los diferentes coeficientes obtenidos tras la transformación. Realmente lo que se hace es tomar el valor absoluto de los coeficientes puesto lo que nos interesa ahora es observar las posiciones en las que se concentra mayor energía y no así el valor de los coeficientes.

2.2.12 Conversión Análogo/Digital Un sistema analógico es aquel en que la magnitud física que se transmite puede tomar cualquier valor numérico dentro del margen de trabajo del propio sistema. En la mayor parte de los casos, éste valor es directamente proporcional a la magnitud física que se desea transmitir. Así, en el caso de un sistema de transmisión de audio, la tensión que se transmite por el medio es directamente proporcional a las variaciones de presión que se capturan

en la proximidad del transductor. En otros casos, la magnitud física original puede haber sido transformada para adaptarla a las características del medio de transmisión o almacenamiento. En la figura 8 se representa esquemáticamente una señal de tensión analógica donde se indica que, desde un punto de vista formal, puede considerarse como una función real (tensión) de variable real (tiempo). El margen de valores que toma la función se denomina margen dinámico y suele estar acotado entre un valor mínimo y uno máximo que dependen de las limitaciones físicas de los sistemas de transducción.

Figura 8. Representación esquemática de una señal analógica (Castro Lechtaler & Fusario, 2013)

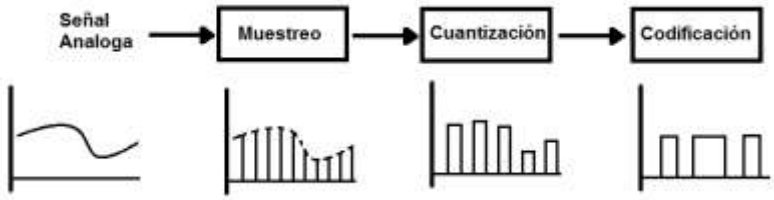


Sin embargo existe un conjunto muy importante de señales denominadas digitales, que implica un conjunto discreto tanto en el dominio del tiempo como para la amplitud.

El proceso de digitalización de una señal analógica parece, en principio, algo natural. De hecho, la representación de la señal analógica que se ha realizado en la figura 8 no es más que una digitalización previa de la función real. En este caso, la digitalización consiste en tomar un número elevado de muestras de la función en el eje temporal (Castro Lechtaler & Fusario, 2013). Si éste número es suficientemente elevado el ojo no será capaz de distinguir entre una

representación realmente analógica (como la que obtendríamos representando la función utilizando un lápiz y un papel) y su equivalente digital. En la figura 9 se muestran en diagramas de bloque el proceso de digitalización de una señal analógica.

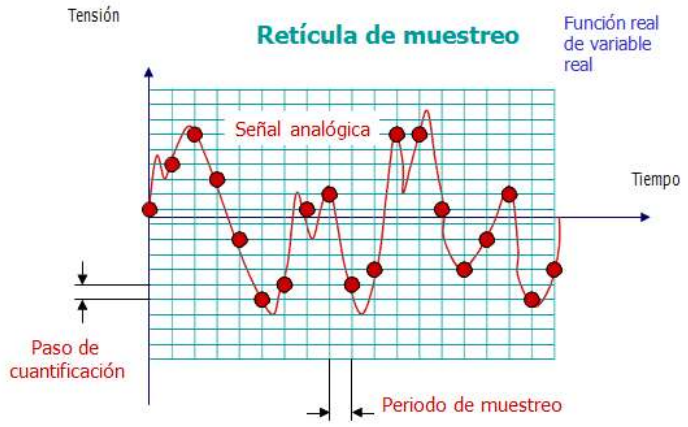
Figura 9. Proceso general de conversión análogo/digital. (Castro Lechtaler & Fusario, 2013)



Etapas Principales: Muestreo, Cuantificación Y Codificación.

El concepto básico de la digitalización de señales se muestra en la figura 10. La señal analógica original se aproxima mediante la ayuda de una retícula rectangular. La separación entre los elementos de la retícula en el eje de abcisas es constante y se conoce como el periodo de muestreo de la señal. Intuitivamente, parece que cuanto menor sea el valor del periodo de muestreo mejor representada quedará la señal analógica.

Figura 10. Conversión análogo/digital. (Castro Lechtaler & Fusario, 2013)



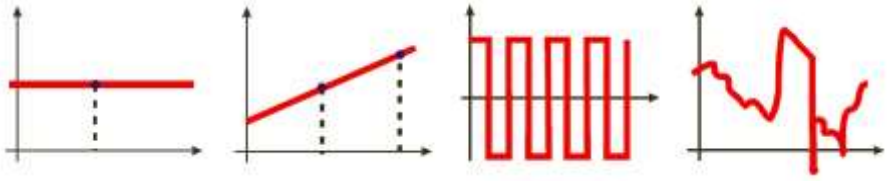
No obstante, el teorema del muestreo establece, en función de las características de la señal, un límite al valor del periodo de muestreo. Si se verifica éste requisito mínimo, la señal analógica puede ser recuperada exactamente a partir de sus muestras sin ningún tipo de ambigüedad.

Bajo estas condiciones, la representación digital de la señal no mejora aunque se reduzca el intervalo de muestreo en el eje temporal. Se analizará con algo más de detalle las condiciones que establece el teorema del muestreo.

Teorema de muestreo

De forma general, la selección de los puntos de muestreo para una señal dependerá fuertemente del conocimiento previo que tengamos de dicha señal. En términos matemáticos, sería equivalente al número de puntos necesarios para determinar de forma unívoca una curva. Así, para una función (señal) continua bastará con un único punto (de muestreo), pues no hay variación con el tiempo y nuestra única incógnita es el valor de la amplitud. En el caso de tratarse de una recta (una señal puramente creciente), entonces bastarían dos puntos. No obstante, nótese que la determinación de los parámetros de una señal cuadrada (amplitud y periodo), lo cual implica dos incógnitas, no resulta de la simple recogida de dos puntos de muestreo, salvo que se dispusiera de alguna información previa sobre su frecuencia. En el otro extremo, para una señal de la que se carece de información, es decir, de dinámica imprevisible incluyendo discontinuidades, el número teórico de puntos de muestreo que se precisa se hace infinito. En la figura 11se ilustra este proceso.

Figura 11. Selección de los puntos de muestreo.



En la mayoría de los casos será suficiente con disponer de información sobre el dominio de actuación de la señal, es decir, su rango de amplitudes, y sobre su frecuencia, que tiene una influencia directa en la determinación de los puntos de muestreo. Estas limitaciones pueden corresponderse con la señal en sí o bien con los propios requerimientos del receptor o usuario.

Así, un ejemplo ilustrativo de señal con rango de frecuencia limitado por su propia naturaleza sería el de los sistemas de audio, cuyo rango de frecuencias queda limitado a menos de 20 KHz, pues dichos armónicos superan los límites de percepción del oído humano (Tomasi, 2003). Nótese que en este caso la fuente (por ejemplo una orquesta) podría generar armónicos de mayor frecuencia, que si se registraran no tendrían utilidad para el observador pero, por el contrario, "consumirían" potencia y, lo que es incluso peor también ancho de banda. La señal de audio únicamente requiere disponer de contenido espectral hasta esta frecuencia máxima de 20 KHz para que pueda ser reproducida con total fidelidad (Rincón Rivera, 2000).

Para evitar la inclusión de frecuencias inservibles, las señales se limitan en banda, es decir, se anulan por encima de un determinado valor de frecuencia. Se dice que una señal es de banda limitada cuando su contenido espectral es nulo a partir de una determinada frecuencia F. Así, la frecuencia de muestreo F_s depende de las características de la señal y su estadística de variación en el tiempo. Evidentemente, cuanto más rápidos son los cambios temporales que experimenta la señal, más elevada debe ser la frecuencia de muestreo a fin de evitar que se

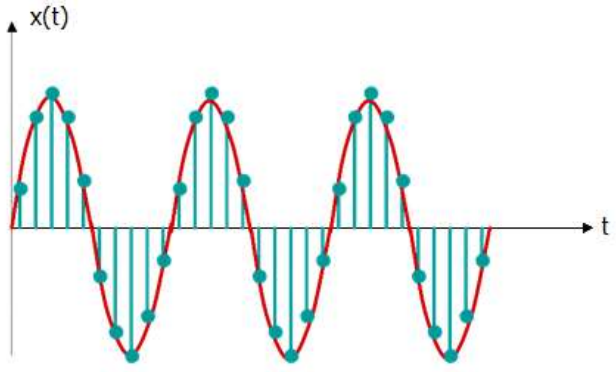
produzca una pérdida de información significativa. La relación entre las variaciones temporales de la señal y la frecuencia de muestreo mínima se establece mediante el teorema del muestreo o criterio de Nyquist.

El teorema del muestreo establece que cuando una señal es de banda limitada puede muestrearse sin que se produzcan pérdidas de información utilizando una frecuencia de muestreo mayor que el doble de su ancho de banda.

$$F_s = \frac{1}{T_s} \geq 2 * F \tag{15}$$

De acuerdo con esta expresión, la frecuencia de muestreo mínima para poder trabajar con señales de audio de alta fidelidad estaría situada por encima de los 40 KHz (20 KHz de ancho de banda). En la figura 12 se ilustra el proceso de muestreo sobre una componente sinusoidal. En este ejemplo se toman un total de 10 muestras por periodo por lo que se verifica, sin ningún tipo de problemas, el teorema del muestreo.

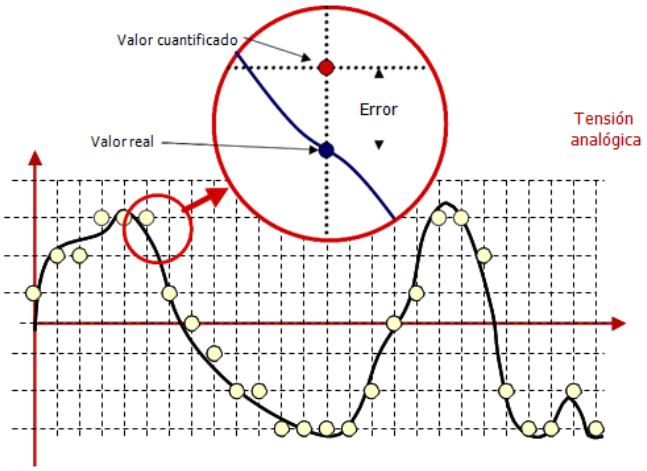
Figura 12. Muestreo de una componente sinusoidal. (Castro Lechtaler & Fusario, 2013)



Paso de cuantificación y número de bits.

El paso de cuantificación define la precisión con la que se codifican las muestras de la señal, como si fueran a representarse en un display, es decir en un visor con un número de decimales fijo, y está directamente relacionado con el número de bits asignado a cada muestra. La figura 13 muestra el proceso de discretización de la amplitud de la señal e indica cómo se introduce un error entre el valor real de la señal analógica y el valor con que se codificará la muestra una vez digitalizada.

Figura 13. Cuantificación y ruido de cuantificación. (Castro Lechtaler & Fusario, 2013)



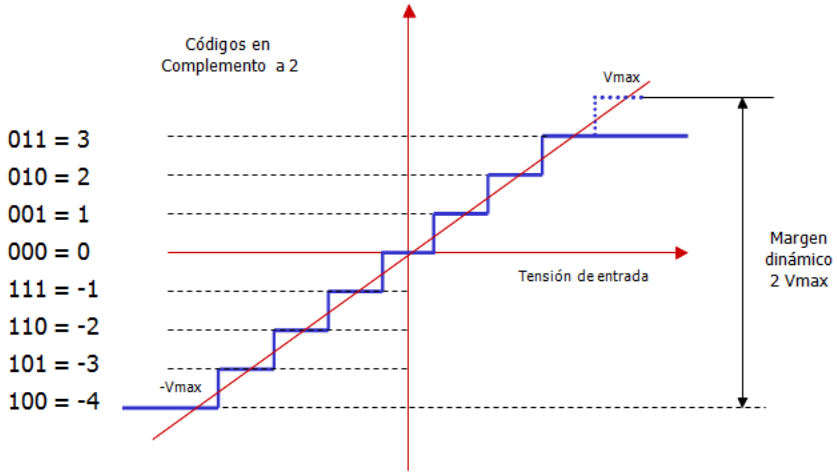
El proceso de digitalización introduce por tanto un error aleatorio en la amplitud de la señal que es equivalente a la adición de una componente de ruido. En efecto, podemos suponer que el valor cuantificado corresponde al de la señal original más un ruido 'de cuantización', que se ha superpuesto con la señal, dando lugar al valor que realmente adquiriremos. El ruido puede ser tanto positivo como negativo y su valor máximo es igual a la mitad del paso de cuantificación.

Codificación.

La codificación PCM (Pulse Code Modulation) consiste en asignar un código binario a cada nivel de cuantificación de la señal. Con ello, cada una de las muestras queda codificada con una palabra de un número de bits fijo. Este tipo de codificación se utiliza en el sistema Compact Disc Digital Audio y en diversos sistemas de vídeo digital no comprimidos. En formatos de audio y vídeo comprimidos (ADPCM, Minidisc, MP3, MPEG) la codificación de la señal en PCM suele constituir la primera fase del proceso de compresión (Rincón Rivera, 2000). Una vez la señal se ha convertido a PCM se analiza y procesa con el objeto de reducir el número total de bits con el que se representa, codificándola en formatos alternativos que pueden resultar más o menos complejos y/o eficientes (Tomasi, 2003). En cualquier caso, para reproducir una señal comprimida también es habitual pasarla previamente al formato PCM y posteriormente convertirla a una señal analógica que pueda aplicarse a los altavoces o al sistema de representación gráfica.

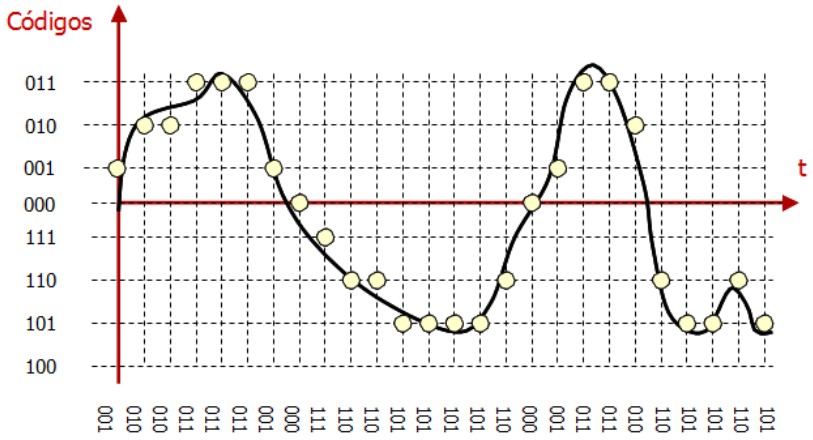
En la figura 14 se muestran los códigos binarios que serían asignados a un cuantificador del tipo uniforme, es decir, que divide el margen dinámico en divisiones del mismo valor. La codificación utilizada se conoce como codificación en complemento a 2 y es la más utilizada en la codificación de audio y vídeo. Los códigos binarios de los niveles de cuantificación positivos empiezan siempre por cero y a continuación indican el número de pasos de cuantificación asignados. De este modo, para codificar el nivel de cuantificación 3 utilizaremos el código 011; donde el primer 0 indica que el nivel es positivo y los otros dos dígitos codifican el número 3 ($1x2^1+1x2^0$).

Figura 14. Codificación PCM en complemento a dos. (Tomasi, 2003)



En la gráfica de la figura 15 se representa esquemáticamente el proceso de codificación de una señal. A cada muestra se le asigna el código binario correspondiente que representará el nivel de amplitud de la señal. Los datos resultantes pueden almacenarse en una memoria para su posterior tratamiento, transmitirse por algún medio adecuado o registrarse en un soporte de almacenamiento masivo.

Figura 15. Ejemplo de codificación de una señal.



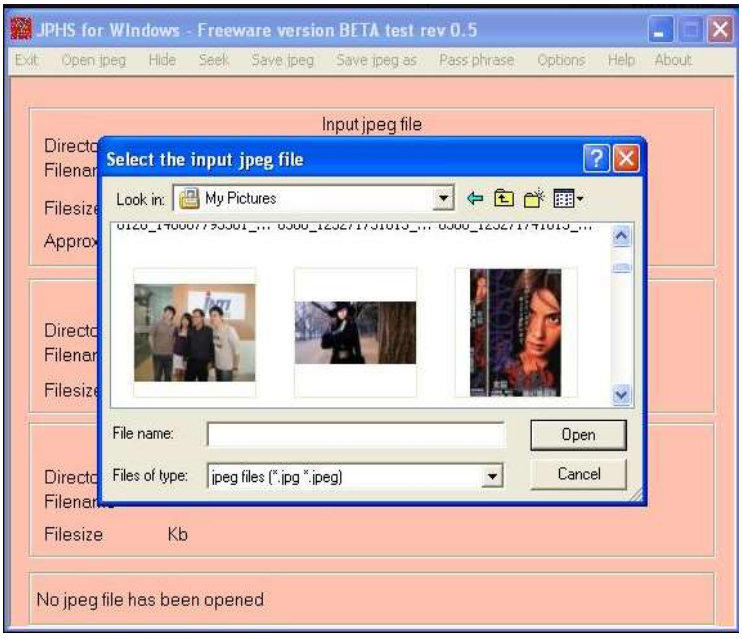
2.3 SOFTWARE ESTEGANOGRÁFICOS

2.3.1 Jphide y Jpseek Son programas que permiten ocultar un archivo en una imagen visual jpeg. Hay varias versiones de programas similares disponibles en Internet, pero JPHIDE y JPSEEK son bastante especiales. El objetivo de este diseño no fue simplemente para ocultar un archivo, sino más bien para demostrar que es imposible detectar que el archivo host lleva un archivo oculto. Dada una imagen, si se aplica una baja tasa de bits por debajo del 5%, no es posible concluir con certeza que el archivo host contiene datos insertados. Por encima de 15% los efectos empiezan a ser visibles a simple vista. Por supuesto que algunas imágenes son mucho mejores que otras cuando se utiliza un archivo de host. Un cielo azul sin nubes sobre una cubierta de nieve no es muy bueno para realizar un ocultamiento ideal. Una cascada en un bosque es la imagen perfecta. (Latham, 2009).

A continuación se dará una pequeña guía de cómo hacer uso de este software.

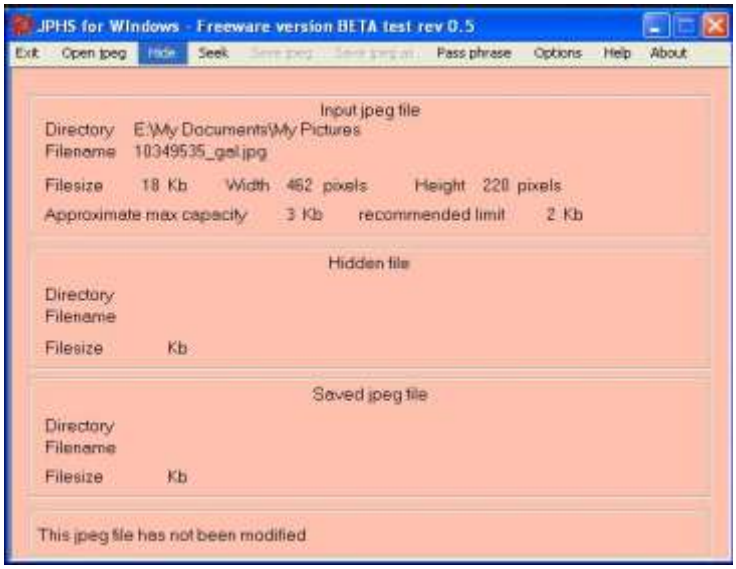
Primero seleccionamos la imagen que se quiere usar.

Figura 16. Elección de la imagen (Latham, 2009)



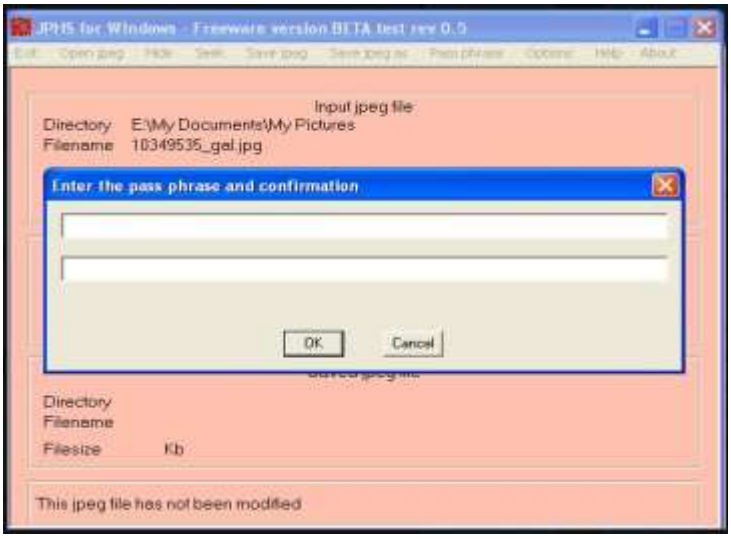
Una vez seleccionada la imagen, haga clic en Hide para ocultar un mensaje.

Figura 17. Pestaña para ocultar mensaje (Latham, 2009)



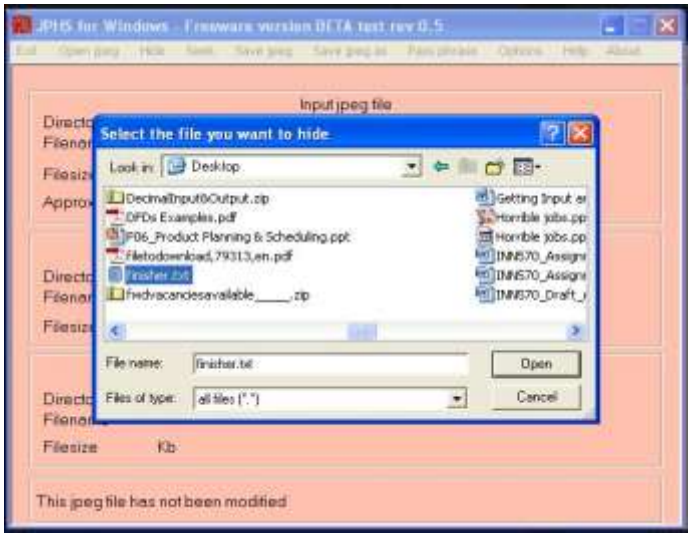
A continuación se introduce la contraseña que se utilizara para abrir el archivo.

Figura 18. Ventana para inserción de contraseña (Latham, 2009)



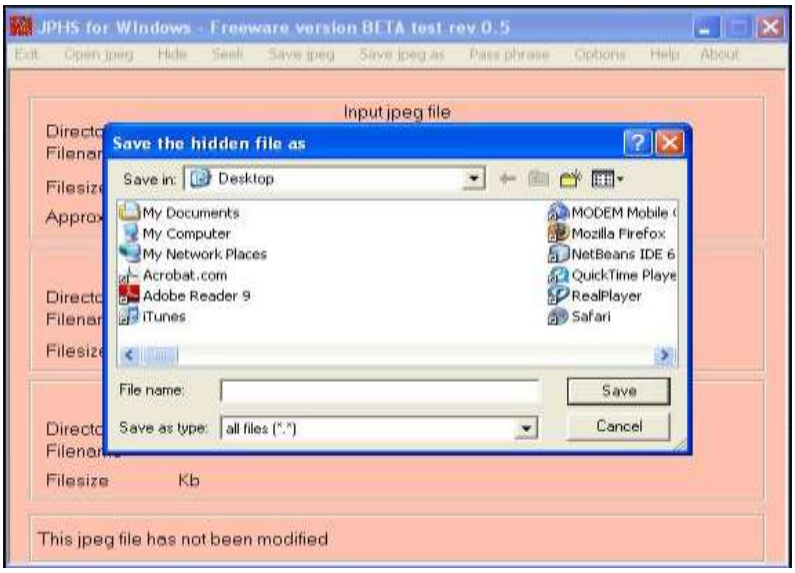
Luego hay que seleccionar el archivo que se desea ocultar. El archivo no puede ser más grande que la imagen.

Figura 19. Elección del archivo a ocultar (Latham, 2009)



Para finalizar debemos guardar el archivo y enviarlo a su destinatario, para que el destinatario pueda ver el verdadero mensaje debe conocer la calve utilizada en el proceso.

Figura 20. Ventana para guardar el esteganoarchivo **(Latham, 2009)**



2.3.2 MP3STEGO Existe un interés creciente para ocultar información en formato MP3 o WAV, ya que ofrecen calidad en relación de compresión de 11 a 1 (128 kilobits por segundo). Esto da una muy buena oportunidad para ocultar la información.

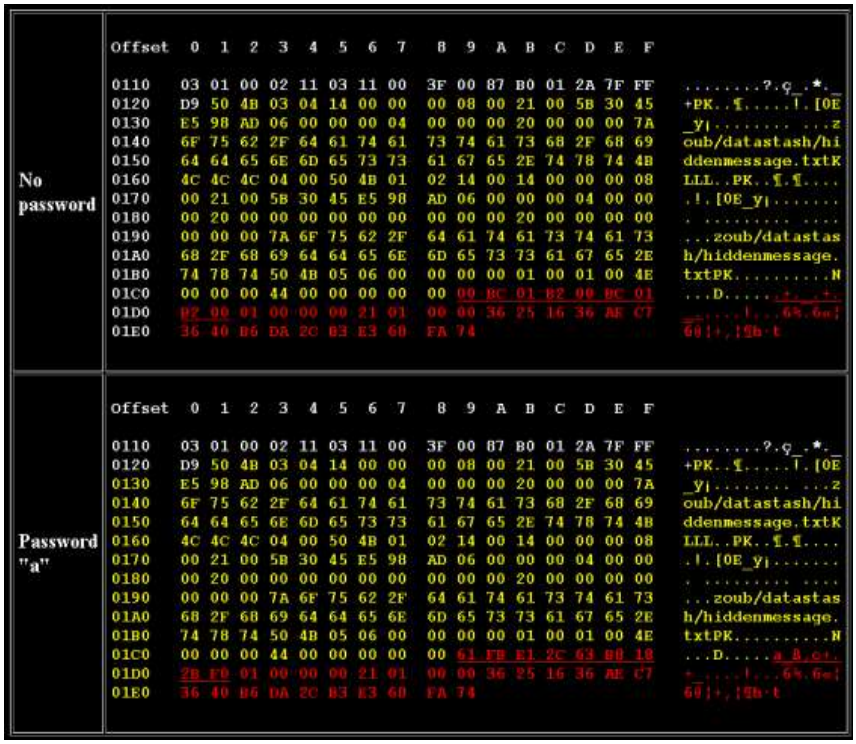
MP3Stego oculta información en archivos MP3 durante el proceso de compresión. Los datos son primero comprimidos, cifrados y luego escondidos en un flujo de bits .Cualquier intruso puede descomprimir el flujo de bits pero se borra gran parte de la información oculta, este es el único ataque existente hacia este software a cambio de una pérdida considerable del mensaje.

El proceso de ocultación tiene lugar en la codificación de la capa III, en el innerloop. El bucle interior cuantifica los datos de entrada y aumenta el tamaño del

paso cuantificador hasta que los datos pueden ser codificados con el número disponible de bits. Otro bucle comprueba que las distorsiones introducidas por la cuantización no superen el umbral definido por el modelo psicoacústico. Al codificar los bits de paridad van cambiando la condición de bucle final del bucle interior. Sólo los valores part2_3_length elegidos al azar son modificados, la selección se realiza utilizando un generador de bits pseudo-aleatorio basado en SHA-1. (Fabian, 2006)

2.3.3 Data STASH Este sistema esteganográfico pertenece a una compañía de Singapur llamada Skyjuice Software. Comercializan un producto esteganográfico llamado Data Stash. El coste del sistema varía entre los 20 y los 350 dólares americanos dependiendo del número de licencias que se adquieran. Los ficheros que se tienen que ocultar bajo las cubiertas se comprimen con ZIP y el resultado se concatena al final de la imagen que se utiliza como cubierta. Supuestamente este sistema incorpora un sistema de cifrado para proteger aún más la información que se esconde. Según los autores, el sistema que incorpora es Blowfish, un algoritmo de cifrado bastante potente en el cual sin conocer la contraseña, es imposible extraer la información que cifra. Se ha utilizado un Análisis de herramientas esteganográfica como lo es “hiddenmessage.txt” que será el fichero ocultado bajo una imagen JPG que se utilizará como cubierta. Es importante comentar que se realizaron dos pruebas: en la primera no se usó la opción de utilizar contraseña, mientras que en la segunda se optó por usar contraseña, por lo que los datos a ocultar tendrían que estar cifrados mediante el algoritmo Blowfish bajo la contraseña marcada. A continuación se presentará una imagen en la que se comparan ambas pruebas. (Cano, 2004)

Figura 21. Comparación de las pruebas de Data Stash (Cano, 2004)



Como se puede observar en la imagen, en ambas pruebas se distinguen tres bloques bien diferenciados que son:

- Imagen JPG que actúa como cubierta (resaltado en blanco en ambas pruebas).
- Compresión ZIP del fichero ocultado (resaltado en amarillo en ambas pruebas).
- Datos de relleno introducidos por Data Stash v1.1b (resaltado en rojo en ambas pruebas).

En la comparativa entre las dos pruebas se puede ver que tan sólo los bytes que se encuentran subrayados en rojo son los que han variado (en una se usaba contraseña y en la otra no). Por tanto, se puede deducir que no existe ningún cifrado mediante Blowfish de los datos ocultados. Para conocer el contenido del mensaje, se utilice o no contraseña, tan sólo bastaría con extraer el bloque resaltado en amarillo y salvarlo con extensión .zip y abrirlo con un visor de ficheros

ZIP. Posteriormente bastaría con extraer el contenido del fichero comprimido y se obtendría la información ocultada sin ningún problema. (Cano, 2004)

2.4 FICHEROS DE AUDIO

2.4.1 Motions picture Experts Group – Audio Layer 3 (.MP3) Este formato es el más usado en la actualidad, todo ello por su eficaz método de compresión, el cual disminuye el tamaño del archivo. La compresión se realiza eliminando frecuencias que no son audibles por el oído humano. Su única ventaja en la esteganografía podría ser su fácil transporte, pues una vez modificado se puede traficar dicha información de una manera sencilla y a mayor velocidad. (Torres, 2011)

2.4.2 Waveform Audio File Format (.WAV) Este formato de audio es el más famoso de Microsoft. Las cabeceras de estos son enormemente más grandes a los archivos de imagen, contando con un total de 43 bytes en la cabecera. El formato de audio .wav es el mejor para la esteganografía, claro cuando hablamos de audio. Este formato principalmente se caracteriza por su sencillez, pues al no tener mucha compresión es fácil de editar por medio Hexadecimal, la desventaja principal es que por no tener esa compresión estos archivos pesan demasiado. (Torres, 2011)

2.4.3 WMA Es el MP3 (formato con pérdida) propietario de Microsoft. A pesar de que el formato mejora la calidad del MP3, no parece que exista ventaja de utilizar un formato propietario disponiendo de alternativas libres. La explicación es que nuestra música, con el paso de los años, puede quedar en un formato obsoleto o descontinuado por Microsoft, mientras que utilizando formatos libres, cuyos códecs están abiertos y son públicos, la compatibilidad y posibilidad de conversión en diferentes plataformas está garantizada. Además, veremos que existen

alternativas libres que lo superan ampliamente en calidad. Auditivamente, el WMA añade artefactos digitales y pérdidas semejantes al MP3.

2.4.4 OGG Sería el equivalente al MP3-WMA pero libre y gratuito. El código de los códecs es abierto y está disponible para la comunidad, que puede seguir mejorándolo. Es una excelente opción dadas sus excelentes tasas de compresión y su calidad, superior a la calidad de WMA, siendo muy superior al MP3. Podemos decir “a grosso modo” que un OGG a 128 kbps suena mucho mejor que un MP3 a 192 kbps. La lista de reproductores tanto hardware como software que soportan OGG ha ido aumentando y hoy día es muy numerosa.

2.4.5 ATRAC3 Es el MP3 de Sony, es bastante antiguo, pero la tecnología que utiliza es bastante sofisticada, consiguiendo excelentes tasas de compresión con una pérdida de calidad bastante escasa. Aunque el formato es bueno, sólo los reproductores de Sony (y no todos ya que en las nuevas versiones ya han abandonado definitivamente este formato) serán capaces de reproducir ATRAC. ATRAC3plus es un formato mejorado de ATRAC. Auditivamente, se aprecian cortes en determinadas frecuencias en instrumentos secundarios y agudos.

2.4.6 M4A / MP4 / AAC El M4A o AAC es otro formato de audio propuesto por el grupo MPEG, llamado MPEG-4 (el MP3 pertenece a la especificación MPEG-1). Los AAC son por tanto M4A, sólo que quizá sea esta extensión más corriente por ser utilizados masivamente por Apple para sus reproductores iPod. Sólo aclarar que MP4 es un formato de vídeo que contiene el audio en AAC, y se llaman por tanto reproductores de MP4 aquellos que son capaces de reproducir vídeo.

La calidad del AAC es muy superior al MP3, digamos que es comparable a la calidad del OGG. Auditivamente, el AAC añade cierta distorsión en las frecuencias medias y artefactos digitales en instrumentos secundarios.

2.4.7 MPC (MUSEPACK) Es uno de los formatos con pérdida más moderna y sofisticada, además de ser libre y gratuito. Los archivos resultantes son de mayor tamaño, pero las pérdidas son mínimas y la calidad de sonido, excelente. Existen plugins para muchos reproductores de audio, y también reproductores que lo soportan nativamente. El códec no está pensado para bitrates bajos, por lo que si lo que queremos es obtener ficheros de peso mínimo, es mejor utilizar OGG. Es el formato con pérdida que ofrece la mejor calidad de todos y las pérdidas son prácticamente inapreciables.

2.4.8 RA (Real Audio) Este formato fue muy extendido en los 90, ya que comprimía hasta límites insospechados aun cuando la calidad era discreta, haciéndolo idóneo para poder transmitir streaming de audio a través de los lentos módems de 56kbps. Actualmente el formato está siendo desbancado por los streaming de MP3 o WMA, ya que las conexiones hoy día son más rápidas y permiten *streamings* de mayor bitrate. Los ficheros Real Audio requieren tener instalado el reproductor Real Audio, lento, pesado y lleno de publicidad. Es un formato propietario cerrado y carece de interés hoy día ya que tiene una clara tendencia al desuso. (https://lamaquinadiferencial.wordpress.com, 2008)

2.5 INTERFACES DE USUARIO CON MatLab

2.5.1 Introducción a MatLab MatLab es una abreviatura de la frase Matrix Laboratory. Es un entorno informático de análisis numérico y representación gráfica de fácil manejo. Originalmente fue escrito para la enseñanza de álgebra lineal, aunque actualmente es un lenguaje de programación. También permite crear funciones propias y programas especiales (denominados archivos-M) en código MatLab, que se pueden agrupar en las llamadas Toolboxes: colección especializada de archivos-M para trabajar en distintos tipos de problemas, por ejemplo de optimización, de estadística, de ecuaciones diferenciales parciales, etc.

Se puede considerar, por otro lado, que MatLab es una calculadora totalmente equipada aunque, en realidad, es mucho más versátil que cualquier calculadora para hacer cálculos matemáticos. Se trata de una plataforma para el desarrollo de aplicaciones y para la resolución de problemas en múltiples áreas de aplicación.

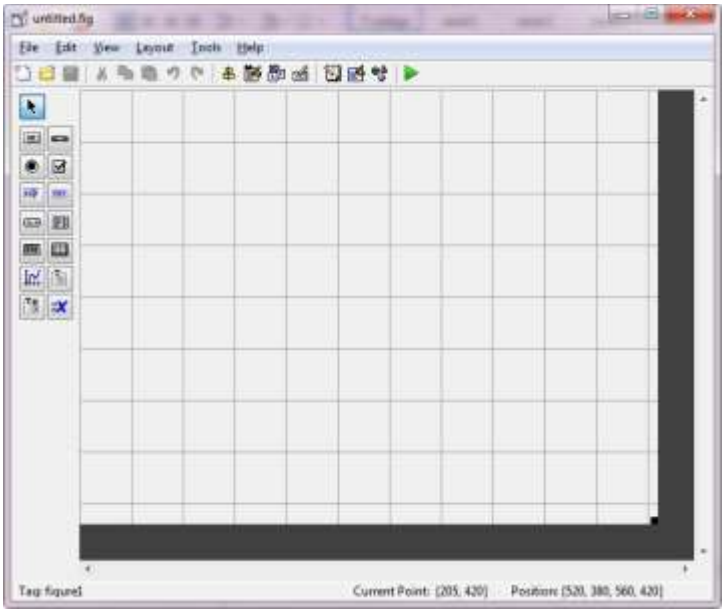
Entre sus utilidades, se encuentra:

- Cálculo matricial y Algebra lineal.
- Polinomios e interpolación. 1
- Regresión y ajuste de funciones.
- Ecuaciones diferenciales ordinarias.
- Integración.
- Funciones y gráficos en dos y tres dimensiones.

Para acceder desde Windows se hace doble click sobre su icono. El programa está accesible desde el siguiente menú: Inicio -> Archivos de Programa -> MatLab. Aparece la pantalla en blanco con una línea de comandos indicada por el símbolo >> desde donde se pueden introducir instrucciones. Para salir de MatLab se utiliza la instrucción quit El elemento básico de MatLab es una matriz rectangular de elementos reales o complejos. MatLab incluso considera los escalares como matrices. Además todas las variables utilizadas en la línea de comandos son almacenadas por MatLab. En caso de dudas se puede utilizar siempre el comando Help. [8]

2.5.2 Guides de MatLab En esta sección se describe los elementos de una programación orientada a objetos, denominada en Matlab, Interfaz Gráfica del Usuario (GUI), el cual va a permitir al usuario, interactuar con el ordenador de una manera rápida en la solución de problemas.

Figura 22. Controles de la interfaz del usuario (GUI)



Los controles son objetos que se ubican dentro de una ventana y permiten mostrar, aceptar o validar datos.

La paleta del formulario editor contiene los controles que usted puede usar en su interfaz de usuario, estos son: Pushbutton, Slider, Togglebutton, Frame, Radio button, Listbox, Checkbox, Pop-up menu, Edit text, Ejes, Static text y Figure.

Éstos componentes son objetos de Matlab y es por lo tanto programable en sus diferentes propiedades, a continuación se presenta información sobre estos componentes.

Pushbutton

Figura 23. Componente PushButton

El control pushbutton genera una acción cuando el usuario hace un clic sobre éste.

Algunas propiedades del control pushbutton:

Para visualizar las propiedades de cualquier control damos doble clic sobre el objeto y aparecerá una ventana con las propiedades del control.

String.- Esta propiedad posee la cadena de caracteres que se mostrará sobre el botón.

Tag - Guide usa la propiedad Tag para etiquetar la función del callback en el archivo m de la aplicación, por defecto se muestra pushbutton.

Programando el callback:

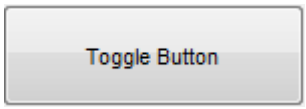
Cuando el usuario pulsa el botón pushbutton, su callback se ejecuta y no devuelve un valor ni mantiene un estado.

El siguiente código ilustra cómo programar el callback de un pushbutton en el archivo m de la aplicación del GUI, para construir una gráfica:

```
function pushbutton1_Callback(hObject, eventdata, handles)
x = 0.2:0.01:8;
y = sin(1./x);
plot(x,y)
grid on
```

Togglebutton

Figura 24. Componente Togglebutton



Los toggle buttons generan una acción e indican un estado binario. Cuando se pulsa el botón togglebutton aparece oprimido y permanece así aun cuando se suelta el botón del mouse al mismo tiempo que el callback ejecuta las ordenes programadas dentro de él.

Algunas propiedades del control Togglebutton:

String.- Esta propiedad posee la cadena de caracteres que se mostrará sobre el botón.

Tag - Guide usa la propiedad Tag para etiquetar la función del callback en el archivo m de la aplicación, por defecto se muestra Togglebutton.

Programando el callback:

La rutina del callback necesita preguntar a togglebutton para determinar en qué estado esta MATLAB y pone el valor igual a Max de la propiedad cuando el togglebutton está oprimido (Max tiene por defecto 1) e igual a Min cuando el togglebutton no está oprimido (Min tiene por defecto 0).

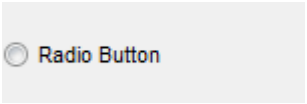
El código siguiente ilustra cómo programar el callback de un togglebutton en el archivo m de aplicación del GUI, para visualizar su estado.

```
function togglebutton1_Callback(hObject, eventdata, handles)

boton_estado = get(handles.togglebutton1,'value') ;
if  boton_estado == 1
        opción = 'Togglebutton  se encuentra  presionado'
elseif  boton_estado == 0
        opción = 'Togglebutton  no se encuentra  presionado'
end
```

Radio button

Figura 25. Componente Radio button



Radio button se utiliza para seleccionar una opción de un grupo de opciones (es decir, sólo un botón está en un estado seleccionado), para activar un radio button, pulsamos el botón del mouse en el objeto.

Algunas propiedades del control Radio button:

El radio button tienen dos estados: seleccionado y no seleccionado al cual se accede a través de su propiedad value.
value = 1(Max), el botón se selecciona.
value = 0(Min), el botón no se selecciona.

Programando el callback:

Los radio buttons son mutuamente exclusivos dentro de un grupo de opciones, los callback para cada radio button se deben poner en la propiedad value igual a 0 en

todos los otros radio buttons del grupo. MATLAB pone la propiedad de value a 1 en el radio button pulsado por el usuario.

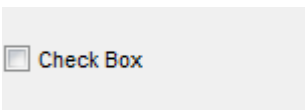
El código siguiente ilustra cómo programar el callback de un radio button en el archivo m de aplicación del GUI, para construir una gráfica con dos opciones:

```
function radio button1_Callback(hObject, eventdata, handles)
 set(handles.radio button2, 'value',0)
opcion1 = get(handles.radio button1,'value');
if opcion1 == 1
x = -3:0.2:4;
y = cos(x);
bar(x,y);  grid on
end

function radio button2_Callback(hObject, eventdata, handles)
set(handles.radio button1, 'value',0)
opcion2 = get(handles.radio button2, 'value');
if opcion2 == 1
x = -3:0.2:4;
y = cos(x);
plot(x,y) ;  grid on
end
```

Check Box

Figura 26. Componente Check Box



Los Checkboxs se utilizan para proporcionar al usuario varias opciones de las que se puede elegir una o más de una cuando se ha pulsado el botón sobre él, e indica su estado como seleccionado o no seleccionado.

Algunas propiedades del control Checkbox:

La propiedad value indica el estado del Checkbox asumiendo el valor del Max o propiedad de Min (1 y 0 respectivamente por defecto):
value = 1(Max), la caja se selecciona.
value = 0(Min), la caja no se selecciona.

Programando el callback

Usted puede determinar el estado actual de un Checkbox desde su callback preguntando el estado de su propiedad en value.

El código siguiente ilustra cómo programar el callback de un checkbox en el archivo m de aplicación del GUI, para determinar el estado del checkbox:

```
function checkbox1_Callback(hObject, eventdata, handles)
boton_estado = get(handles.checkbox1,'value');
if boton_estado == 1
        boton_estado = ' el checkbox ha sido seleccionado'

else
        boton_estado = 'el checkbox no ha sido seleccionado'

end
```

Edit text

Figura 27. Componente Edit Text

Los controles Edit text son campos que permiten a los usuarios el ingreso y salida de información, por defecto se visualiza el texto Edit Text.

Se usa Edit text cuando se quiere ingresar un texto, o una cadena que posteriormente quiera ser evaluada dentro de la interfaz.

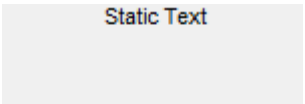
Programando el callback

Para obtener la cadena tecleada por el usuario en un edit, se utiliza la propiedad string del edit en el callback de un pushbutton, el código siguiente ilustra cómo programar el callback en el archivo m de aplicación del GUI.

function pushbutton1_Callback(hObject, eventdata, handles)
usuario_string = get(handles.edit1, 'string ')

Static Text

Figura 28.Componente Static Text



El control Static text se utiliza para mostrar texto que el usuario no puede modificar. El texto estático se usa frecuentemente para etiquetar otros mandos y

proporciona las direcciones al usuario, o indica valores asociados con un deslizador (Slider).
Slider

Figura 29. Componente Slider

Los deslizadores o barras de desplazamiento permiten explorar fácilmente una larga lista de elementos o una gran cantidad de información, y acepta la entrada numérica dentro de un rango específico, permitiéndole al usuario mover una barra corrediza. El desplazamiento de la barra se efectúa presionando el botón del mouse y arrastrando la diapositiva, o pulsando el botón que posee una flecha. La ubicación de la barra indica un valor numérico.

Algunas propiedades del control Slider:

Existen cuatro propiedades que controlan el rango y tamaño del paso del deslizador:
value - contiene el valor actual del deslizador.

Max - define el valor máximo del deslizador, el valor por defecto es 1, el cual puede ser modificado.

Min - define el valor mínimo del deslizador, el valor por defecto es 0, el cual puede ser modificado.

El código siguiente ilustra cómo programar el callback en el archivo m de aplicación del GUI, para obtener el valor del deslizador.

function slider1_Callback(hObject, eventdata, handles)

```
slider_valor = get(handles.slider1,'value')
```

Panel

Figura 30. Componente panel

Un control Panel proporciona un agrupamiento identificable para controles. Los paneles no tienen ninguna rutina de callback asociados con.

Agregando componentes en el panel

Los paneles son opacos. Si se agrega un panel después de agregar componentes que se quiere posicionar dentro del marco, se necesita traer esos componentes con la opción adelante. Se usan las operaciones Bring to front (traer adelante) y Send to back (enviar atrás) en el menú del formulario para este propósito.

Listbox

Figura 31. Componente Listbox

Los Listboxs permiten mostrar una lista de ítems entre los cuales el usuario puede seleccionar uno o más ítems.

Algunas propiedades del control Listbox:

La propiedad string contiene la lista de ítems desplegada en el listbox. El primer ítem en la lista tiene el índice 1.

La propiedad value contiene el índice en la lista de cadenas que corresponde al ítem seleccionado. Si el usuario selecciona múltiples ítems, entonces el value es un vector de índices.

El código siguiente ilustra cómo programar el callback en el archivo m de aplicación del GUI, para conseguir el ítem seleccionado:

```
function listbox1_Callback(hObject, eventdata, handles)
indice = get(handles.listbox1, 'value ')
```

La propiedad ListboxTop es un índice en la serie de cadenas definida por la propiedad string y debe tener un valor entre 1 y el número de cadenas.

Selección simple o múltiple

Los valores de las propiedades Min y Max determinan si los usuarios pueden hacer selecciones simples o múltiples:

Si Max - Min> 1, entonces las cajas de la lista permiten la selección del ítem múltiple.

Si Max - Min <= 1, entonces las cajas de la lista no permiten la selección del ítem múltiple.

Programando el callback

Matlab evalúa el callback del listbox después de que el usuario pulsa el botón sobre un ítem, eso cambia la propiedad de value (es decir, cuando el usuario pulsa el botón en un ítem, y no al pulsar el scrollbar en el listbox).

Esto significa que el callback se ejecuta después del primer clic en un solo ítem o cuando el usuario está haciendo las selecciones múltiples.

En estos casos, se necesita agregar otro componente, como un pushbutton y programar su rutina del callback para preguntar el valor de la propiedad listbox (y posiblemente la propiedad de SelectionType), en la función del archivo m de la aplicación.

ºPop-up menu

Figura 32. Componente Pop-up menu

Los Pop-up menus permiten visualizar una lista de ítems a diferencia de los listbox, esto es posible cuando los usuarios presionan la flecha.

Algunas propiedades del control Pop-up menu:

La propiedad string contiene la lista de cadenas visualizadas en el pop-up menu.

La propiedad value contiene el índice del ítem seleccionado de la lista de cadenas, el primer ítem en la lista tiene el índice 1.

Los pop-up menu son útiles cuando se desea proporcionar varias opciones mutuamente exclusivas a los usuarios, y no usar una mayor cantidad de espacio que una serie de radio button requeriría.

Programando el callback

Se puede programar el callback del pop-up menu para trabajar verificando sólo el índice del ítem seleccionado (contenido en la propiedad value) o usted puede obtener la actual cadena contenida en el ítem seleccionado.

El código siguiente ilustra cómo programar el callback en el archivo m de aplicación del GUI, para conseguir el ítem y usa una declaración del switch (interruptor) para tomar acción basada en el valor obtenido.

```
function pop-up menu1_Callback(hObject, eventdata, handles)
opcion = get(handles.pop-up menu1, 'value ')
switch  opcion
case 1
    caso = 'El usuario seleccionó el primer ítem'
case 2
    caso = 'El usuario seleccionó el segundo ítem'
otherwise
     caso='El usuario seleccionó otro ítem excepto el 1° y el 2°'
end
```

Activando o desactivando controles

Los controles hasta ahora presentados pueden ser activados o desactivados para responder al botón del mouse usando la propiedad enable (habilitado), los estados son los siguientes:

On . - El control está habilitado
Off. - El control no está habilitado

Cuando un control está inhabilitado no ejecuta su rutina del callback. (Marchena, 2013)

Método DCT (Discrete Cosine Transform, Transformada Discreta Coseno).

Las llamadas transformadas son herramientas matemáticas que permiten representar cualquier función - continua o discreta - a través de una serie de coeficientes.

En general, se necesitarán un número infinito de coeficientes en el caso continuo, y tantos coeficientes como valores en el caso discreto, para poder representar de manera absolutamente precisa la imagen. La ventaja que tiene trabajar con transformadas es que la mayor parte de la información suele estar concentrada en un número relativamente pequeño de coeficientes, por lo que se pueden obtener buenas aproximaciones de la imagen original a partir de un subconjunto de la totalidad de sus coeficientes, que serán más o menos precisas en función del número de coeficientes que se conserven.

Los formatos de compresión de archivos multimedia más comunes emplean distintos tipos de transformada, como la Transformada Discreta del Coseno. La imagen se divide en regiones de 8 x 8 píxeles de tamaño. A cada trozo se le aplica una transformada, y los coeficientes resultantes se truncan, teniendo en cuenta que los sentidos del ser humano son más sensibles a determinadas características, se les da prioridad a los que mejor se percibe, para que las distorsiones sean poco perceptibles. Finalmente se aplica un algoritmo de compresión sin pérdida al resultado, que permite recuperar exactamente los mismos datos que fueron obtenidos para cada una de las

muestras durante el proceso de digitalización, eliminando la redundancia de la cadena de bits correspondientes, de forma que al descomprimirla se obtiene una copia idéntica a la original, y se obtiene el archivo final.

Si se quiere ocultar un mensaje dentro de una imagen, habrá que manipular directamente los coeficientes, e introducir en ellos los bits del mensaje. Como es lógico, ya que los bits que se preservan en los coeficientes son los que mayor cantidad de información transportan, también serán más sensibles a alteraciones, por lo que se podrá ocultar muchos menos bits del mensaje huésped si se quiere mantener un nivel de distorsión en el resultado final que sea realmente imperceptible.

3. METODOLOGÍA PROPUESTA

3.1 MÉTODO PROPUESTO

CODIGO FUENTE Y DISEÑO DE GUI

El sistema utiliza una portadora estéreo en formato WAV con profundidad de 16 bits principalmente a una frecuencia de muestreo estándar de 44.1 KHz estéreo. Esta frecuenta es elegida porque es el formato más común y el más usado en los dispositivos de audio en la actualidad, también es compatible con frecuencias de muestreo superiores, siempre y cuando sean a 16 bits. Ya que el oído humano sano es sensible a las frecuencias entre los 20 Hz y 20 KHz se pueden aprovechar las frecuencias que superan el rango del espectro audible, con base en eso se desarrolla una técnica en donde es posible ocultar información secreta en un archivo de audio que es indistinguible del original al oído humano.

El sonido se muestrea y se escala en Matlab como enteros de 8 bits, la longitud total del archivo es almacenada en un vector, donde también se crea una cabecera inicial en la que se incorpora la longitud del archivo en 10 bits y aparte el nombre en 150 bits, ambas longitudes son de tamaño fijo y también manejan el formato entero de 8 bits; con la información anterior se recrea un nuevo vector con los valores tanto de la longitud original como del nombre y con la información del archivo de audio, en el que se distribuyen 4 bits para el canal izquierdo y 4 bits para el derecho de la portadora WAV estéreo, el proceso puede verse en el diagrama de bloques de la figura 33.

Es necesario que la portadora tenga los parámetros mínimos establecidos para que la operación fluya sin problemas, pero los archivos a ocultar pueden ser de

cualquier tipo siempre y cuando tengan la extensión .mp3 para que puedan ser utilizadas por el programa, por ejemplo, sí se tuviera una información importante comprimida en un archivo .zip y se quisiera ocultar, se puede cambiar la extensión .zip por .mp3 porque el algoritmo de programa es multipropósito pero está enfocado a la ocultación de mp3, de igual forma creará un estego-archivo reproducible tal como si se hubiera usado un mp3 normal. Lo que varía en el proceso, es que el mp3 generado al no contener información de audio real no va a ser reproducible, pero se puede usar al devolver su extensión a la original.

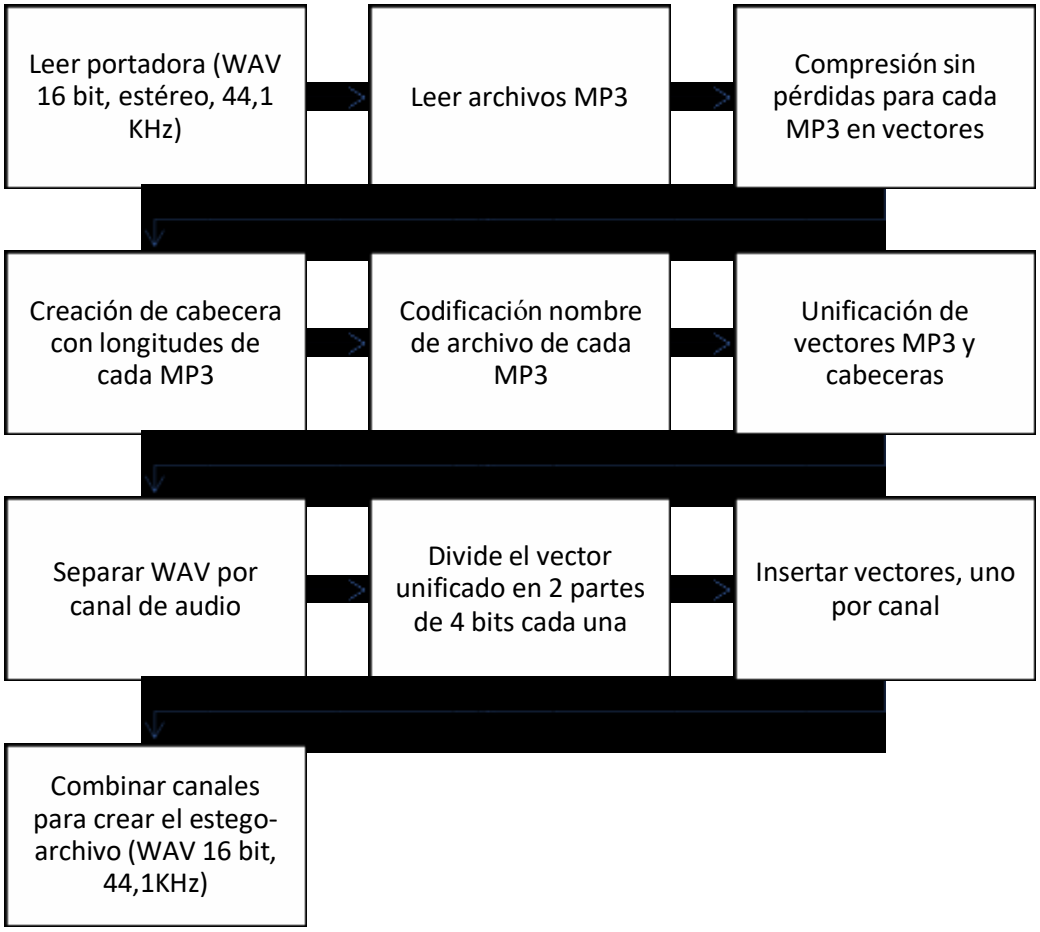
En el receptor el proceso se realiza de forma inversa en el cual se toma el estego-archivo, y a partir de ahí, se comienza a reconstruir el vector con información de los archivos incrustados por el emisor para luego extraer cada uno en una carpeta generada por el programa, este proceso puede verse en el diagrama de bloques de la figura 34.

A continuación se describen los procesos ejecutados por el algoritmo propuesto para el emisor y receptor.

Diagrama de bloques

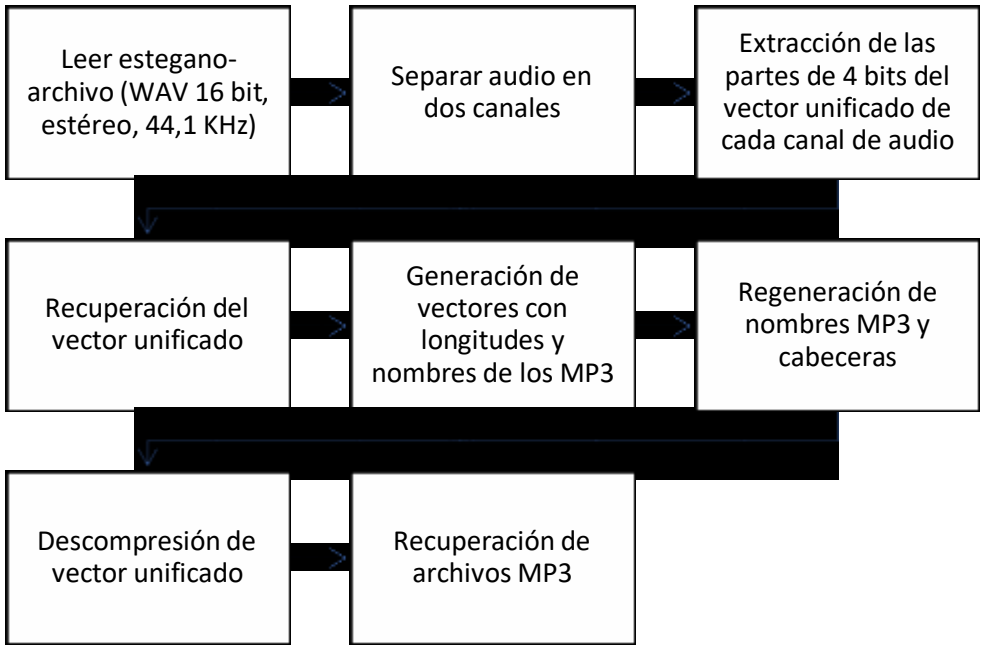
Emisor

Figura 33. Diagrama de bloques emisor



Receptor

Figura 34. Diagrama de bloques receptor



3.2 INTERFAZ GRÁFICA

En esta sección se muestra la GUI diseñada y la función que ejecuta cada uno de los botones que contiene.

3.2.1 Menú principal. Esta interfaz simplifica la compresión al usuario final, el cual puede elegir únicamente el proceso que desee.

Figura 35. Menú Principal

3.2.2 Interfaz del emisor.

Figura 36. Interfaz del emisor

Botón portadora WAV

El botón *portadora WAV* de la sección 1 de la figura corresponde a la función que lee el archivo de audio en formato WAV, se recomienda que este archivo tenga como mínimo una frecuencia de muestreo de 44.1 KHz y una profundidad 16 bits para que el sistema opere sin errores.

Programando el callback:

```
function btncover_Callback(hObject, eventdata, handles) clear
all;clc;
global y f nombre c R
[nombre]=uigetfile('*.wav','cargar archivo');
addpath(ubicacion);[y,f]=audioread(nombre);
c=9.53674e-7; % constante de conversión
R=(c*length(y)); % tamaño
msgbox([num2str(R) 'MB restantes'])
```

Al ser el botón inicial aplican las funciones de borrado completo de variables y pantalla; las variables definidas como globales se crean para que puedan ser utilizadas en otras funciones dentro del mismo GUI.

La variable *y*, corresponde a los datos que conforman el archivo de audio WAV.
La variable *f*, corresponde a la frecuencia de muestreo del archivo WAV.

La variable *c*, es la constante de conversión del terabyte y es usada para calcular en *R* el tamaño real en bytes del archivo, este valor se muestra a través de un mensaje al usuario para que conozca el espacio disponible con el que cuenta para incrustar archivos.

Botones MP3

Los cinco *botones MP3* de la sección 2 de la figura, tienen la función de leer los archivos mp3, uno por botón, hasta un total de cinco. Los mp3 deben cargarse secuencialmente para evitar errores.

Programando el callback:

```
global elemento_1 c R archivo1
[archivo1]=uigetfile('*.mp3','cargararchivo');
addpath(ubicacion);fid1 = fopen(archivo1,'r');
%los archivos se leen y se asiganan a una variable como arreglo 8bit.
elemento_1 = fread(fid1,'uint8=>uint8');%uint8=>uint8 o *uint8 es para la
precision
fclose(fid1);
R=(R-c*length(elemento_1));
msgbox([num2str(R) 'MB restantes'])
```

La función contiene el código general para la lectura de mp3 y cálculo del espacio restante, esta función es la misma para los cinco botones.

Se llaman algunas variables como *c* y *R* definidas previamente y se crean dos variable nuevas que corresponden a los datos del archivo mp3 correspondiente a cada botón.

De tal forma que la variable *archivo1* contiene la información del nombre del mp3 y la variable *elemento_1* contiene los datos de audio del mp3, datos que se almacenan como enteros de 8 bits.

Botón mezclar

El botón *mezclar* ubicado en la sección 3 es donde está la estructura principal del

programa y es la etapa que se encarga de generar el estego-archivo.

Programando el callback:

En esta función se llaman variables creadas previamente en el botón mp3 que serán necesarias más adelante.

```
function btnocultar_Callback(hObject, eventdata, handles)
global elemento_1 elemento_2 elemento_3 elemento_4 elemento_5 y ...
archivo1 archivo2 archivo3 archivo4 archivo5
```

Se hace necesario añadir al path de Matlab a la ruta para evitar errores y poder organizar los archivos de salida en el proceso. La función de este código es obtener la ruta actual y crear la carpeta “Resultados” en caso de que no exista.

```
[~,struc] = fileattrib;
PathCurrent =
struc.Name;
if(exist([PathCurrent'/Resultados'],'file')==0) mkdir([PathCurrent
'/Resultados']);
guardar = strcat(getfield(struc,'Name'),'\Resultados'); else
guardar = strcat(getfield(struc,'Name'),'\Resultados');
end
```

A través del siguiente ciclo se crean los vectores con información de la longitud de cada mp3 y los nombres, la estructura del código es la misma para los cinco archivos mp3, solo se describe para el primero de ellos, obviando que las variables cambian según el elemento al que correspondan.

```
% inicializo el contador de archivos canciones=0;
    switch
    isempty(elemento_1)==false
    case true
    Long_1 = length(elemento_1); % longitud del elemento entero decimal
    cad1=uint8(num2str(Long_1));% Convierte longitud a enteros de 8 bits
    cad1_long=length(cad1); % Asigno el valor de la longitud de cad1
    z1=10-cad1_long; % Cuenta lo que falta para completar 10 bits
    cad1=[zeros(1,z1) cad1]'; % Rellena con zeros restantes hasta 10 bits
    cadnomf1=uint8(archivo1); % Conversión del nombre a enteros de 8 bits
    nz1=150-length(cadnomf1); % Cuenta el sobrante y completa 150 bits
    cadnomf1=[zeros(1,nz1) cadnomf1]'; % Rellena con zeros hasta 150 bits
    cad1=[cad1;cadnomf1]; % Crea un vector con longitud y nombre dentro
    cad1_long=length(cad1); % Recalcula longitud vector
    canciones=canciones+1; % Contador para el proceso de incrustación
    case false % en caso de que el archivo no exista
    Long_1 = 0; % se anula tanto longitud como vector
    cad1_long=0;
end
```

Se crea un vector vacío con la longitud exacta para más adelante llenarlo con la información final en un vector unificado al que llamaremos *Concat_5MP3.*

```
% creo un vector de ceros con las longitudes de todo para evitar errores
Concat_5MP3=uint8(zeros(cad1_long+Long_1+cad2_long+Long_2+...
    cad3_long+Long_3+cad4_long+Long_4+cad5_long+Long_5,1));
```

Se ingresa toda la información de audio de los archivos mp3 en *Concat_5MP3* de tal forma que el orden general sea para cada uno; cabecera, nombre y datos.

```
% se rellena el vector con los valores correspondientes li=0; %
límite inferior
ls=0; % límite superior for
j=1:canciones
lf=(eval(strcat('Long_',num2str(j)))); % longitud del fichero
ls=li+160; % nuevo límite superior
Concat_5MP3(li+1:li+160,1)=eval(strcat('cad',num2str(j)));
Concat_5MP3(ls+1:ls+lf,1)=eval(strcat('elemento_',num2str(j)));
li=160+lf+li; % nuevo límite inferior
end
```

Esta sección del código valida si se cumplió con la condición de no exceder el tamaño máximo permitido para el ingreso de información en los *botones MP3,* una vez verificado comienza con el proceso de incrustación de datos en el WAV; caso contrario muestra un mensaje de error al usuario.

```
% valido total de muestras para continuar
elemento_embed=Concat_5MP3; % Variable con información final de los
mp3 RemSamples = length(y)-length(elemento_embed); % muestras
residuales if ( RemSamples >= 0)
Long_5mp3 = length(elemento_embed);
% comienza el proceso de inserción de los mp3 en el archivo host WAV
elemento_estego = y; % y es la información contenida en el WAV
elemento_estego = ((elemento_estego+1)./2).*65535; % +1 para valores
positivos
% Cuantización de [0,65535] que es el número total de muestras para 16 bit
% Inserción de la señal
for i = 1:Long_5mp3
bit4_1 = bitand(uint8(elemento_embed(i)),uint8(15));
bit4_2 = bitand(bitshift(uint8(elemento_embed(i)),-4),uint8(15));
```

```
% Insertar la primera señal de 4 bits elemento_estego(i,1)
= bitand(uint16(elemento_estego(i,1)),uint16(65520));
elemento_estego(i,1) =
bitor(uint16(elemento_estego(i,1)),uint16(bit4_1));
% Insertar la segunda señal de 4 bits
elemento_estego(i,2) =
bitand(uint16(elemento_estego(i,2)),uint16(65520));
elemento_estego(i,2) =
bitor(uint16(elemento_estego(i,2)),uint16(bit4_2)); end
% Cuantización de [-1,1]
elemento_estego = ((elemento_estego./65535).*2)-1;
% Guarda la señal estego en un archivo WAV
wavwrite(elemento_estego,44100,16,strcat(guardar,'/','Estego_archivo.wa v'));
addpath(guardar);
msgbox('Proceso de inserción completado.');
else
msgbox(char('Falló el proceso.','La señal secreta excede el límite de ', ...
    'longitud de señal de la portadora'),'Error','error')
end
```

Botón borrar

Este botón elimina las variables sin eliminar la carpeta “Resultados” y su contenido, permitiendo realizar el proceso de lectura de archivos nuevamente e informando al usuario sobre esto.

Programando el callback:

```
clear all;
clc;
```

```
msgbox('Debes comenzar de nuevo.','Información','help')
```

Botón Reproducir

La única función de este botón es reproducir el archivo de audio que se eligió como portadora en formato WAV, el archivo reutiliza variables como *y* & *f*, definidas previamente. Para reproducir el archivo se crea un reproductor en la variable *pl*, este archivo se puede reproducir sin ningún problema mientras se realiza el proceso del botón mezclar.

Programando el callback:

```
global y f pl
pl=audioplayer(y,f);
play(pl);
```

Botón Detener

Su función es parar la reproducción del botón reproducir descrito en el punto anterior haciendo uso de la variable global *pl*.

Programando el callback:

```
global pl
stop(pl);
```

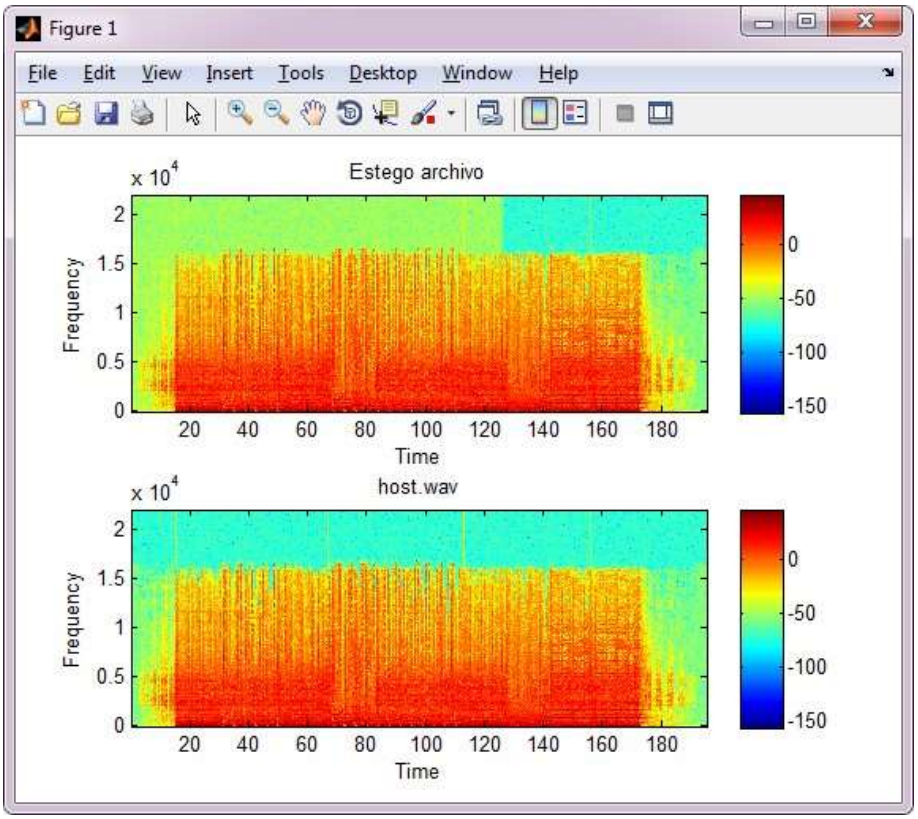
Botón Espectrograma

La función utilizada para comparar el archivo original de la portadora WAV con la generada por el programa con nombre “estego-archivo” en el dominio del tiempo

es specgram, es una función propietaria de Matlab. Se han tomado dos tipos de archivos .wav con la finalidad de que se pueda apreciar visualmente el proceso de incrustación de archivos.

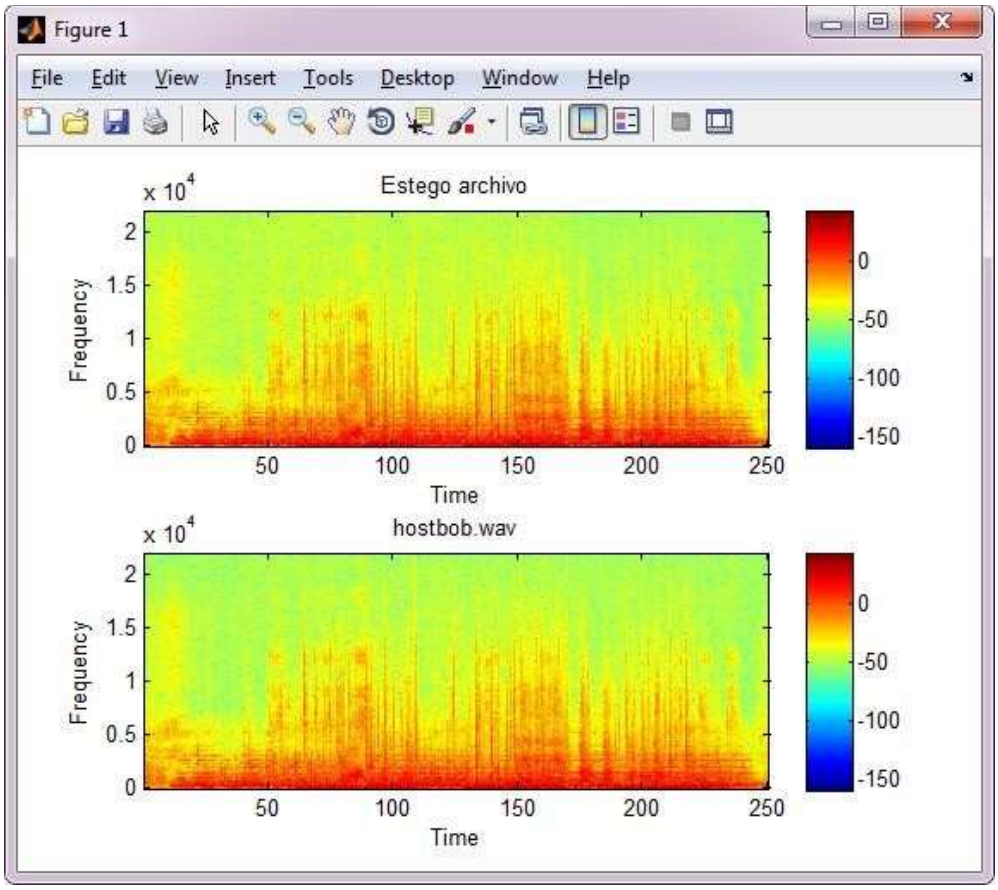
En la figura 37 se puede observar claramente el espacio que ocupan los archivos incrustados en la portadora de color verde y de color celeste el espacio sobrante, esto se debe a que al tomar como base un archivo mp3 la cantidad de frecuencias contenidas en el es menor, pues el algoritmo mp3 elimina las frecuencias superiores a los 16 KHz, por eso al escalarlo se ve aquel espacio celeste que representa la ausencia de información o frecuencias por encima de la frecuencia de corte.

Figura 37. Espectrograma mp3 escalado



En la figura 38 se ha utilizado un archivo .wav de onda completa, como podemos ver ahora no se puede diferenciar a simple vista si el estego-archivo del original, esto es lo que hace que el método sea efectivo.

Figura 38. Espectrograma WAV



Programando el callback:

```
function btnespectro_Callback(hObject, eventdata, handles)
% Lee el estegano-archivo
global nombre
[d,sr] = audioread('Estego_archivo.wav');
% Plotear el espectrograma
figure('Color',[1 1 1])
```

```
subplot(211)
specgram(d(:,1),1024,sr);
title('Estego-archivo') colorbar
% Lee la cubierta
[d,sr] = audioread(nombre);
% Plotea
subplot(21
2)
specgram(d(:,1),1024,sr);
title(nombre)
colorbar
```

Botón ayuda

Abre el archivo de ayuda con un visor externo previamente instalado en el sistema.

Programando el callback:

```
function botonayuda_Callback(hObject, eventdata, handles)
winopen('Ayuda emisor.pdf');
```

3.2.3 Interfaz del Receptor. Esta interfaz fue diseñada por aparte para permitir un acceso rápido al usuario final y también porque no depende de las variables creadas en el emisor.

Figura 39. Interfaz del receptor

Botón abrir estego-archivo

Su función es leer los datos de audio en el estego-archivo y cargarlos en Matlab añadiendo también la ruta de donde se carga el archivo.

Programando el callback:

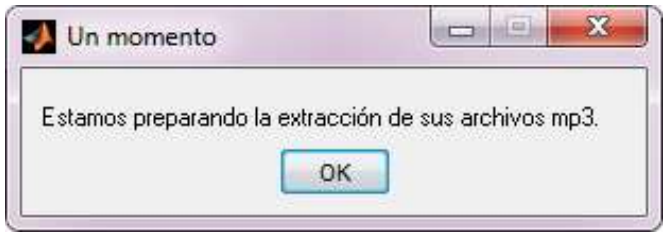
```
function Abrir_Callback(hObject, eventdata, handles) clear
all;clc;
global elemento_estego;
[archivo, ubicacion]=uigetfile('*.wav','cargar archivo');
addpath(ubicacion);[elemento_estego]=audioread(archivo);
end
```

Botón procesar

En este botón se encuentra la función principal, la cual se encarga de recuperar la información embebida en el estego-archivo y generar los archivos finales dentro de la carpeta “Resultados”. Se debe tener en cuenta que si a la entrada del emisor se ingresan dos archivos con el mismo nombre, a la salida se reescribirá el archivo con la información del último disponible.

Esta es la rutina de extracción, en la cual se regenera el vector unificado embebido en el estego-archivo anteriormente, mientras se realiza el proceso se notificará al usuario que espere.

Figura 40. Ventana de alerta del botón procesar



Programando el callback:

```
global elemento_estego
[~,struc] = fileattrib;
guardar = strcat(getfield(struc,'Name'),'\Resultados'); msgbox('Estamos
preparando la extracción de sus archivos mp3.','Un momento')
% se revierte la cuantización [-1,1] elemento_estego
= ((elemento_estego+1)./2).*65535;
% Se cuantiza la señal de cubierta de [0,65535].
Long_5MP3 = length(elemento_estego);
Extraccion = zeros(Long_5MP3,1);
```

```
% Rutina de extracción
for i = 1:Long_5MP3
% Extracción de la primer señal de 4 bits
s4_1_ext = bitand(uint16(elemento_estego(i,1)),uint16(15));
% Extracción de la segunda señal de 4 bits
s4_2_ext = bitand(uint16(elemento_estego(i,2)),uint16(15));

% Combinacion de las dos señales de 4 bits para generar una de 8 bits
Extraccion(i,1) = bitor(uint8(0),uint8(s4_2_ext)); Extraccion(i,1)=...
bitor(bitshift(uint8(Extraccion(i,1)),4),uint8(s4_1_ext));
end
```

A continuación se crean los vectores con la información de las longitudes y nombres para cada mp3 con la información obtenida en el vector unificado extraído en el paso anterior.

```
li=0; ls=0; c=[]; names=[]; nacum=[]; inf=[]; sup=[]; j=0;
% evalua si el valor es numérico para proseguir
whileisnan(str2double(char(Extraccion(li+1:li+10,1)')))==false
lf=str2double(char(Extraccion(li+1:li+10,1)'));
names=(Extraccion(li+11:li+160,1));
nacum=[nacum; names];
ls=li+160;
c=[c lf]; inf=[inf
ls+1]; sup=[sup
ls+lf];
li=160+lf+li;
j=j+1;
end
```

```
for J=1:j
eval(sprintf('Signal_%d_sep = 0', J)) end
clc;
```

Debido a que las salidas difieren según la cantidad de entradas en el emisor, se crean unas variables que se podrían considerar como dinámicas, pues se crean solo cuando se hacen necesarias para ahorrar espacio en memoria.

Se descomprime y recupera cada archivo mp3, se describe el código general para cada archivo presente obviando que para cada archivo sus variables son secuenciales, el único valor que varía es el de la variable *nacum*, como ya se sabe allí es donde se acumulan los nombres cada 150 bits y al final se notifica al usuario la cantidad de archivos extraídos.

```
switch exist ('Signal_1_sep', 'var')==true case
      true Signal_1_sep=Extraccion(inf(1):sup(1));
        nombre=nacum(1:150)';,nombre(nombre==0)=[];
      fid1=fopen(strcat(guardar,'/',char(nombre)),'w');
      fwrite(fid1,Signal_1_sep,'uint8');fclose(fid1);
end
msgbox(sprintf('Proceso de extracción completado.\nExtraído(s) %d
archivo(s)',j),'Resultado')
```

Figura 41. Ventana Resultado

Con esto se concluye la recuperación de los archivos los cuales son recopilados en una carpeta nombrada “Resultados”, estos pueden ser reproducidos nuevamente y al igual que los originales también conserva todos los metadatos además del nombre.

4. CONCLUSIONES Y OBSERVACIONES

Tras las pruebas realizadas se observó que el algoritmo utilizado es vulnerable a cambios en las propiedades principales del archivo, como la amplitud, frecuencia o fase, ocasionando que al archivo final no se le podrá recuperar la información.

Una de las ventajas del método aplicado, es que no solo permite el envío de archivos de audio, sino que también permite enmascarar otro tipo de archivos cambiando las extensiones a .mp3, en este sentido la versatilidad del algoritmo propuesto facilita el ocultamiento de todo tipo de archivos haciéndolos pasar por alto a los ojos de un tercero mal intencionado.

Durante el desarrollo de la primera versión del algoritmo, se observó que los datos ocultos se distribuían de manera uniforme dentro del archivo contenedor lo que hacía fácil la detección de la información oculta con un simple análisis del espectrograma de la señal (ver figura 37), esto se solucionó escalando la información en todo el espectro del archivo contenedor modificando la rutina de ocultamiento en la fase de creación del vector de magnitud del archivo de información.

El método permite que el estego-objeto mantenga su tamaño en bits invariable lo que impide a simple vista detectar anomalías en el archivo original, esto es así gracias a que la información se oculta reemplazando un bit de datos por un bit (el menos significativo) dentro del fichero contenedor.

Como observaciones al presente trabajo y en relación a trabajos futuros debe considerarse que:

-La aplicación se diseñó para esconder una cantidad limitada de archivos (cinco en total), la capacidad de ocultamiento en bits está determinada por el tamaño del carrier, un archivo contenedor de mayor tamaño permitirá esconder una mayor cantidad de información. Para esto debe modificarse la aplicación y permitir así un número mayor de archivos ocultos.

-Aunque la encriptación no fue el tema abordado en este proyecto, como valor agregado es posible insertarle algún tipo de encriptación o clave de seguridad al archivo para hacerlo aún más robusto antes de recuperar la información en el receptor.

REFERENCIAS BIBLIOGRÁFICAS

Amazon. (2013). *www.amazon.com*. Obtenido de http://www.amazon.com/The-Code-Breakers-Story-Secret-Writing/dp/B000M1IL8K

Bonhams. (10 de Octubre de 2011). *http://www.bonhams.com/*. Obtenido de http://www.bonhams.com/auctions/19421/lot/1019/

Cano, D. G. (Diciembre de 2004). *http://e-archivo.uc3m.es/*. Obtenido de http://e-archivo.uc3m.es/bitstream/handle/10016/7119/PFC_David_Garcia_Cano_2004_201033204919.pdf?sequence=1

Castro Lechtaler, A. R., & Fusario, R. J. (2013). *Comunicaciones, Una introducción a las redes digitales de transmisión de datos y señales isócronas.* Buenos Aires: Alfaomega.

Clemente, A. N. (Junio de 2015). *http://achtung.es/*. Obtenido de http://achtung.es/papers/stegano.pdf

Death-Master. (2004). *http://www.bitgamia.com/*. Obtenido de http://www.bitgamia.com/wp-content/uploads/2011/09/Introducci%C3%B3n-a-la-esteganografia-por-Deathmaster.pdf

Fabian. (13 de Junio de 2006). *http://www.petitcolas.net/*. Obtenido de http://www.petitcolas.net/steganography/mp3stego/

Galcerán, G. M. (2013). *Universidad Politecnica de Cataluya.* Obtenido de http://upcommons.upc.edu/pfc/handle/2099.1/17358

Gamez, B. E. (7 de MAYO de 2009). *Instituto Politecnico Nacional* . Obtenido de http://tesis.ipn.mx/handle/123456789/4032

Jayaram, R. (Agosto de 2011). *http://www.airccse.org/*. Obtenido de http://www.airccse.org/journal/jma/3311ijma08.pdf

Latham, A. (Agosto de 2009). *http://linux01.gwdg.de/*. Obtenido de http://linux01.gwdg.de/~alatham/stego.html

Marchena, J. M. (18 de abril de 2013). *http://interfazgraficamatlab.blogspot.com/*. Obtenido de http://interfazgraficamatlab.blogspot.com/2013/04/interfaz.html

Mifsud, E. (26 de Mayo de 2012). *http://recursostic.educacion.es/*. Obtenido de http://recursostic.educacion.es/observatorio/web/ca/software/software-general/1040-introduccion-a-la-seguridad-informatica?start=1

Morales, R. C., Delgado Guitierrez , G., & Vazquez Medina, R. (s.f.). Esteganografia en audio por el criterio de la transformada discreta del coseno. *Instituto Politecnico Nacional - Mexico*, 3.

Muñoz, A. (02 de 01 de 2014). *http://www.criptored.upm.es/*. Obtenido de http://www.criptored.upm.es/crypt4you/temas/privacidad-proteccion/leccion7/leccion7.html%20Dr.%20Alfonso%20Mu%C3%B1oz%20-%2002/01/2014

Obregozo, I. S. (2011). *E- Prints Complutense*. Obtenido de http://eprints.ucm.es/13512/

Petitcolas, F. (2014). *http://www.petitcolas.net/*. Obtenido de http://www.petitcolas.net/kerckhoffs/la_cryptographie_militaire_i.htm

Rincón Rivera, D. (2000). MP3, Sonido digital al alcance de todos. *Univeridad Politecnica de Cataluña*, 9.

Spencer, E. (2004). *Ilookforensics.org.* Recuperado el 12 de febrero de 2015, de www.ilookforensics.org

Tomasi, W. (2003). *Sistemas de comunicaciones electronicas.* Mexico: Pearson Education.

Torres. (28 de Octubre de 2011). *http://es.scribd.com/*. Obtenido de http://es.scribd.com/doc/70635986/Esteganografia-Desde-Cero

Vico, J. D. (Septiembre de 2010). *http://oa.upm.es/*. Obtenido de http://oa.upm.es/5353/2/TESIS_MASTER_JESUS_DIAZ_VICO.pdf

Whethamstede, J., & Blois, L. d. (2011). *http://es.knowledger.de/*. Obtenido de http://es.knowledger.de/0114787/JohannesTrithemius