Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

“Seguridad por medio de Esteganografía”

TESIS

Para Obtener el Título de:

Licenciado en Ingeniería en Ciencias de la Computación

Presenta:

Luis Gerardo Munive Morales

Asesora:

Dra. Bárbara Emma Sánchez Rinza

Puebla, Pue. Mayo 2016

ÍNDICE

INTRODUCCIÓN.

A través de la historia las personas han ocultado información a través de distintos métodos, en la actualidad las cosas no han cambiado, simplemente han evolucionado, principalmente en las comunicaciones digitales, donde se han implementado nuevas técnicas para evitar, o minimizar, ataques de revelación, supresión o alteración de la información intercambiada entre diferentes actores.

Un elemento fundamental en este proceso ha consistido en apoyar la seguridad de los procedimientos implementados en el uso de la criptografía. Esta ciencia tiene entre sus virtudes facilitar servicios de autenticidad, confidencialidad e integridad de la información almacenada o intercambiada. Su problema fundamental recae en que las comunicaciones cifradas pueden ser detectadas, es decir un tercero puede tener conocimiento de un intercambio de información aunque no por ello dicha información pueda ser revelada, sin embargo puede tomar acciones que tengan como consecuencia impedir dicha comunicación. Para complementar a esta ciencia surge el interés de uso de la esteganografía.

La esteganografía estudia el conjunto de técnicas, cuyo fin es esconder información de manera que la existencia del mensaje escondido pase desapercibida para un observador casual y/o un intruso que desee interferir con la comunicación de dicho mensaje e inclusive alterar el mensaje.

CAPÍTULO I

ESTEGANOGRAFÍA

1.1 INTRODUCCIÓN Y DEFINICIÓN

La palabra esteganografía viene del griego *esteganos* (cubierto o secreto) y *grafos* (escritura o pintura). Es la ciencia de esconder información de manera que la existencia del mensaje escondido pase desapercibida, ésta nos permite usar una gran variedad de contenedores y algoritmos para poder ocultar la información que queramos enviar, con esto logramos que sólo sea recuperada por un usuario legítimo que conozca un determinado algoritmo de extracción de la misma.

Es necesario mencionar la diferencia que existe entre esteganografía y criptología, esto debido a que en un gran número de ocasiones ambos conceptos son combinados para lograr una mayor protección de la información, como consecuencia suelen ser confundidos

Un ejemplo básico de esteganografía se muestra en la Figura 1.1.1, donde observamos una tira de papel que al ser enredada a un objeto con forma y tamaño específico nos muestra el mensaje oculto.

Figura 1.1.1 Ejemplo de Esteganografía.

Por otro lado la criptografía se encarga de cifrar ó codificar información de tal manera que sea ininteligible, es por esto que se combina con la esteganografía, para así tener la información encriptada y oculta. En la figura 1.1.2 encontramos un ejemplo de un “cifrado césar”, el cual consta en escribir el mensaje con un alfabeto formado por las letras del

alfabeto latino pero siendo rotadas N posiciones a la derecha, en nuestro ejemplo el alfabeto tiene un desplazamiento de 13 posiciones y se ocupa para encriptar la palabra “Hola”.

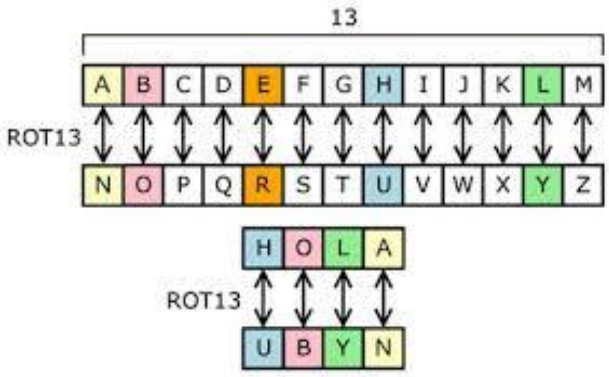

Figura 1.1.2 Cifrado César[1] con rotación de 13 posiciones.

Una vez que se ha comprendido la diferencia entre estas ciencias identificaremos los distintos actores implicados en el campo de la Esteganografía:

a) *Objeto contenedor:* Se trata de la entidad que se emplea para portar el mensaje oculto. Acudiendo al ejemplo antes mencionado, el objeto contenedor es la tira de papel.

b) *Estego-objeto:* Se trata del objeto contenedor más el mensaje encubierto. Siguiendo con el ejemplo, se trata de la tira de papel con el mensaje escrito pero sin estar enredado, es decir con el mensaje oculto.

c) *Adversario:* Son todos aquellos entes a los que se les oculta la información. En el ejemplo de la prisión, se trata del guardia que entrega los mensajes a uno y otro prisionero. Este adversario puede ser pasivo o activo. Un adversario pasivo sospecha que se puede estar produciendo una comunicación encubierta y trata de descubrir el algoritmo que extrae del estego-objeto, pero no trata de modificar dicho objeto. Un adversario activo, además de tratar de hallar el algoritmo de comunicación encubierta, modifica el estego-objeto con el fin de corromper cualquier intento de mensajería subliminal.

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

[1]Cifrado César. Cifrado por sustitución en el que una letra en el texto original es reemplazada por otra letra que se encuentra un número fijo de posiciones más adelante en el alfabeto. Por ejemplo, con un desplazamiento de 3, la A sería sustituida por la D (situada 3 lugares a la derecha de la A) , la B sería reemplazada por la E, etc. Este método debe su nombre a Julio César, que lo usaba para comunicarse con sus generales.

Existe una gran variedad de métodos que nos permiten ocultar información (algoritmos esteganográficos), los cuales podemos englobar en 2 grandes categorías:

- *Enmascaramiento y filtrado.* En este caso la información se oculta dentro de una imagen digital empleando marcas de agua que incluyen información, como el derecho de autor, la propiedad o licencias. El objetivo es diferente de la esteganografía tradicional (básicamente comunicación encubierta), ya que es añadir un atributo a la imagen que actúa como cubierta. De este modo se amplía la cantidad de información presentada.
- *Algoritmos y transformaciones.* Esta técnica oculta datos basados en funciones matemáticas que se utilizan a menudo en algoritmos de la compresión de datos. La idea de este método es ocultar el mensaje en los bits de datos menos importantes.

1.2. ESTRUCTURA DE LOS ARCHIVOS.

Como se ha mencionado con anterioridad los archivos multimedia son de gran utilidad como objetos contenedores, antes de analizar los diferentes algoritmos esteganográficos es necesario comprender su estructura para así poder entender cómo se ocultaría un mensaje de texto dentro de estos archivos.

1.2.1. IMAGEN.

Una imagen se encuentra conformada por píxeles (pixel: picture element), la combinación de dichos píxeles es lo que le da color y forma a la imagen terminada, cuando le hacemos mucho zoom muy grande a una imagen, tal y como se muestra en la figura 1.2.1.1., vamos a encontrarnos con que todo está conformado por pequeños cuadritos y son estos a los que nosotros llamamos píxeles, pues son la unidad mínima de representación para una imagen.

Figura 1.2.1.1. Visualización de los pixeles de una imagen.

Un píxel está conformado por una tripleta de datos conocidos como RGB (Red**,** Green**,** Blue), esta tripleta de datos es la que conforman el color final del píxel, esto es: los tres colores se mezclan en diferentes proporciones generando distintos colores.

Como podemos observar en la Figura 1.2.1.2, distintas combinaciones en RGB generan distintos tipos de colores, el rango de cambió en los valores RGB es de 0 a 255 para cada valor.

R → 255	R → 255	R → 16
G → 0	G → 100	G → 155
B → 0	B → 0	B → 51

Figura 1.2.1.2 Ejemplo de colores RGB

1.2.2 AUDIO.

El audio digital se diferencia del sonido analógico tradicional en que es una señal discreta en lugar de una continua. Esta señal discreta es creada llevando a cabo un proceso de muestreo de una señal analógica continua. La frecuencia de muestro es diferente según los propósitos, es decir la frecuencia de muestreo estándar para un CD de audio digital se encuentra aproximadamente en un rango de 44kHz [11]. (En la figura 1.2.2.1 muestra dicho proceso), sin embargo en dicha figura se ha exagerado la naturaleza discreta de la señal digital, por otro lado las frecuencias de muestreo permiten que las diferencias existentes entre al audio digital y una señal analógica original sean mínimas.

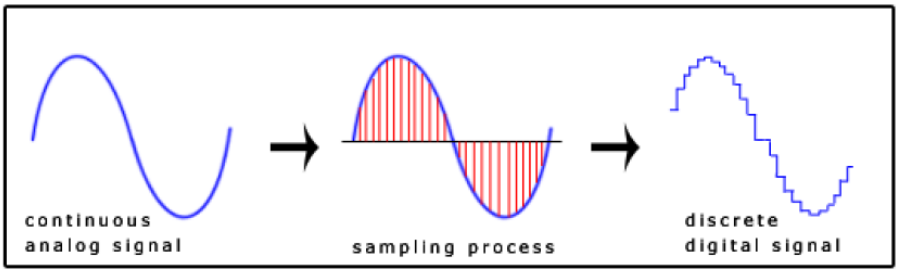

Figura 1.2.2.1. Muestreo de señal analógica a audio digital

La representación de la información en los archivos de sonido se realiza generalmente mediante el registro consecutivo de las muestras que componen el sonido. Al igual que ocurre con los archivos de imagen, el sonido es representado siempre en forma de bits, y cada vez con una calidad mayor, de forma que se podrá manipular los bits menos significativos de los archivos para ocultar información en ellos.

Antes de comprender como se puede ocultar información en archivos de sonido, comprendamos de mejor manera como una señal acústica es digitalizada.

Previo a la digitalización el sonido debe de ser transformado de su forma original (vibraciones que se propagan a través del aire), a un formato manejable, como lo es una corriente eléctrica, este proceso es realizado por los micrófonos.

La digitalización está compuesta por 3 procesos:

1. *Muestreo.* Dada una señal analógica se procede a tomar muestras de la misma con una determinada frecuencia. Cuanto mayor sea esta frecuencia mayor será la fidelidad, un ejemplo de este muestreo se observa en la Figura 1.2.2.2.

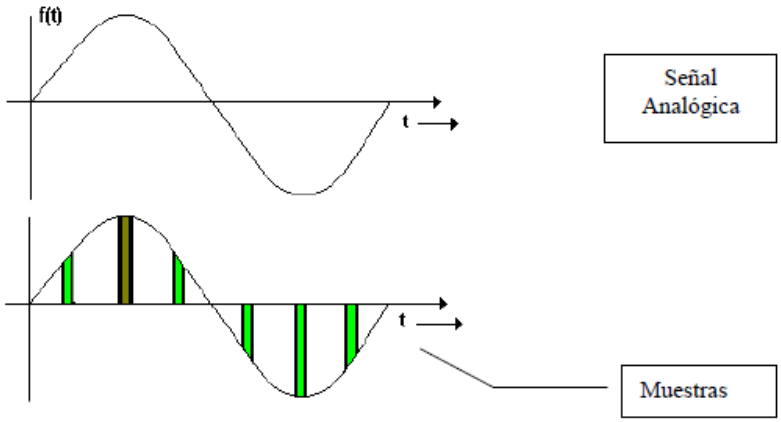

Figura 1.2.2.2. Muestreo de una señal.

2. *Cuantificación.* Una vez obtenida la muestra de estudia su nivel, y se le asigna el valor más aproximado que se le encuentre en un abanico de valores previamente establecido, cuanto más valores o niveles tenga este abanico, es decir, cuantos mas bits destinemos a la cuantificación, mejor será la calidad del sonido. Por ejemplo una cuantificación de 16 bits ofrecerá un tamaño de muestreo de 65536 valores (2^{16}).

3. *Conversión a binario.* Ya con el valor de la muestra obtenido se le transforma en números binarios.

En el sonido ocurre una cosa muy curiosa, nos da prácticamente igual que la calidad sea alta o baja, pues en calidades altas la cantidad de bits usados para representar el sonido es muy alta, y un cambio ínfimo no influye en el resultado; mientras que en los de calidad baja, aunque la modificación sea más llamativa, al tratarse de un sonido de baja calidad, cualquier modificación puede pasar inadvertida como ruido de fondo.

En la actualidad existe un gran número de formato de audio, los más conocidos son el formato “mp3” y “WAV”, el formato WAV es un formato de audio sin compresión, lo cual lo que lo convierte en un “objeto contenedor” ideal.

Waveform Audio File Format. Se trata del formato de sonido desarrollado por Microsoft para su sistema operativo Windows. Se compone de una cabecera de 43 bytes y un conjunto arbitrario de bytes que contienen las muestras una tras otra, sin ningún tipo de compresión y con cuantificación uniforme. Al ser un formato realmente sencillo, es muy útil para realizar tratamiento digital de sonido. Su principal ventaja es la sencillez del formato, y su mayor inconveniente la cantidad de espacio requerido (una muestra de 10 segundos en calidad de CD -PCM, 44 KHz, 16 bit, estéreo ocupa 1,6 Mb). Cada archivo WAV al momento de ser leído por un reproductor de audio, es identificado por los distintos miembros que conforman la cabecera. En la Figura1.2.2.2 se muestra el mapa de la Cabecera en archivos WAV de una manera más detallada, donde encontraremos los campos que compone a las cabeceras así como el tamaño en bytes.

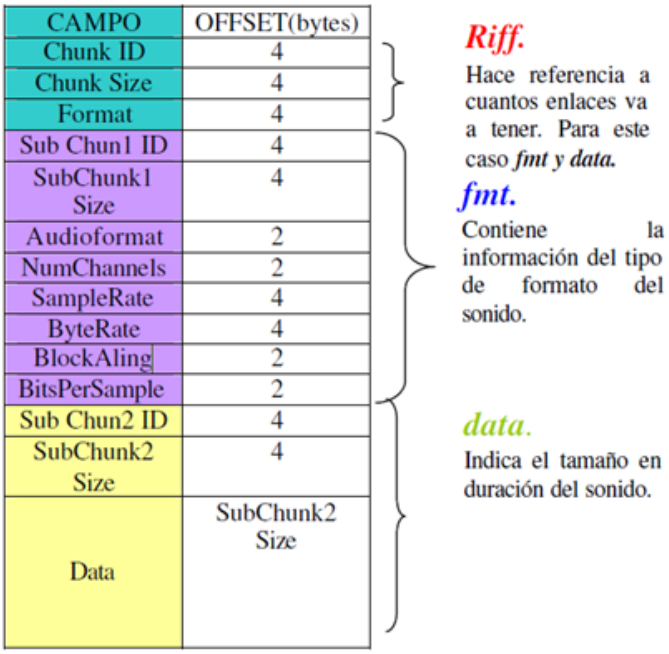

Figura 1.2.2.2. Mapa de la Cabecera en archivos WAV.

Enfocándonos en las comunicaciones digitales y en la computación en general la esteganografía digital incluye codificación al interior de una capa de transporte, tal como un archivo de documento, archivo de imagen, programa o protocolo, es por ello que los archivos multimedia son ideales para la esteganografía debido a su gran tamaño, al mismo tiempo cuentan con diferentes algoritmos esteganográficos los cuales en ocasiones solo pueden ser aplicados a archivos multimedia específicos y otros se pueden aplicar de manera más general.

1.3 ALGORITMOS ESTEGANOGRÁFICOS PARA IMÁGENES.

1.3.1 LEAST SIGNIFICANT BIT (LSB).

El algoritmo LSB (sustitución del bit menos significativo), este algoritmo tiene algunas variantes, las cuales analizaremos a continuación:

1.3.1.1SIMPLE LSB.

Consiste en ocultar x bits del mensaje subliminal en cada pixel de la imagen, modificando los x bits menos significativos del mensaje original, esto es, para poder aplicar el algoritmo primero es necesario convertir el objeto contenedor y el mensaje a binario, este proceso se explica a continuación.

En cada pixel de la imagen obtenemos una tripleta RGB compuesta por números enteros desde el cero (0) hasta el (255), y como cada número tiene su propia representación en binario, lo que haremos será convertir dicha tripleta en su equivalente en binario, por ejemplo, el pixel conformado por (148, 28, 202) equivale en binario a (10010100, 00011100, 11001010).

Posteriormente lo que se hace es editar el bit menos significativo, (aquel que se encuentre de último a la derecha), como se puede observar en la Figura 1.3.1.1.1 se han resaltado en color rojo , debajo en la columna LSB se han alterado los bits (en rojo) pero los demás siguen intactos y el resultado de la tripleta RGB sufre algunos cambios, pero son mínimos. Al observar cuidadosamente ambos colores es muy improbable que se encuentre tipo de diferencia visual, aunque en realidad hubo un cambio, después de alterar el bit menos significativo, la tripleta RGB es distinta de la que teníamos al inicio, pero el color aparentemente es el mismo.

	Binario	LSB	
R -> 148	10010100	10010101	R -> 149
G -> 28	00011100	00011101	G -> 29
B -> 202	11001010	11001011	B -> 203

Figura 1.3.1.1.1 Aplicación del algoritmo LSB en un pixel

Ahora es momento de ocultar un mensaje, por ejemplo, si deseamos esconder la palabra “Hacking” tenemos que recordar que cada letra del mansaje (Carácter) se puede representar

por un Byte siendo así la “H“= 01001000 entonces si tenemos 3 pixeles podemos esconder esa secuencia mediante LSB tal y como se muestra en la Figura 1.3.1.1.2

	Binario	“H”	LSB		
	R -> 148	10010100	0	10010100	R -> 148	
	G -> 28	00011100	1	00011101	G -> 29	
	B -> 202	11001010	0	11001010	B -> 202	
	R -> 178	10110010	0	10110010	R -> 178	
	G -> 71	01000111	1	01000111	G -> 71	
	B -> 22	00010110	0	00010110	B -> 22	
	R -> 87	01010111	0	01010110	R -> 86	
	G -> 219	11011011	0	11011010	G -> 218	
	B -> 100	01100100	-	01100100	B -> 100	

Figura 1.3.1.1.2 Ocultación de la “H”.

De esta forma podemos esconder no solo texto sino también todo tipo de información, pues todo es representable en valores binarios; la manera de recuperar la información es solo recibir la imagen alterada y empezar a leer los bits menos significativos, pues cada 8 bits tenemos la representación de un carácter.

1.3.1.2 OPTIMAL LSB.

Similar al anterior, pero con el objetivo de no afectar tanto la calidad de la imagen, esto se logra de la siguiente manera.

Después de ocultar “x” bits, el bit (x+1)-ésimo menos significativo es modificado de forma que el valor final del pixel sea más cercano al valor original. Es decir, si el pixel original al que llamaremos “p0” tiene un valor (en binario) de 11001001_2 que equivale a 20110, y queremos ocultar 111_2, el valor final del pixel, “pf”, sería 11001111_2 equivalente a 207_{10}, pero si cambiamos el cuarto bit menos significativo, el valor final sería 11000111_2 igual a 199_{10}, .Por lo tanto, el valor absoluto de la distorsión se reduce de 6 a 2 tal y como se muestra en la Figura 1.3.1.2.1, sin afectar el mensaje subliminal y alterando lo menor posible la imagen.

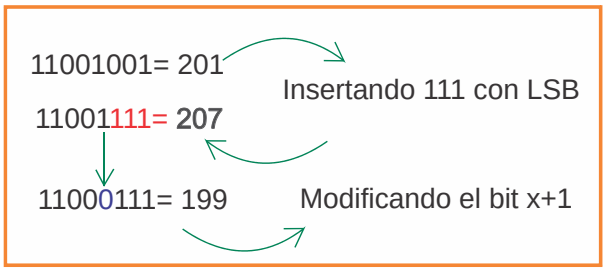

Figura 1.3.1.2.1 Optimal LSB

1.3.2 PIXEL-VALUE DIFFERENCING (PVD).

Este algoritmo permite variar la capacidad de cada pixel en función de su predisposición a ocultar información de forma imperceptible. Los pixeles que forman las fronteras de la imagen presentan más tolerancia a la distorsión que los pixeles de zonas interiores. De esta forma, la imagen se divide en bloques disjuntos, teniendo cada bloque pixeles consecutivos en la imagen, este método se utiliza con más frecuencia en imágenes a escala de grises.

El método PVD nos propone lo siguiente:

- En el proceso de incorporación de un mensaje secreto, una imagen de la cubierta se divide en bloques que no se sobreponen de dos píxeles consecutivos.
- Un valor de diferencia se calcula a partir de estos valores de los dos píxeles en cada bloque. Todos los posibles valores de diferencia se clasifican en una serie de rangos.
- El valor de la diferencia calculada entonces reemplazado por un nuevo valor para incrustar el valor de una sub-secuencia del mensaje secreto.
- El número de bits que puede ser incrustado en un par de píxeles es decidido de la siguiente manera: se debe calcular el valor de la diferencia más grande entre los otros tres y / o cuatro píxeles cercanos al píxel objetivo.

Para poder comprender mejor el método lo explicaremos con un pequeño ejemplo, supongamos que tenemos un bloque de dos pixel muestra que son (90; 110). El valor de la diferencia es 20, que es en el rango de 16 a 32. La anchura de la gama es 16 = 24, lo que significa que un valor de diferencia en el intervalo se puede utilizar para incrustar cuatro bits de datos secretos. Supongamos que los cuatro bits principales de los datos secretos son 1100. El valor de este flujo de bits es 12. Se añade al valor límite inferior 16 de la gama para producir el nuevo valor de la diferencia 28. Por último, los valores (86; 114) se calculan para uso como los valores de gris en la imagen. Tenga en cuenta que 114-86 = 28. Este proceso lo podemos observar en la Figura 1.3.2.1

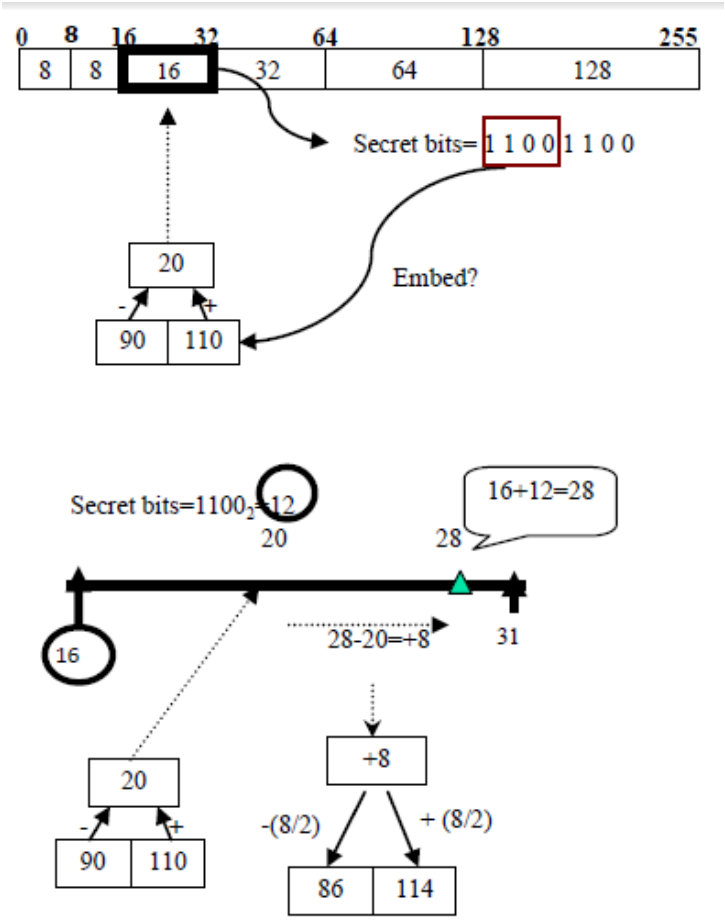

Figura 1.3.2.1 Método PVD.

1.3.3 MBNS (MULTIPLE-BASED NOTATIONAL SYSTEM)

Con un concepto similar al método PVD, este método se basa en el Sistema de Visión Humana para determinar la capacidad subliminal de cada pixel de la imagen original. Esta capacidad aumentará si la variación local de los valores de los pixeles de la zona a la que pertenece el pixel es alta. Dicha variación se calcula teniendo en cuenta tres pixeles adyacentes al pixel en cuestión tal y como se muestra en la Figura 1.3.3.1, por lo que la estimación de la capacidad final del pixel es más acertada que en el método anterior, que sólo tiene en cuenta un pixel adyacente, es decir si el pixel se encuentra en un borde, ó en un zona demasiado obscura es ideal para ocultar información, de lo contrario no es recomendable usarlo para ocultar información.

g(x-1,y-1) top left pixel	g(x-1,y) top pixel
g(x,y-1) left pixel	g(x,y) target pixel

Figura 1.3.3.1 Pixel objetivo para ocultar información y pixeles vecinos

El algoritmo MBSN es el siguiente:

1. Asumamos t_i para i = 1,2,…, L es una secuencia binaria de datos secreta (parte del mensaje a ocultar) cuya longitud es l. Divida L bits de datos secretos en n segmentos con la longitud de cada segmento siendo L / N bits. Convertir cada segmento en su valor decimal x_i, i = 1, 2, ..., n.

2. Supongamos que P(i, j) es un pixel en el cual se va a ocultar información. Entonces, la carga útil de P (i, j) se determina mediante los tres píxeles que lo rodean, es decir, P (i - 1, j), P (i - 1, j - 1) y P (i, j - 1). Calcula la variación δ (i, j) de P (i, j) con P (i-1, j), P (i - 1, j - 1) y P (i, j - 1).

3. Determinar la base de P (i, j) de acuerdo con su variación local δ (i, j). Deje que b (i, j) sea la base de P (i, j). La cual se calcula por:

$$b(i, j) = \min\left(\left| \frac{\delta(i,j)}{\Delta} \right|, 16 \right)$$

   La variable Δ es una constante y controla la capacidad de ocultación de cada pixel. Mientras más pequeño sea Δ tendremos una carga útil mayor, y viceversa. Si b (i, j) ≤ 1, entonces no se ocultará información en P (i, j); es decir, se omitirá este píxel. De lo contrario, se calcula el dígito correspondiente en MBNS para ocultar la información en P (i, j).

$$r(i, j) = P(i, j) \bmod b(i, j)$$

4. Se oculta la información s (i, j) en el pixel P (i, j) usando el algoritmo siguiente:

   Si r (i, j) == s (i, j)
   La tarea de ocultar s (i, j) en P (i, j) es cumplida.
   Si no
   Se modifica P (i, j) hasta que el residuo r (i, j) sea igual a s (i, j).

   Después de este paso, si q ≠ 0, entonces x_i debe ser actualizada por x_i = q, y leer el segundo pixel P (i, j) = P (i+1, j) y para poder ocultar la información se repite el proceso desde el paso 2 hasta el paso 4, Si q = 0 se significa que el primer segmento de la información se ha ocultado. En ese caso podemos leer el siguiente segmento

de la información a ocultar y repetir los pasos del 2 al 6 hasta que se termine de leer toda la información a ocultar.

En cualquier caso, aunque los métodos PVD y MBNS producen resultados finales en los que la información subliminal pasa más desapercibida puesto que mantienen mejor las características de la imagen original, también tienen una capacidad subliminal menor que los métodos Simple LSB y Optimal LSB, que, a cambio de una menor calidad final de imagen (arriesgando por tanto el objetivo primordial de la esteganografía, que es pasar desapercibida), ofrecen mayor capacidad de ocultación.

Paridad de bloques

Otra forma de sustitución se basa en dividir el objeto portador (imagen o audio) en bloques o segmentos de un determinado tamaño. Estableciendo una ordenación arbitraria de los mismos, se calculará la paridad de cada bloque en dicho orden y, si la paridad del bloque b_i coincide con el bit i-ésimo del mensaje subliminal a ocultar, no se hará nada; si no coincide, se modificará el bit menos significativo de algún elemento del bloque. El receptor únicamente tendrá que calcular la paridad de los bloques, obviamente usando la misma ordenación que el emisor, esto nos da como resultado un punto de partida para otros algoritmos esteganográficos.

1.4.1 PHASE CODING METHOD (MÉTODO DE CODIFICACIÓN DE FASE)

El método de codificación de fase funciona mediante la sustitución de la fase de un segmento de audio inicial con una fase de referencia que representa los datos. La fase de segmentos subsecuentes se ajusta a fin de preservar la fase relativa entre los segmentos.

Este método, cuando se puede utilizar, es uno de los métodos de codificación más eficaces en términos de la relación señal a ruido percibido. Cuando la relación de fase entre cada componente de frecuencia se cambia dramáticamente, se producirá dispersión de fase notable. Sin embargo, siempre y cuando la modificación de la fase es suficientemente pequeño (suficientemente pequeño depende del observador; profesionales en la emisión de radio puede detectar modificaciones que están imperceptible para un observador promedio), una codificación inaudible se puede lograr. La fase de codificación se basa en el hecho de que los componentes de la fase de sonido no son tan perceptibles para el oído humano como un ruido es. En lugar de la introducción de perturbaciones, la técnica codifica los bits de mensaje como cambios de fase en el espectro de fase de una señal digital, el logro de una codificación inaudible en términos de relación de ruido percibido.

El algoritmo se explica a continuación:

1. Dividir una señal de sonido original en segmentos más pequeños de tal manera que las longitudes son del mismo tamaño que el tamaño del mensaje que va a codificarse, en la Figura 1.4.1.1 se muestra un ejemplo de esta división.

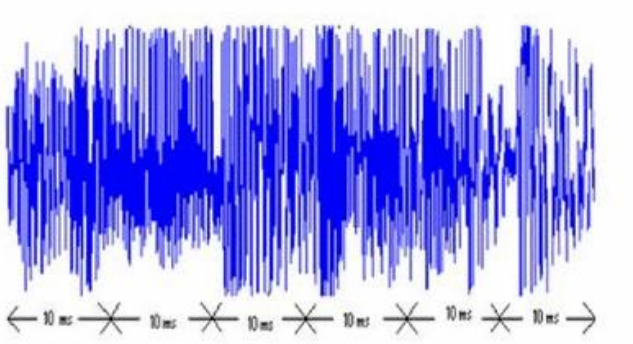

Figura 1.4.1.1 División de una señal de sonido.

2. Una Transformada Discreta de Fourier (DFT) se aplica a cada segmento para crear una matriz de las fases y magnitudes de transformada de Fourier

3. Se calculan diferencias de fase entre los segmentos adyacentes.

4. Los desfases entre segmentos adyacentes son fácilmente detectable. Esto significa, podemos cambiar las fases absolutas de los segmentos, pero las diferencias de fase relativos entre segmentos adyacentes debe ser preservado. Así que la información secreta sólo se inserta en el vector de fase de la primera señal de segmento como se muestra en la Figura 1.4.1.2

$$Phase_{new} = \begin{cases} \frac{\pi}{2} \; if \; message \; bit = 0 \\ \frac{-\pi}{2} \; if \; message \; bit = 1 \end{cases}$$

Figura 1.4.1.2 Vector de fase de la primera señal de segmento

5. Una nueva matriz de fase se crea utilizando la nueva fase del primer segmento y las diferencias de fase originales.

6. La señal de sonido se reconstruye aplicando el inverso Transformada Discreta de Fourier usando la nueva matriz de fase y matriz original magnitud y luego concatenando los segmentos de sonido de nuevo juntos. El receptor debe conocer la longitud del segmento para extraer la información en secreto para el archivo de sonido. Entonces el receptor puede usar la DFT para obtener las fases y extraer la información secreta (en la Figura 1.4.1.3 se ejemplifica el procedimiento de codificación de fase)

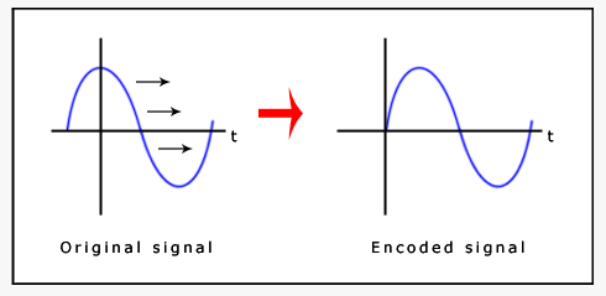

Figura 1.4.1.3 Procedimiento de codificación de fase

1.4.2 SPREAD SPECTRUM.

Propaga datos a ocultar en la señal a través del espectro de frecuencias, Spread Spectrum (SS) es un concepto desarrollado en las comunicaciones para asegurar una recuperación adecuada de una señal enviada por un canal ruidoso mediante la producción de copias redundantes de la señal de datos. Este método intenta difundir información secreta a través del espectro de frecuencia de señal de audio tanto como sea posible, es decir propaga el mensaje secreto sobre espectro de frecuencias del archivo de sonido, usando un código que es independiente de la señal real. Como resultado, la señal final ocupa un ancho de banda en exceso de lo que realmente se requiere para la transmisión.

Dos versiones de SS se pueden utilizar en la esteganografía de audio:

- Secuencia directa. El mensaje secreto se extiende por una constante llamada la velocidad de chip y luego modulada con una señal pseudo aleatoria. Se intercala después con la señal de la cubierta.
- Salto de frecuencia. El espectro de frecuencia de archivos de audio se altera de modo que salta rápidamente entre las frecuencias.

En la Figura 1.4.2.1 se ilustra el diseño del spread spectrum cuando se aplica a en la esteganografía para archivos de audio.

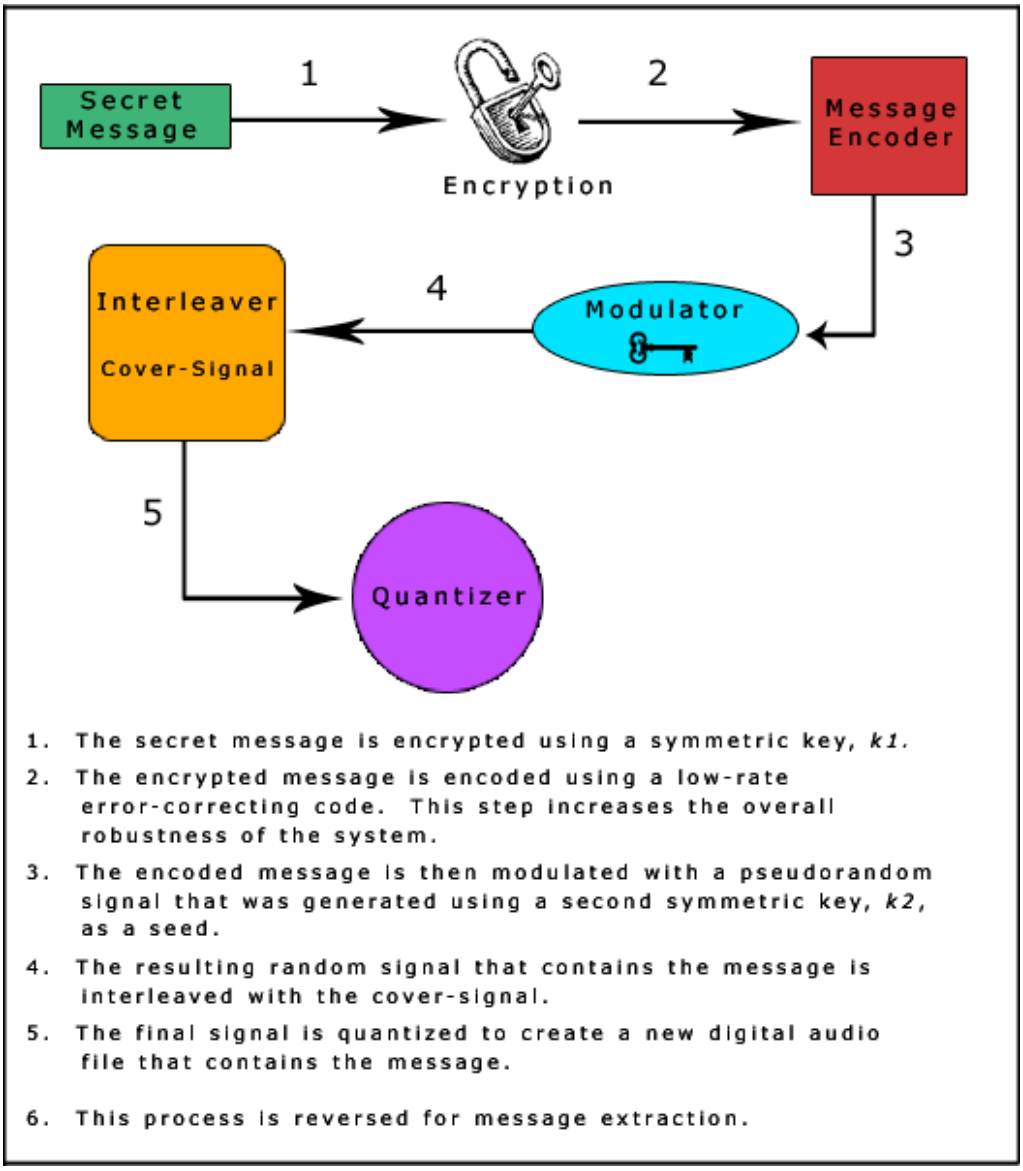

Figura 1.4.2.1. Spread spectrum aplicado en la esteganografía.

1.4.3 ECHO HIDING

En este método la información es incrustada en un archivo de sonido mediante la introducción de un eco en la señal discreta. Al igual que el método de spread spectrum, también ofrece ventajas en que permite una alta velocidad de transmisión de datos.

Para ocultar los datos correctamente, tres parámetros del eco son variadas: la amplitud, la velocidad de desintegración, y el desplazamiento (tiempo de retardo) de la señal original. Los tres parámetros se fijan por debajo del umbral de la audición humana por lo que el eco no se resuelve fácilmente. Además, el desplazamiento se varía para representar el mensaje binario a codificar. Un valor de desplazamiento

representa un uno binario, y un valor de segundo desplazamiento representa un cero binario.

Si sólo uno de eco se produce a partir de la señal original, sólo un bit de información podría ser codificado. Por lo tanto, la señal original se descompone en bloques antes de que comience el proceso de codificación. Una vez que se completa el proceso de codificación, los bloques se concatenan de nuevo juntos para crear la señal final.

Para comprender de mejor manera el mostraremos el proceso para ocultar la palabra “HEY”, primero vamos a dividir la señal por completo en bloques, aunque es recomendable tener un número aleatorio de muestras entre cada par de bloques sin uso para reducir la probabilidad de detección. En la Figura 1.4.3.1 se muestra como la señal se divide en bloques, y cada bloque se le asigna un uno o un cero basado en el mensaje secreto, para este caso, el mensaje es el equivalente binario de “HEY “.

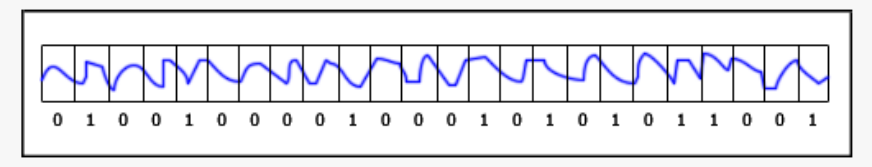

Figura 1.4.3.1 Mensaje a ocultar dividido en bloques.

Usando que la aplicación del proceso de ocultación de eco por lo general puede resultar en una señal que tiene una mezcla bastante notable de ecos, aumentando así el riesgo de detección. Una segunda aplicación del proceso de ocultación de eco aborda este problema. En primer lugar una señal de eco se crea a partir de toda la señal original con el valor binario cero offset. A continuación, una segunda señal de eco se crea a partir de toda la señal original utilizando el valor de compensación binario. Así, el "uno" señal de eco sólo contiene solo unos, y la señal de eco "cero" sólo contiene ceros. Para combinar los dos ecos en conjunto para obtener la codificación final, se utilizan dos señales del mezclador. Las señales del mezclador tienen un valor de uno o cero, dependiendo de qué bit se va a codificar en el bloque. En nuestro ejemplo utilizando el mensaje 'HEY', que se pueden conseguir las dos señales del mezclador siguiente, tal y como se muestra en La Figura 1.4.3.2.

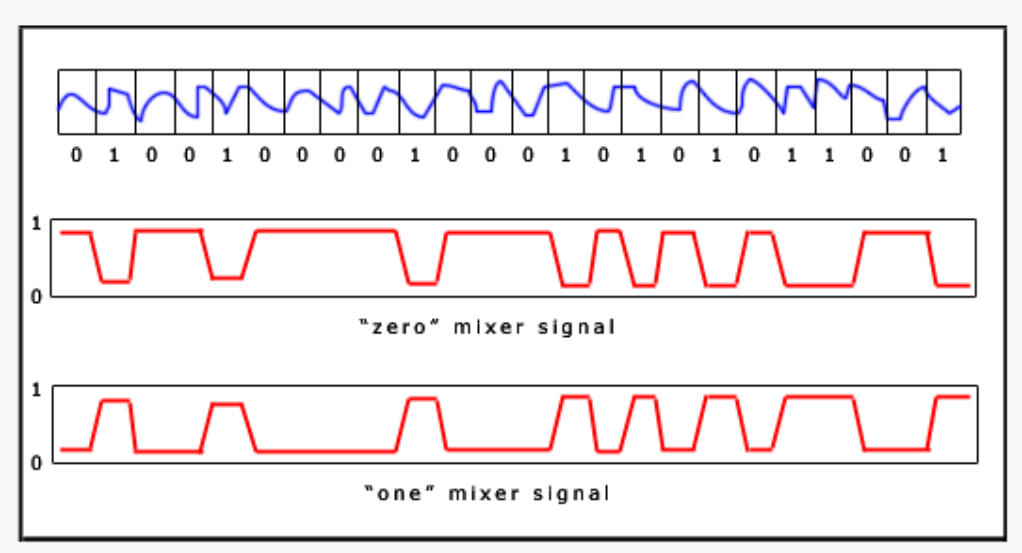

Figura 1.4.3.2 Mensaje y señales obtenidas.

El "uno" señal de eco se multiplica por el "uno" de la señal del mezclador y la señal de eco "cero" se multiplica por la señal del mezclador "cero". Luego se añaden los dos resultados juntos para obtener la señal de final. La señal final es menos abrupta que la obtenida usando el primer eco de ocultar la implementación. Esto es porque los dos ecos del mezclador son complementos el uno del otro y que las transiciones de rampa se utilizan dentro de cada señal. Estas dos características de las señales del mezclador producen transiciones más suaves entre ecos.

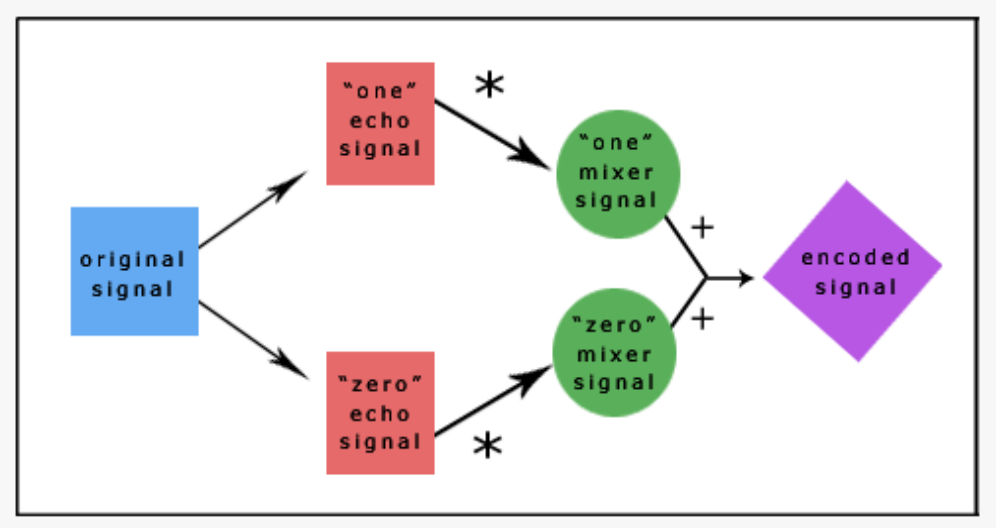

Figura 1.4.3.3 Aplicación del proceso de ocultación de eco.

Para extraer el mensaje secreto de la señal final, el receptor debe ser capaz de romper la señal en la misma secuencia de bloque utilizado durante el proceso de codificación. A continuación, la función de auto correlación del espectro de la señal, la Transformada de Fourier del espectro de frecuencias de la señal puede ser usada

para decodificar el mensaje porque revela un pico en cada tiempo de eco compensado, lo que permite que el mensaje pueda ser reconstruido.

1.4.4 LSB

Al igual que las imágenes de documentos, los archivos de sonido pueden ser modificados de tal manera que contienen información oculta, como la información de derechos de autor; esas modificaciones se deben hacer de tal manera que sea imposible que un pirata para eliminarla, al menos no sin destruir la señal original. Los métodos que incorpora datos en archivos de sonido utilizan las propiedades del sistema auditivo humano (SAH). El SAH percibe el ruido aleatorio aditivo así como las perturbaciones en un archivo de sonido también puede ser detectado. Pero hay algunos "agujeros" que pueden explotar. Mientras que el SAH tener un gran rango dinámico, que tiene un rango bastante pequeño diferencial.

La sustitución del bit menos significativo es la forma más sencilla para incrustar información en un archivo de audio digital. Al sustituir el bit menos significativo de cada punto de muestreo con un mensaje binario, LSB de codificación permite una gran cantidad de datos a codificar. En LSB codificación, la velocidad de transmisión de datos ideal es de 1 kbps por 1 kHz. En algunas implementaciones de codificación LSB, sin embargo, los dos bits menos significativos de una muestra se sustituyen con dos bits de mensaje. Esto aumenta la cantidad de datos que se puede codificar sino que también aumenta la cantidad de ruido resultante en el archivo de audio. Un nuevo método que aumenta el límite máximo de cuatro bits para extraer un mensaje secreto a partir de un archivo de sonido codificada LSB, el receptor necesita tener acceso a la secuencia de los índices de la muestra utilizados en el proceso de incrustación. Normalmente, la longitud del mensaje secreto para ser codificado es menor que el número total de muestras en un archivo de sonido. Uno debe decidir entonces sobre cómo elegir el subconjunto de muestras que contendrá el mensaje secreto y comunicar esa decisión al receptor. Una técnica trivial es empezar al principio del archivo de sonido y realizar codificación LSB hasta que el mensaje ha sido completamente incrustado, dejando las muestras restantes sin cambios. Esto crea un problema de seguridad, sin embargo, en que la primera parte del archivo de sonido tendrá diferentes propiedades estadísticas que la segunda parte del archivo de sonido que no se ha modificado. Una solución a este problema es para rellenar el mensaje secreto con bits aleatorios de modo que la longitud del mensaje es igual al número total de muestras.

Hay dos desventajas principales asociados con el uso de métodos como el LSB de codificación. El oído humano es muy sensible y con frecuencia puede detectar la más mínima poco de ruido introducido en un archivo de sonido, segunda desventaja sin embargo, es que esto no es robusto. Si un archivo de sonido integrado con un mensaje secreto utilizando LSB codificación era volver a muestrear, la información incrustada se perdería. Robustez se puede mejorar algo mediante el uso de una técnica de redundancia, mientras que codifica el mensaje secreto. Sin embargo, las técnicas de redundancia reducir la tasa de transmisión de datos significativamente.

CAPÍTULO II

ANÁLISIS Y DISEÑO.

2.1 APLICACIONES DE LA ESTEGANOGRAFÍA.

Contrario a lo que se puede creer la esteganografía se a practicado desde que el hombre tuvo la necesidad de ocultar información, sin embargo en la era digital a tomado un nuevo impulso, esto a consecuencia de las aplicaciones que tiene, como son:

- *Comunicación confidencial y almacenamiento de datos secretos.* En este ámbito la esteganografía nos proporciona la capacidad para ocultar la existencia de datos confidenciales, el hacer imposible el poder detectar los datos ocultos y finalmente nos da la opción de que los datos a ocultar se encuentren encriptados.

- *Protección de la alteración de datos*. En este caso la podemos ver a la esteganografía como un "Sistema de certificado de documentos digitales", es decir, los usuarios pueden mandar sus datos ocultos dentro de un objeto contenedor desde y hacia cualquier parte del mundo a través de la web, teniendo la seguridad de que nadie puede falsificar, manipular o alterar estos datos, si este objeto contenedor llegará a ser manipulado o alterado la extracción de los datos ocultos en éste resultaría imposible.

- *Sistema de control de acceso para la distribución de contenidos digitales*. Hoy en día, los contenidos digitales son cada vez más y más comúnmente distribuidos a través de Internet. Por ejemplo, las compañías de música liberan nuevos álbumes en su página web de forma gratuita o con algún costo. Sin embargo, en este caso, todos los contenidos se distribuyen por igual a las personas que pueden hacer el acceso a la página. Por lo tanto, un sistema de distribución Web ordinaria no es adecuado para un "caso por caso" y/o distribución "selectiva". Es aquí donde a través de la esteganografía se puede emitir una "clave de acceso" especial para extraer el contenido de forma selectiva a los archivos que sean cargados a internet.

- *Sistemas de bases de datos de Medios*. En esta área de aplicación el secreto esteganografía no es importante, pero la unificación de dos tipos de datos en una sola es la más importante, esto es debido a que los datos de los medios de comunicación (imagen de la foto, cine, música, etc.) tienen alguna relación con otra información. Una foto de la imagen, por ejemplo, puede tener l título de la imagen y alguna información objeto físico, la fecha y el momento en que se tomó la imagen, la cámara y la información de los fotógrafos, etc. Cuando se quiere encontrar una imagen específica dentro de un grupo de imágenes, la búsqueda resulta complicada debido a que no todas

las imágenes pueden contar con los datos antes mencionados. Este problema se conoce técnicamente como "metadatos (por ejemplo, los datos de anotación) y en un sistema de base de datos de medios de comunicación (un software álbum de fotos) se separan de los datos de medios (datos de la foto) para el sistema de gestión de base de datos (DBMS), esta separación resulta ser un enorme problema, es aquí en donde entra la esteganografía ya que unifica dos tipos de datos en uno por medio de la incorporación de operación. Así, los metadatos puede ser fácilmente transferida de un sistema a otro sin ningún problema. En concreto, puede incrustar toda la información deseada instantáneamente en cada fotografía digital.

2.2 SISTEMA PROPUESTO

Este trabajo se enfoca en 2 aplicaciones de la esteganografía: Comunicación confidencial y almacenamiento de datos secretos y la Protección de la alteración de datos, del mismo modo utiliza como objeto contenedor un archivo de audio con formato WAV, debido a que este tipo de archivo no es un formato comprimido como el mp3, como consecuencia el WAV es un archivo grande, lo que nos permite ocultar un mayor número de datos.

El sistema a desarrollar nos permitirá ocultar información (texto simple) dentro de un archivo de audio con formato WAV, del mismo modo se podrá aplicar el proceso inverso obteniendo los datos ocultos tal y como se muestra en la Figura 2.2.1

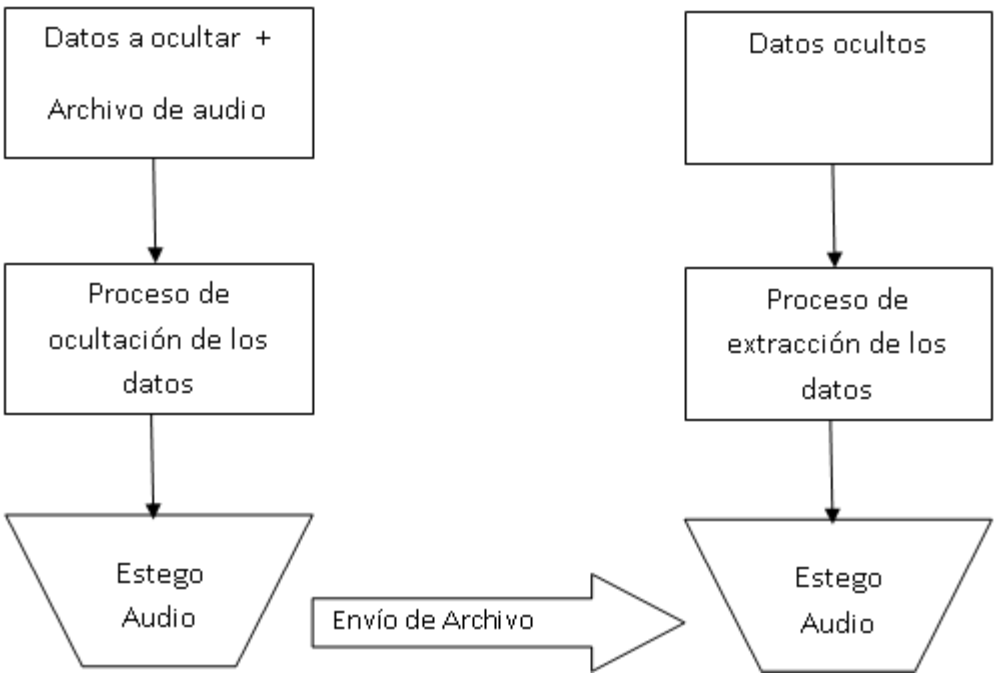

Figura 2.2.1 Diagrama del funcionamiento general del sistema

En el diagrama anterior podemos observar los dos procesos que podrá llevar a cabo el sistema, exceptuando el "Envío del archivo", éste dependerá completamente del usuario, durante el proceso de ocultamiento de datos de datos de aplica un algoritmo esteganográfico a un objeto contenedor, en este caso se aplicará el algoritmo LSB y el objeto contenedor será un archivo de audio en formato WAV, como resultado de este proceso se obtiene un estego-audio (es decir cuando los datos se encuentran dentro del archivo de audio). En palabras más simples el sistema se encargará de producir este estego-audio uniendo el texto que se desea ocultar, el cual es ingresado a través de una ventana en el sistema por el usuario y un archivo de audio, de igual manera el sistema se encargará de procesar el estego audio para obtener los datos ocultos.

2.3 CASOS DE USO

El diagrama de casos de uso sirve para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas. O lo que es igual, un diagrama que muestra la relación entre los actores y los casos de uso en un sistema. dentro de un diagrama de casos de uso encontramos los siguientes elementos:

- *Casos de Uso.* secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema.

- *Actores.* Esto es aquello que interactúa con el sistema, desde una persona, la cual tiene un rol dentro del mismo (usuario, administrador, etc.), hasta otro sistema informático.

En este sistema sólo contamos con dos actores, un emisor y un receptor, tal y como se muestra en la Figura 2.3.1

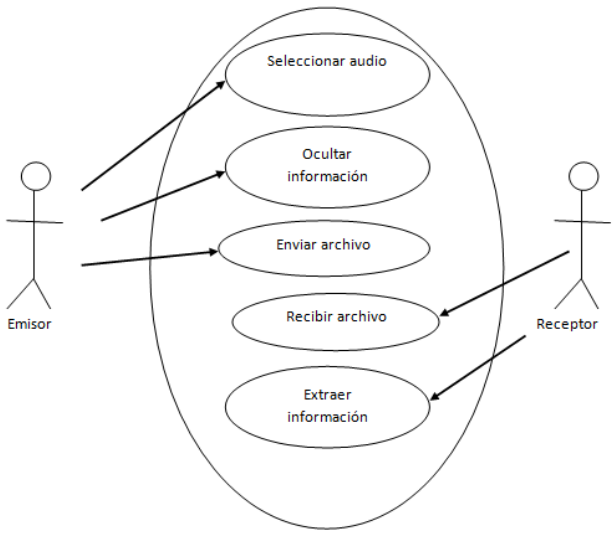

Figura 2.3.1 Diagrama de Casos de Uso.

Este diagrama de casos de uso nos muestra el funcionamiento del sistema, en el cual encontramos dos actores que hacen uso del mismo, un emisor y un receptor, el emisor pretende compartir información que él considera delicada, es por ello que antes de enviarla decide ocultarla, para ello primero debe seleccionar un archivo de audio en el cual ocultar la información, posteriormente el sistema se encargará de generar el estego-audio, pero el envío de esta es totalmente independiente al sistema, el usuario puede tomar cualquier opción para transportar el estego-audio, a través de internet, en un dispositivo de almacenamiento, etc. Finalmente nuestro último actor (receptor) después de recibir el estego-audio lo procesa y obtiene la información oculta. El sistema permite que los roles se los actores se intercambien tantas veces como se desee y esto no ocasiona ningún problema.

2.4 DIAGRAMA DE FLUJO.

El diagrama de flujo es una representación gráfica de un proceso. Cada paso del proceso es representado por un símbolo diferente que contiene una breve descripción de la etapa del mismo. Los símbolos gráficos del flujo del proceso están unidos entre sí con flechas que indican la dirección de flujo del proceso.

El diagrama de flujo ofrece una descripción visual de las actividades implicadas en un proceso mostrando la relación secuencial ente ellas, facilitando la rápida comprensión de cada actividad y su relación con las demás, el flujo de la información y los materiales, las ramas en el proceso, la existencia de bucles repetitivos, el número de pasos del proceso, las operaciones de interdepartamentales.

A menudo, se utilizan para especificar algoritmos de manera gráfica, ya que éstos esquematizan conceptos vinculados a la programación, la economía, los procesos técnicos y/o tecnológicos, la psicología, la educación y casi cualquier temática de análisis. Como se había mencionado anteriormente estos diagramas utilizan símbolos con significados definidos que representan los pasos del algoritmo, y representan el flujo de ejecución mediante flechas que conectan los puntos de inicio y de fin del proceso.

- Los símbolos más utilizados en este tipo de diagramas son:
- *Óvalo o Elipse*: Inicio y Final (Abre y cierra el diagrama).
- *Rectángulo*: Actividad (Representa la ejecución de una o más actividades o procedimientos).
- *Rombo*: Decisión (Formula una pregunta o cuestión).

- *Círculo*: Conector (Representa el enlace de actividades con otra dentro de un procedimiento).

- *Triángulo boca abajo*: Archivo definitivo (Guarda un documento en forma permanente).

- *Triángulo boca arriba*: Archivo temporal (Proporciona un tiempo para el almacenamiento del documento).

Además encontramos diferentes formatos de diagramas de flujo, los que mencionamos a continuación:

- *Formato vertical*: En él, el flujo y la secuencia de las operaciones, va de arriba hacia abajo. Es una lista ordenada de las operaciones de un proceso con toda la información que se considere necesaria, según su propósito.

- *Formato horizontal*: En él, el flujo o la secuencia de las operaciones, va de izquierda a derecha.

- *Formato panorámico*: El proceso entero está representado en una sola carta y puede apreciarse de una sola mirada mucho más rápido que leyendo el texto, lo que facilita su comprensión, aun para personas no familiarizadas. Registra no solo en línea vertical, sino también horizontal, distintas acciones simultáneas y la participación de más de un puesto o departamento que el formato vertical no registra.

- *Formato Arquitectónico*: Describe el itinerario de ruta de una forma o persona sobre el plano arquitectónico del área de trabajo. El primero de los diagramas de flujo es eminentemente descriptivo, mientras que los utilizados son fundamentalmente representativos.

En esta ocasión utilizaremos el formato vertical, para explicar los diferentes diagramas de flujo del sistema.

A continuación la Figura 2.4.1 nos muestra el primer diagrama de flujo, en el cual podemos observar los diferentes procesos que ocurren para generar el estego-audio.

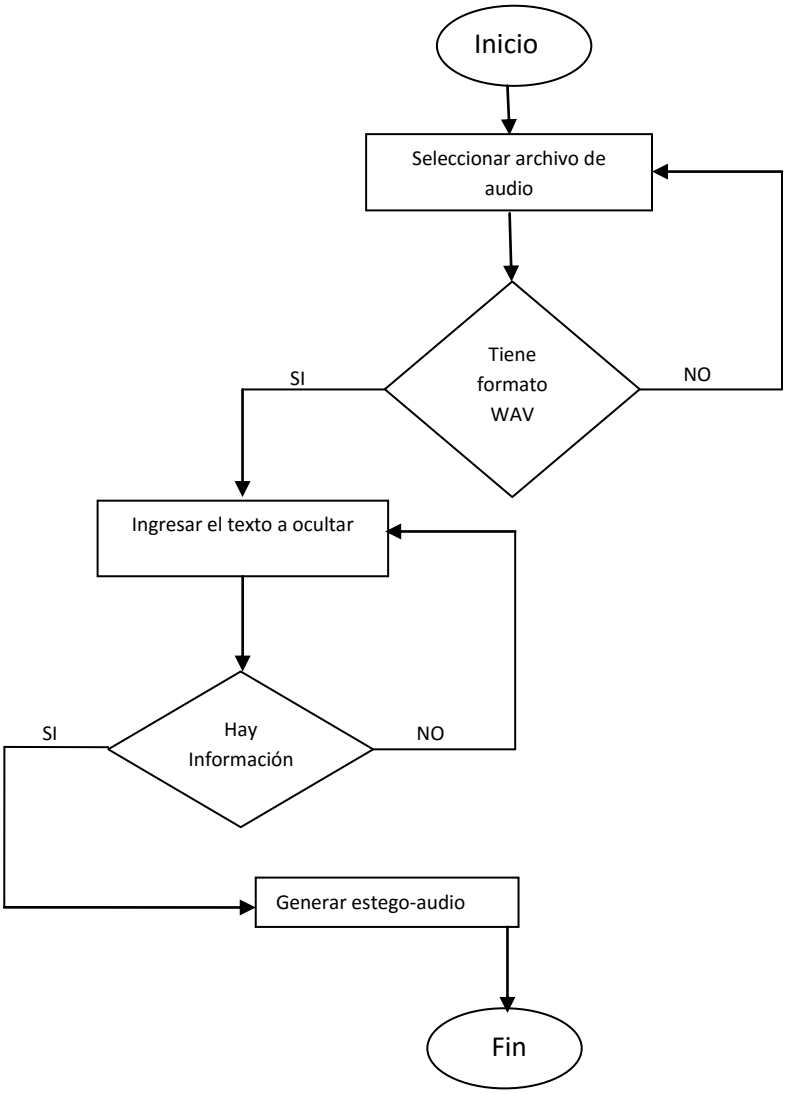

Figura 2.4.1 Generando estego-audio.

Como se puede observar en el diagrama anterior (Figura 2.4.1) el sistema obliga a que el archivo se audio seleccionado como objeto contenedor tenga el formato adecuado, de no ser así no permite continuar con el proceso esteganográfico, ocurre lo mismo con el texto que se va a ocultar, éste se ingresará a través de una caja de texto, si esta caja de texto está vacía el proceso no continúa, una vez que se ha seleccionado el archivo adecuado e ingresado la información el proceso esteganográfico se lleva a cabo generando así el estego-audio.

Ahora en el siguiente diagrama que se muestra en la Figura 2.4.2 observamos el cómo se obtiene la información oculta.

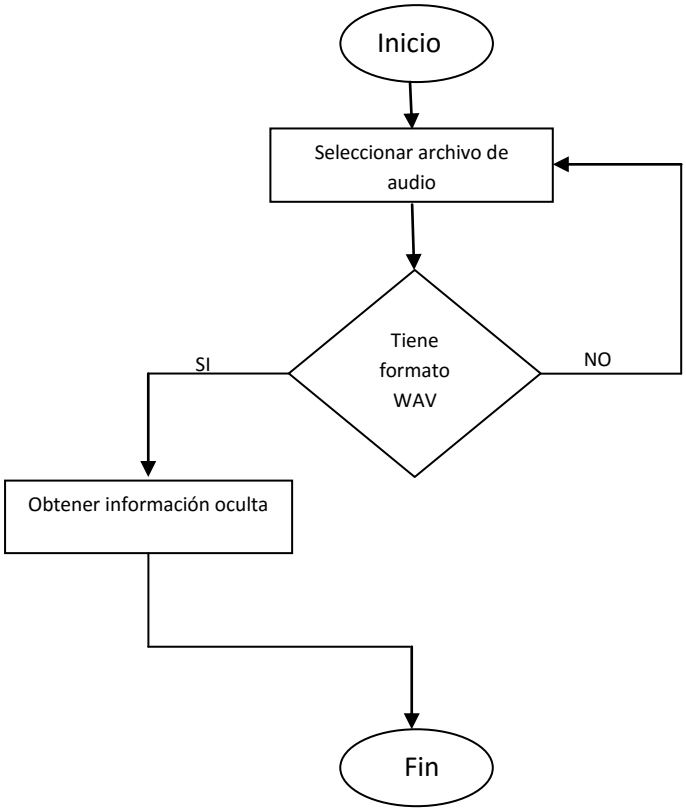

Figura 2.4.2 Obteniendo información.

Observando el diagrama anterior (Figura 2.4.2) podemos apreciar que para obtener la información oculta se necesita un solo proceso, así mismo se verifica que el archivo de audio este en el formato adecuado, de lo contrario no se puede seguir con el proceso.

2.5 DIAGRAMA DE ITERACIÓN

Los diagramas de iteración son modelos que describen como grupos de objetos colaboran para conseguir algún fin. Estos diagramas aparte de mostrar los objetos, muestran los mensajes que pasan entre ellos, de la misma forma capturan el comportamiento de un caso de uso. Los diagramas de interacción se expresan de dos maneras:

- Diagramas de secuencia
- Diagramas de colaboración.

Cada uno de estos diagramas resalta ciertos aspectos en particular. Los diagramas de colaboración muestran las relaciones entre los objetos y los mensajes que intercambian, mientras que un diagrama de secuencias muestra las interacciones expresadas en función de secuencias temporales, es decir muestra los objetos participantes en la interacción por sus líneas de vida, y los mensajes que intercambian entre ellos al correr del tiempo, sin embargo no muestran los enlaces entre los objetos.

En este caso utilizaremos un diagrama de secuencia para mostrar las interacciones expresadas en función del tiempo, sin embargo antes de observar éste diagrama hay que entender una serie de aspectos más para poder comprenderlo al 100%.

Un diagrama de secuencia, en concreto muestra los objetos participantes y los mensajes que intercambian entre ellos a lo largo del tiempo, igualmente son más apropiados para especificar restricciones de interacción en tiempo real, un diagrama de secuencia tiene dos dimensiones:

- La vertical , que representa el tiempo
- La horizontal, que representa los distintos objetos.

Dentro de estos diagramas el tiempo avanza desde el comienzo hasta el final de la página, aunque se puede tomar en sentido contrario, la exactitud temporal solo toma importancia en las aplicaciones de tiempo real, por lo que los ejes del tiempo suelen tener marcas temporales, por otro lado el orden horizontal de la aparición de los objetos no tiene ninguna importancia.

Una vez comprendidos estos aspectos, observaremos en la Figura 2.5.1 el diagrama de secuencia del sistema cuando se oculta información.

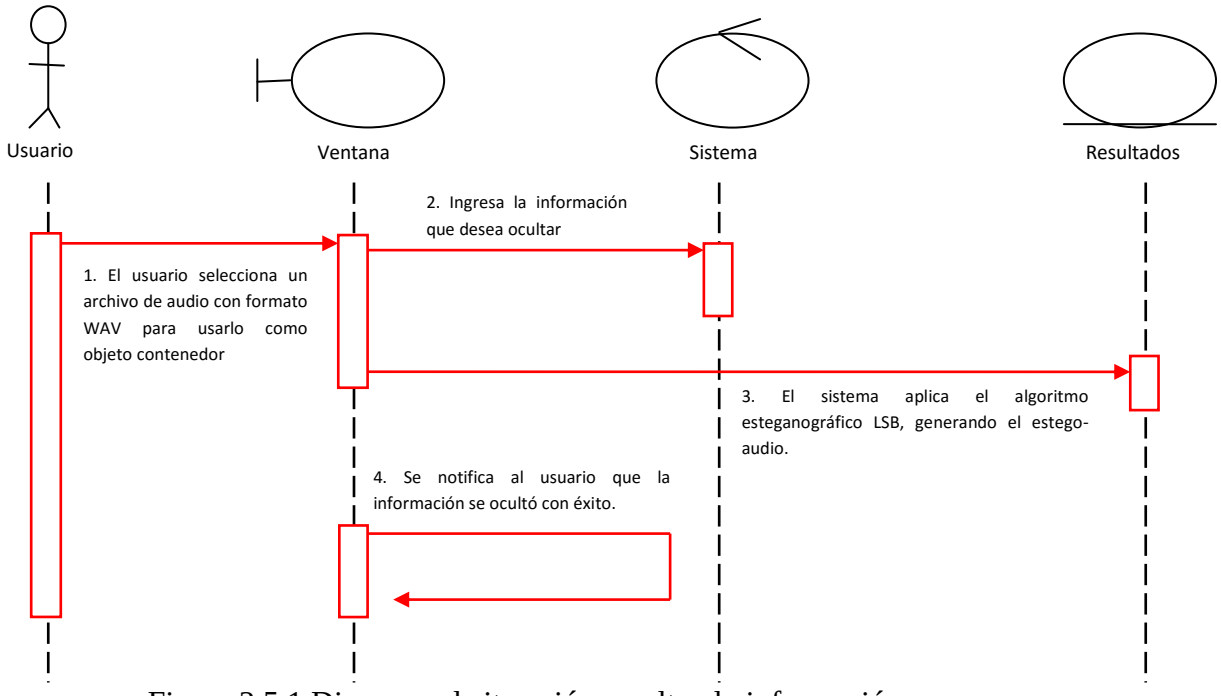

Figura 2.5.1 Diagrama de iteración, ocultando información.

En este diagrama (Figura 2.5.1) se muestra de forma más específica la interacción que se lleva a cabo entre el usuario y el sistema. Primero el sistema mostrará una ventana a través de la cual el usuario podrá seleccionar un archivo de audio con formato WAV para usarlo como objeto contenedor, posteriormente en la misma

ventana el usuario encontrará una caja de texto en la cual podrá introducir los datos (texto) que desea ocultar, posteriormente el sistema aplica el algoritmo esteganográfico LSB para audio, con el fin de ocultar los datos, y finalmente se notifica al usuario si el proceso terminó de manera exitosa.

A continuación, en la Figura 2.5.2 observamos un segundo diagrama de iteración, el cual nos explica el proceso que ocurre cuando el usuario procesa un archivo de audio para obtener los datos ocultos que se encuentran en el.

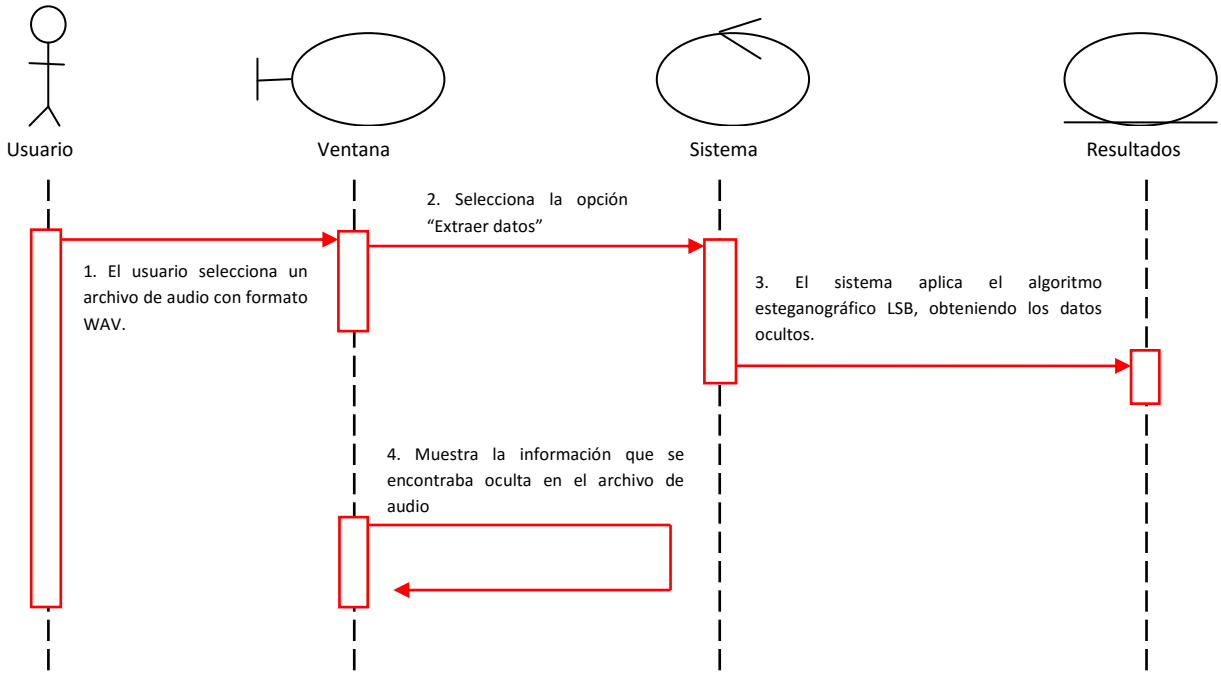

Figura 2.5.2 Diagrama de iteración, obteniendo datos ocultos.

Como podemos observar, este diagrama (Figura 2.5.2) no difiere mucho del que se mostró en la Figura 2.5.1. Por el momento centrémonos en el diagrama actual (Figura 2.5.2), de igual forma se muestra una ventana a través de la cual el usuario selecciona un archivo de audio con formato WAV, posteriormente en la misma ventana selecciona "Extraer datos" para que el sistema recupere los datos ocultos en dicho archivo, finalmente se mostrará la información recuperada en una caja de texto ubicada en la misma ventana.

2.6 DISEÑO DE LA INTERFAZ DE USUARIO.

A través de este capítulo se han especificado todos los diagramas que en algunos casos componen y otros ejemplifican el sistema, es momento ahora de definir la interfaz del usuario, ésta tiene que ser sencilla e intuitiva para así facilitar el uso del sistema al usuario.

En la Figura 2.6.1 podemos observar la ventana principal.

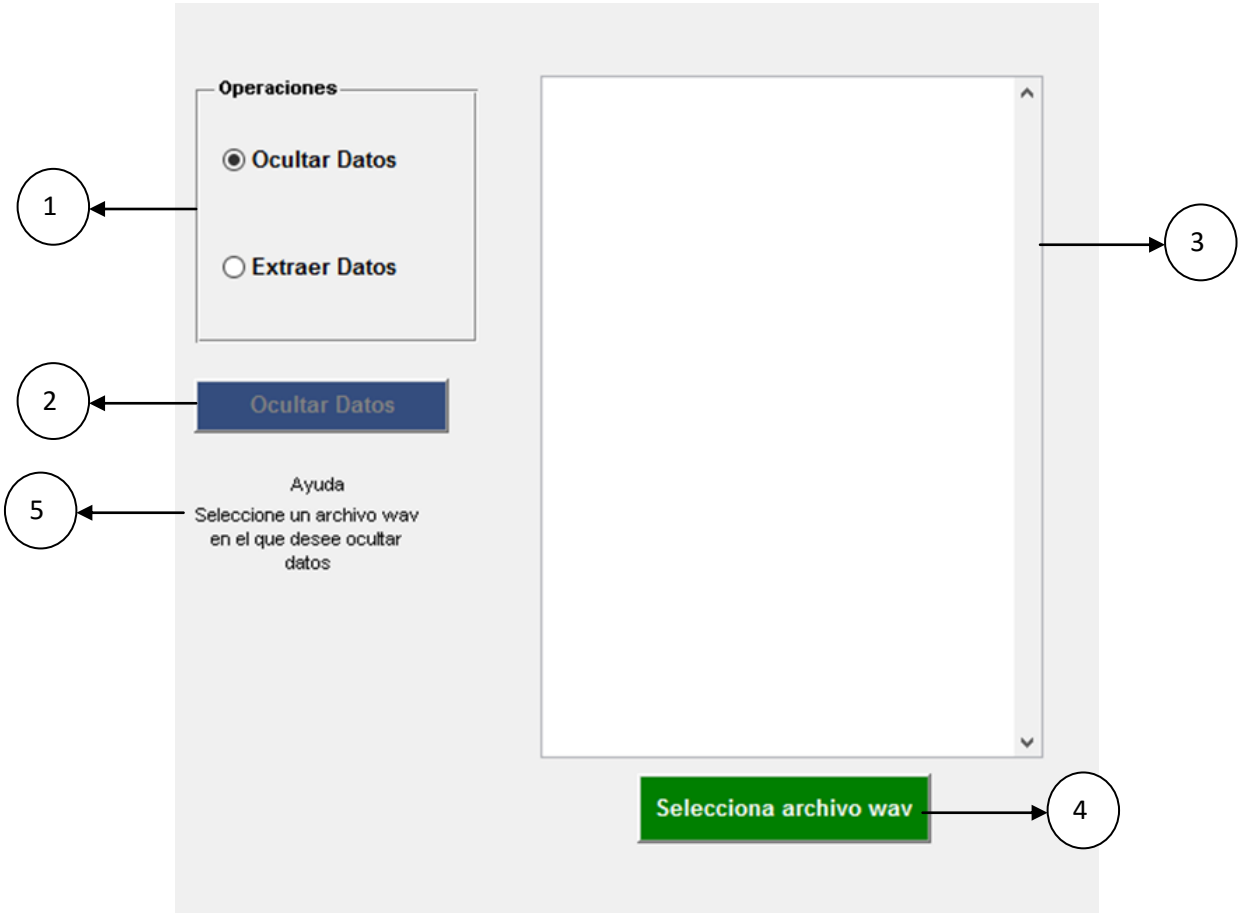

Figura 2.6.1 Pantalla principal.

En esta ventana tenemos cinco elementos fundamentales, el número uno nos muestra las operaciones que puede realizar el sistema (ocultar y extraer datos), en el número dos encontramos el botón "Ocultar Datos", el cual al realizar dicha acción aplica el Algoritmo LSB para ocultar los datos en el objeto contenedor, posteriormente en el número tres encontramos un cuadro de texto, en el cual colocaremos la información que se desea ocultar, el cuarto elemento nos permite seleccionar el archivo wav para realizar cualquier operación mencionada en el elemento uno, finalmente en el elemento número cinco encontramos un pequeño cuadro de texto el cual proporciona ayuda al usuario.

A partir de la ventana principal encontramos un par de avisos al usuario para distintas situaciones, en la Figura 2.6.2, se muestra cuando no se a escrito algún texto para ocultarse.

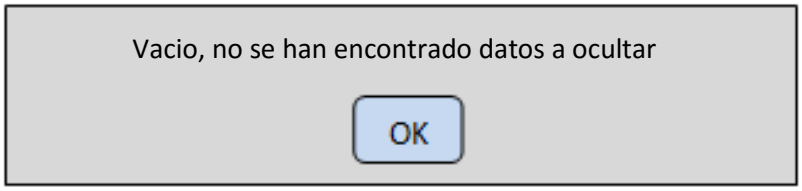

Figura 2.6.2 Aviso de formato invalido

La segunda ventana notifica al usuario que los datos se han ocultado correctamente, permitiéndole así guardar y/o enviar el estego-audio de la manera que más le convenza, dicha ventana de muestra en la Figura 2.6.3.

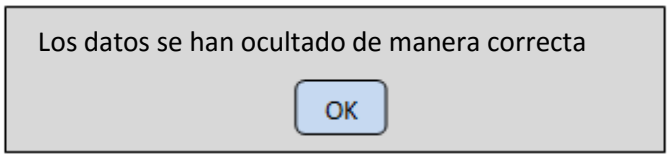

Figura 2.6.3 Aviso de proceso concluido correctamente.

CAPÍTULO III

IMPLEMENTACIÓN

3.1 HERRAMIENTA DE DESARROLLO.

MATLAB (abreviatura de MATrix LABoratory, "laboratorio de matrices") es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Windows, Mac OS X y GNU/Linux .

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones, a saber, Simulink (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario - GUI). Además, se pueden ampliar las capacidades de MATLAB con las cajas de herramientas (toolboxes); y las de Simulink con los paquetes de bloques (blocksets).

Es un software muy usado en universidades y centros de investigación y desarrollo. En los últimos años ha aumentado el número de prestaciones, como la de programar directamente procesadores digitales de señal o crear código VHDL. Al ser un lenguaje de alto nivel y un entorno interactivo utilizado por millones de ingenieros y científicos de todo el mundo. Se le permite explorar y visualizar las ideas y la colaboración entre disciplinas, incluyendo la señal y el procesamiento de imágenes, comunicaciones, sistemas de control, y las finanzas computacionales.

3.2 IMPLEMENTACIÓN

La implementación del sistema se realizó usando Matlab, esto facilitó un poco el trabajo ya que esta herramienta ya cuenta con funciones que nos permiten seleccionar y manipular archivos multimedia de una manera más sencilla, de igual manera nos permite crear una interfaz con el usuario de una forma más rápida.

En la Figura 3.2.1 se muestra la interfaz principal del sistema.

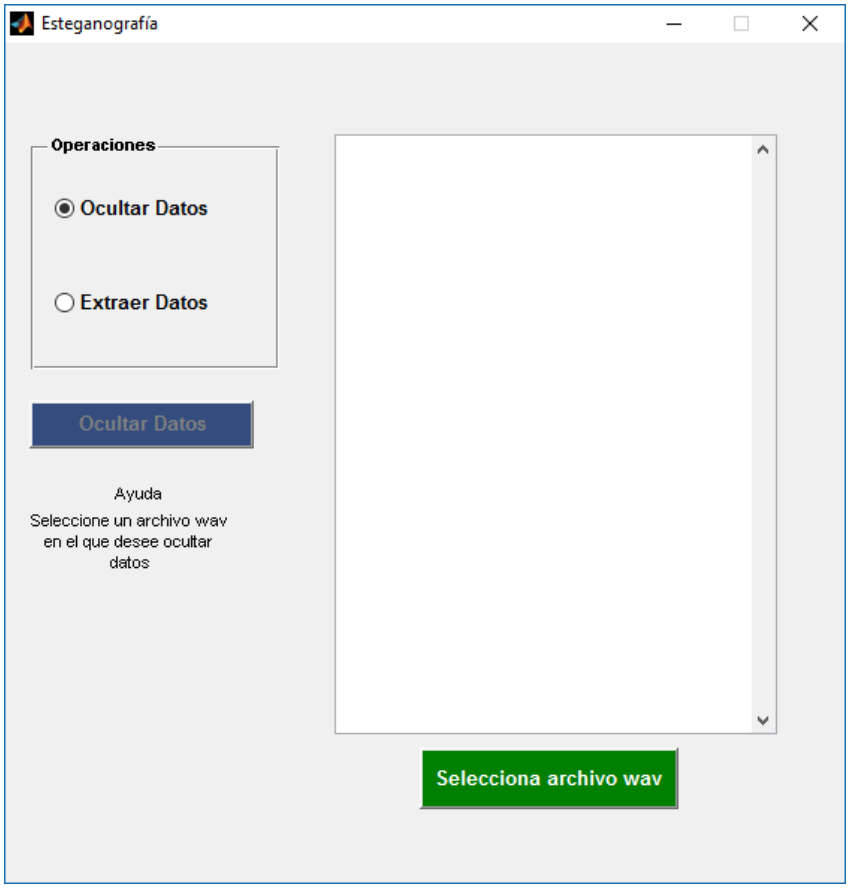

Figura 3.2.1 Ventana principal del Sistema

Empezaremos por explicar la parte de Ocultamiento de datos, para ello es esencial que primero seleccionemos un archivo de audio con formato wav el cual se utilizará como objeto contenedor, para ello el sistema nos abre una ventana, la cual se muestra en la Figura 3.2.2

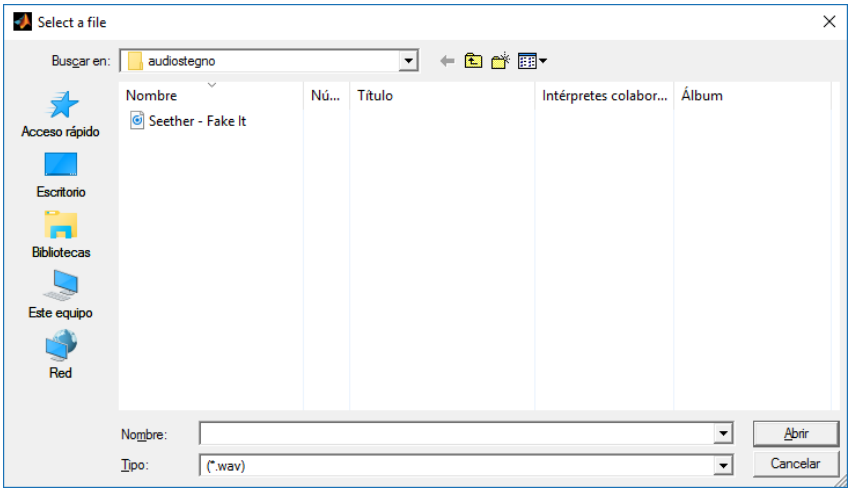

Figura 3.2.2 Seleccionando archivo de audio.

Ésta ventana filtra de forma automática los archivos con el formato requerido por el sistema, para que el usuario solo seleccione el archivo deseado, una vez seleccionado el archivo procederemos a escribir los datos a ocultar en el cuadro de texto correspondiente, como podemos observar en la Figura 3.2.3

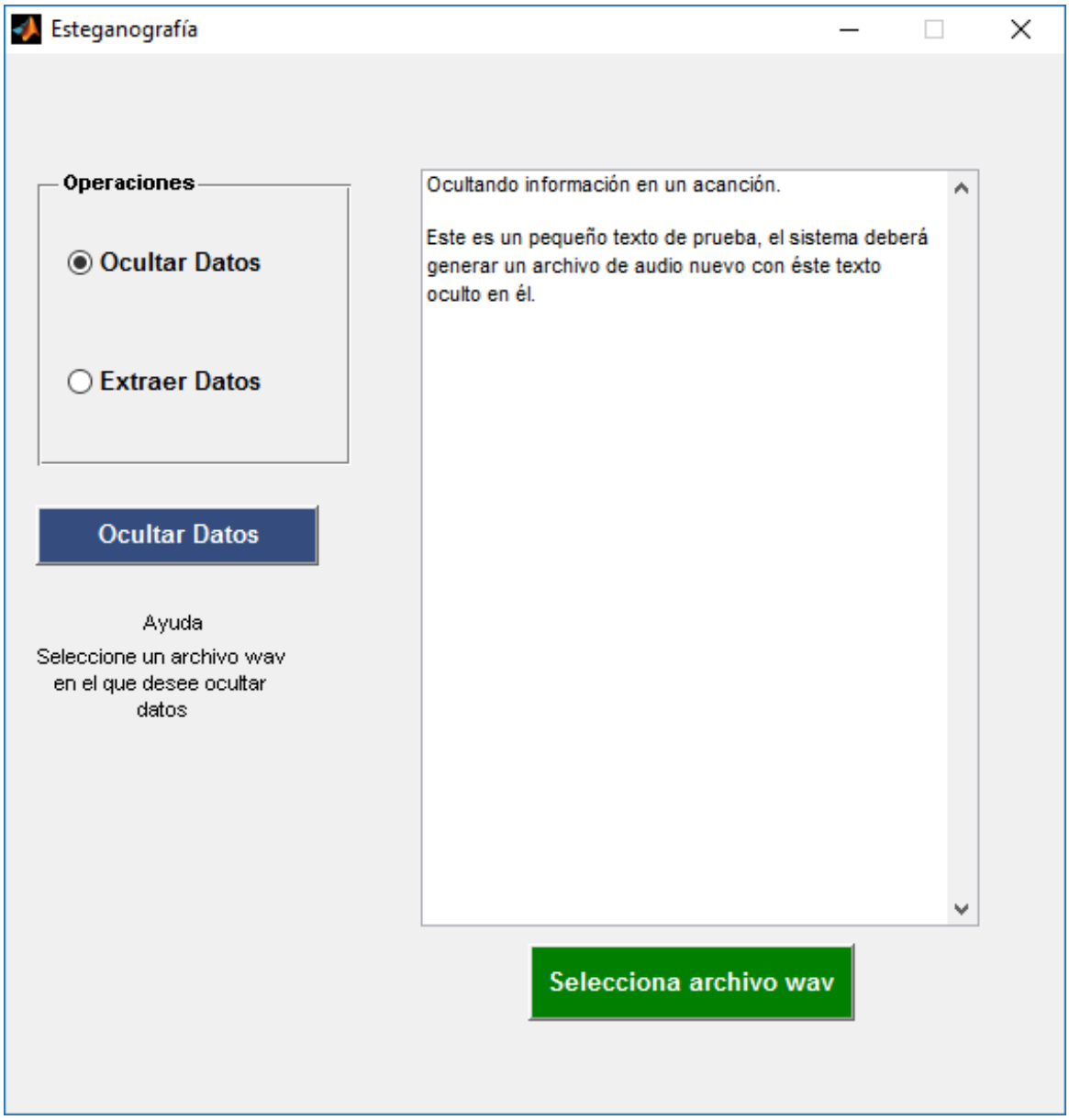

Figura 3.2.3 Introduciendo información a ocultar.

Una vez que se ha terminado de introducir dicha información, se hace clic en el botón “Ocultar Datos”. En caso de que el cuadro de texto de encuentre vacio se muestra un aviso al usuario, este aviso se puede observar en la Figura 3.2.4.

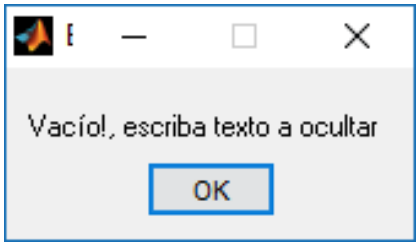

Figura 3.2.5 Aviso de texto vacío.

Por el contrario si se encuentra texto, se procesa, y de igual manera se notifica al usuario con el aviso de la Figura 3.2.6.

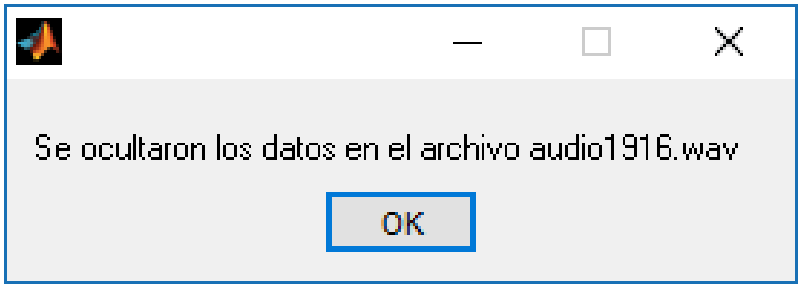

Figura 3.2.6 Aviso de texto ocultado con éxito.

Cuando el proceso ocurre de manera exitosa el sistema genera un nuevo archivo de audio con el texto oculto, el nombre de dicho archivo se muestra en este aviso.

A continuación se explicarán aquellas funciones que hacen posible el ocultar los datos de manera exitosa.

Primero se almacena una copia de los primeros cuarenta y tres bytes del archivo, este se debe a que los primeros 40 bytes conforman la cabecera del archivo, éstos no deben ser modificados de ninguna manera, los 3 bytes posteriores conforman la longitud de las muestras de datos del archivo wav, de igual forma éstos no deben modificarse, finalmente de copia el resto del contenido, este proceso se muestra en la Figura 3.2.7.

```
[y,fs,nbits,opts]=wavread([handles.pname handles.fname],[1 2]);
fid1=fopen([handles.pname handles.fname],'r');
header=fread(fid1,40,'uint8=>char');
data_size=fread(fid1,1,'uint32');
[dta,count]=fread(fid1,inf,'uint16');
fclose(fid1);
```

Figura 3.2.7 Respaldo del archivo wav.

Posteriormente se obtiene el texto a ocultar y se transforma a binario y procedemos a aplicar el algoritmo LSB finalmente creamos un archivo de audio nuevo, copiamos los elementos que no se deben modificar e inmediatamente insertamos una "identidad", esto es un arreglo binario específico el cual nos permitirá saber en un futuro que este archivo de audio contiene datos ocultos, después de la identidad de coloca el resto del archivo wav con los datos ocultos, finalmente se le asigna un nombre, formado por la palabra audio + un número aleatorio entre 1 y 2000. Lo antes mencionado se puede observar en la Figura 3.2.8.

```
[m_msg,n_msg]=size(msg);
msg_double=double(msg);
msg_bin=de2bi(msg_double,8);
[m,n]=size(msg_bin);
msg_bin_re=reshape(msg_bin,m*n,1);
m_bin=de2bi(m_msg,10)';
n_bin=de2bi(n_msg,10)';
len=length(msg_bin_re);
len_bin=de2bi(len,20)';
identity=[1 0 1 0 1 0 1 0]';
dta(1:8)=bitset(dta(1:8),lsb,identity(1:8));
dta(9:18)=bitset(dta(9:18),lsb,m_bin(1:10));
dta(19:28)=bitset(dta(19:28),lsb,n_bin(1:10));
dta(29:28+len)=bitset(dta(29:28+len),lsb,msg_bin(1:len)');
randname=num2str(randint(1,1,[1 2000]));
fid2=fopen(['audio' randname '.wav'],'w');
fwrite(fid2,header,'uint8');
fwrite(fid2,data_size,'uint32');
fwrite(fid2,dta,'uint16');
fclose(fid2);
```

Figura 3.2.8. Algoritmo LSB.

Ahora la manera en la que se extraen los datos es la siguiente, primero en la interfaz principal seleccionamos “Extraer Datos” en la parte de Operaciones, esto se muestra en la Figura 3.2.9

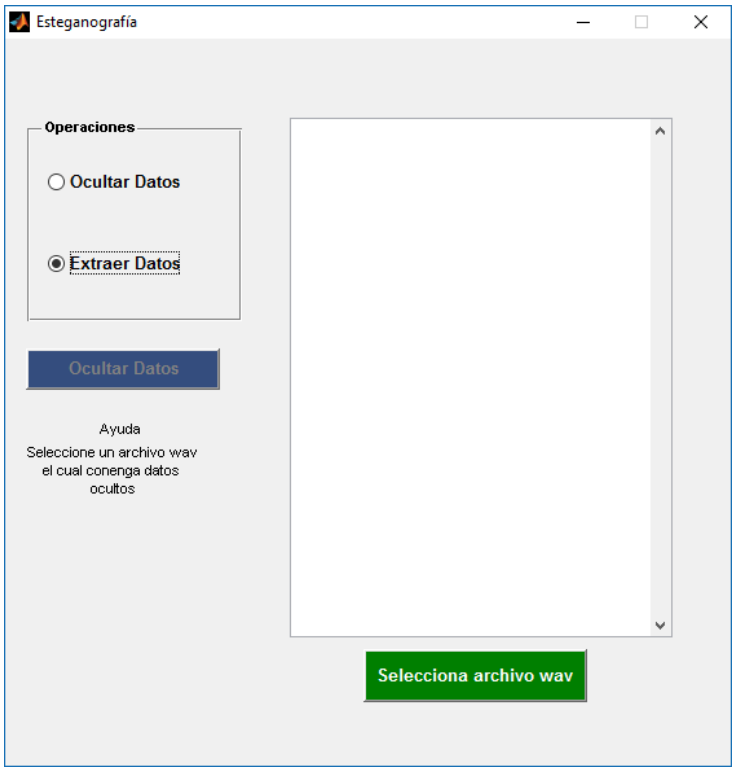

Figura 3.2.9 Extraer datos.

Una vez que seleccionamos esta operación proseguimos a seleccionar el archivo de audio a procesar, si este archivo no contiene “datos” se notificará al usuario con el aviso de la Figura 3.2.10.

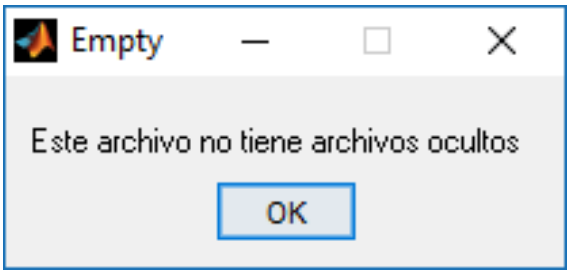

Figura 3.2.10 Archivo “vacío”

En caso contrario se mostrarán los datos que estaban ocultos en el archivo, esto se observa en la Figura 3.2.11.

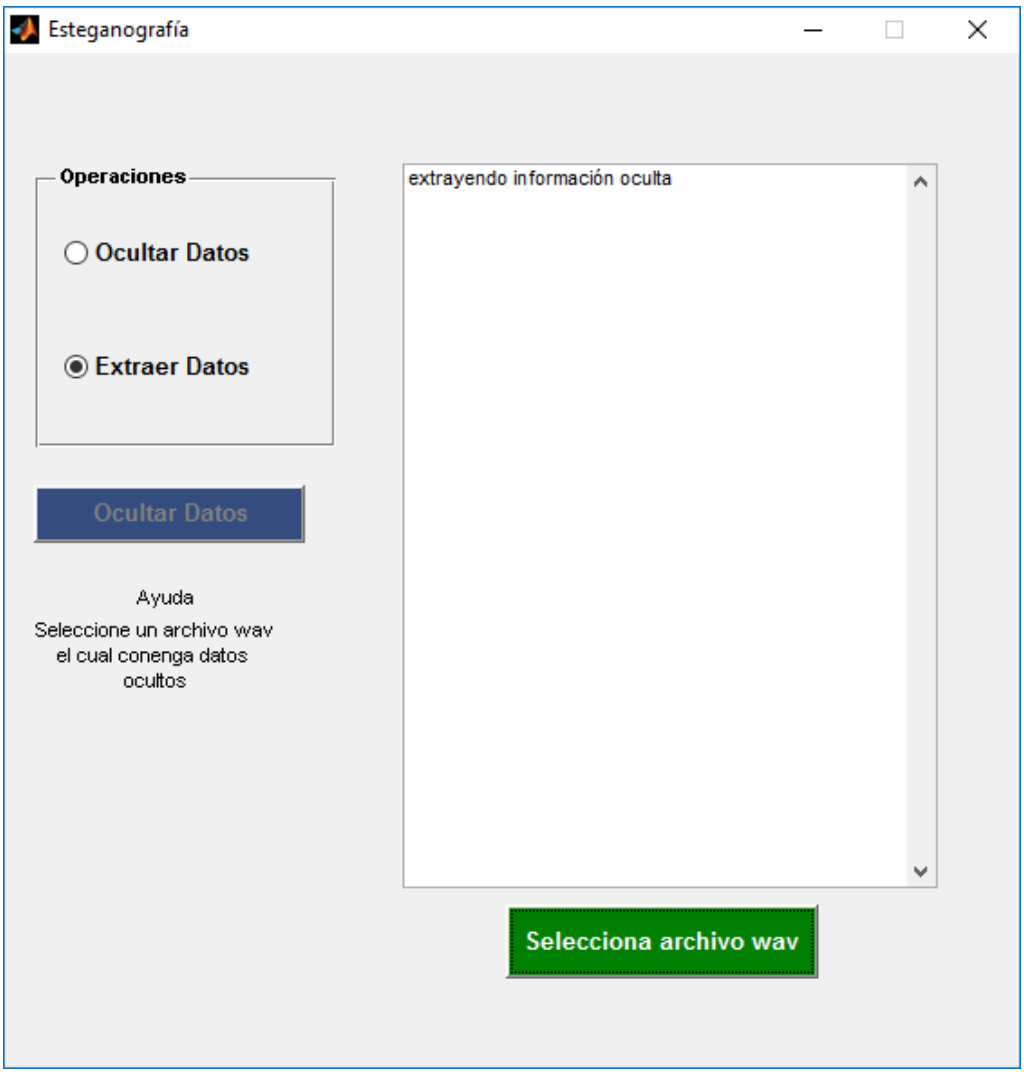

Figura 3.2.11. Mostrando datos ocultos.

En este proceso después de abrir el archivo y omitir los bytes de la cabecera se verifica si posee la identidad, de ser así se aplica LSB para obtener la información oculta, esto se muestra en la Figura 3.2.12.

```
fid1=fopen([pathname filename],'r');
header=fread(fid1,40,'uint8=>char');
data_size=fread(fid1,1,'uint32');
[dta,count]=fread(fid1,inf,'uint16');
fclose(fid1);
lsb=1;
identity=bitget(dta(1:8),lsb)';
if identity==[1 0 1 0 1 0 1 0]

  len_bin=zeros(20,1);
  m_bin=zeros(10,1);
  n_bin=zeros(10,1);

  m_bin(1:10)=bitget(dta(9:18),lsb);
  n_bin(1:10)=bitget(dta(19:28),lsb);
  m=bi2de(m_bin');
  n=bi2de(n_bin');
  len=m*n*8;
  secmsg_bin=zeros(len,1);
  secmsg_bin(1:len)=bitget(dta(29:28+len),lsb);
  secmsg_bin_re=reshape(secmsg_bin,len/8,8);
  secmsg_double=bi2de(secmsg_bin_re);
  secmsg=char(reshape(secmsg_double,m,n));
  set(handles.edit1,'string',secmsg);
```

Figura 3.2.12. Obteniendo datos ocultos (código)

CAPÍTULO IV

RESULTADOS

En el presente trabajo se logró esconder texto dentro de un archivo de audio, a través de la aplicación de un algoritmo esteganográfico, el archivo de audio sufrió mínimas modificaciones, ahora lo que resta es verificar si estas modificaciones afectaron de algún modo la calidad del audio, levantando sospechas sobre el contenido de este archivo.

Este análisis es necesario ya que como se menciono con anterioridad el objetivo principal de la esteganografía es ocultar información dentro de un objeto contenedor, de tal forma de que el contenido de dicho objeto pase desapercibido.

Para comprobar la eficiencia del sistema se ocultaron datos en distintos de archivos de audio, es decir se usará una canción instrumental, una pop y finalmente una canción de rock ó nü metal, del mismo modo se ocultaron diferentes cantidades de datos, éstos van desde un par de palabras, hasta un texto completo.

Lo anterior es para verificar que tanta información se puede ocultar sin afectar tanto la calidad del audio y por lo tanto saber en cuales “estilos” son más viables para usarlos como objetos contenedores.

Con el fin de llevar a cabo este análisis se realizó una pequeña modificación al sistema, esta consta de mostrar una ventana indicando el número de caracteres que se ocultaron durante el proceso, ésta ventana se puede observar en la Figura 4.1.

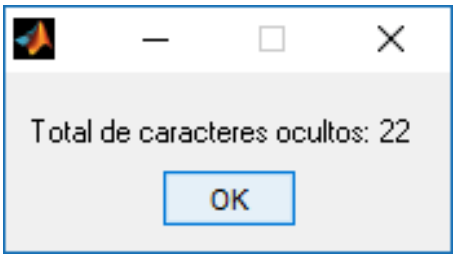

Figura 4.1.1 Total de caracteres ocultos.

4.2 ANÁLISIS DE RESULTADOS.

Los archivos de audio que se eligieron para la prueba se muestran en la Figura 4.2.1

Título	Artista	Género
O Sole Mio	David Garrett	Instrumental
Momentos	Noel Schajris	Pop
Before I Forget	Slipknot	Nü Metal

Figura 4.2.1. Listado de archivos de audio.

Como se puede observar, estas canciones van desde la más tranquila hasta la mas "ruidosa".

Las pruebas se realizaron con diferente cantidad de información, la mínima fue de 800 caracteres y la máxima de 9429 caracteres. En estas pruebas se busca medir la distorsión que sufre el archivo de audio después de aplicarle el algoritmo LSB, para ello los resultados se medirán de la siguiente manera.

A cada archivo de audio se le ocultará la misma cantidad de caracteres y después se escuchará dicho archivo comparándolo con el original, con el objetivo de encontrar un cambio (o distorsión) en el archivo procesado, anotando qué diferencias hay entre los 2 archivos, esta diferencia (en caso de que exista) se medirá en 3 niveles: Ninguna, Apenas perceptible y muy perceptible.

Del mismo modo de ambos archivos de audio (original y con información oculta) se obtendrán graficas en dominio de la frecuencia para un mejor análisis, para ello es necesario determinar un rango de frecuencias, en esta ocasión se utilizará un rango entre 100 y 3,000 Hz debido a que es el rango de frecuencias en las cuales las ondas sonoras son perfectamente audibles. Primero se obtendrá una gráfica por separado del audio original y del que contiene datos ocultos, finalmente se graficarán ambos archivos al mismo tiempo para verificar la existencia de discrepancias entre ambas, para este proceso se usarán las funciones de matlab.

En la siguiente tabla se muestran los resultados obtenidos con la primera prueba, en la cual se ocultaron 800 caracteres.

Audio	Cant. Caracteres	Distorsión
O Sole Mio	800	Ninguna
Momentos	800	Ninguna
Before I Forget	800	Ninguna

Figura 4.2.2 Resultados de primera prueba.

Procederemos a analizar la primera canción. Primero graficaremos ambos archivos, los resultados de pueden observar en la Figura 4.2.3 podemos observar la gráfica de frecuencias de O Sole Mio.

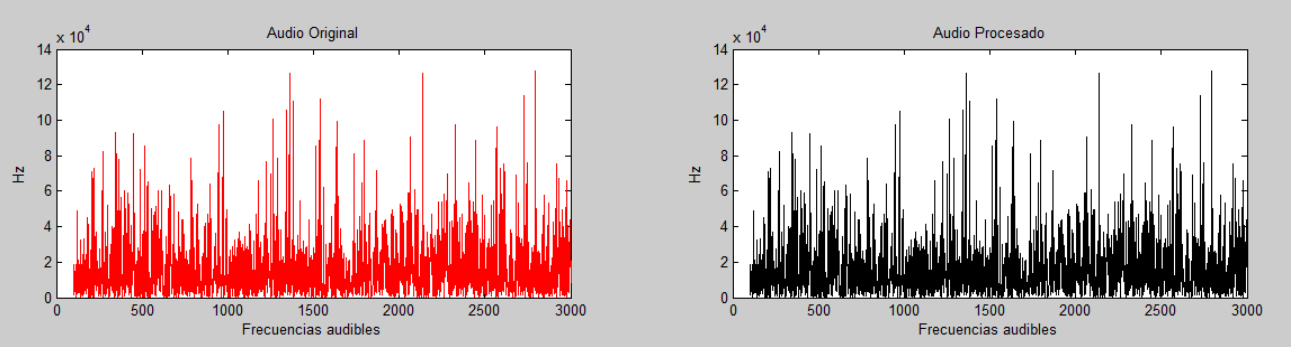

Figura 4.2.3 Audios O Sole Mio.

Como podemos observar no hay diferencia alguna entre ambas gráficas, pero para constatarlo sobrepondremos ambas, el resultado se aprecia en la Figura 4.2.4.

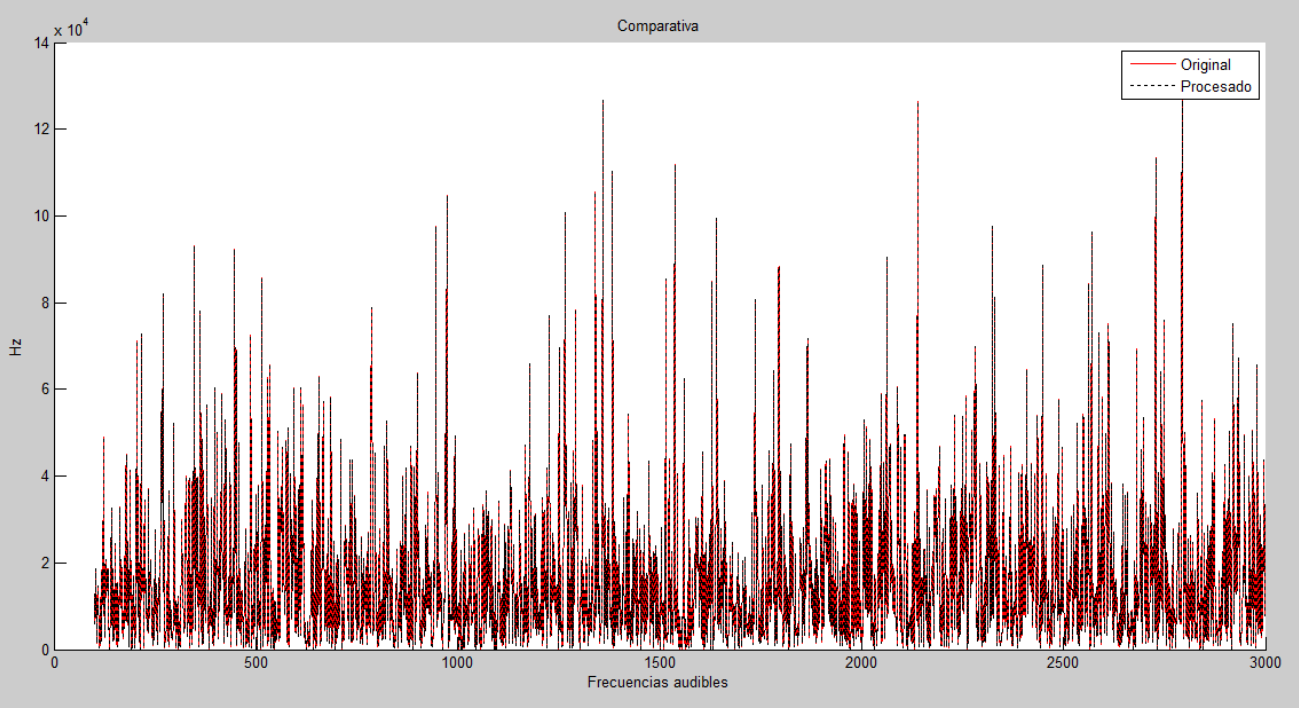

Figura 4.2.4 Comparativa de O Sole Mio.

Ciertamente se reafirma el hecho de que no hay diferencia entre ambos audios.

Enseguida analizaremos la canción de Momentos, se procede a graficar ambos archivos, los resultados se muestran en la Figura 4.2.5.

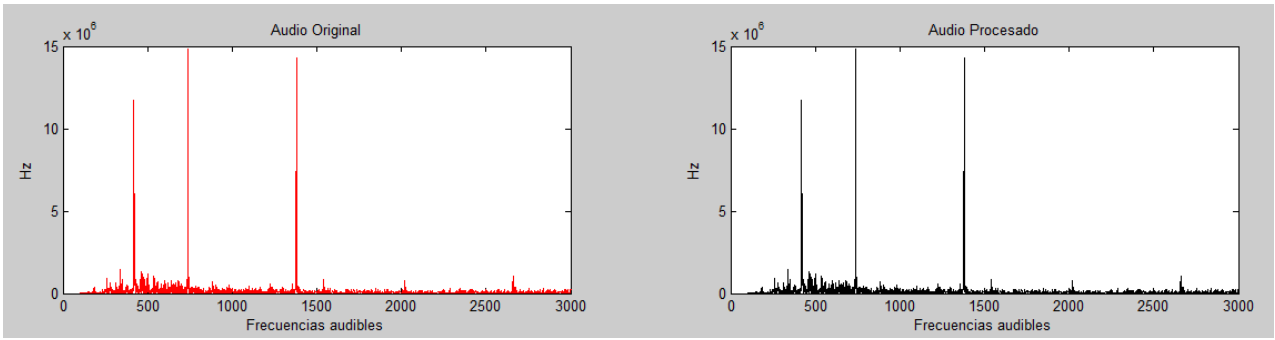

Figura 4.2.5 Audios de Momentos

Igual que con la canción anterior no se observan diferencias algunas, para comprobarlo se sobrepondrán ambas gráficas, el resultado se aprecia en la Figura 4.2.6.

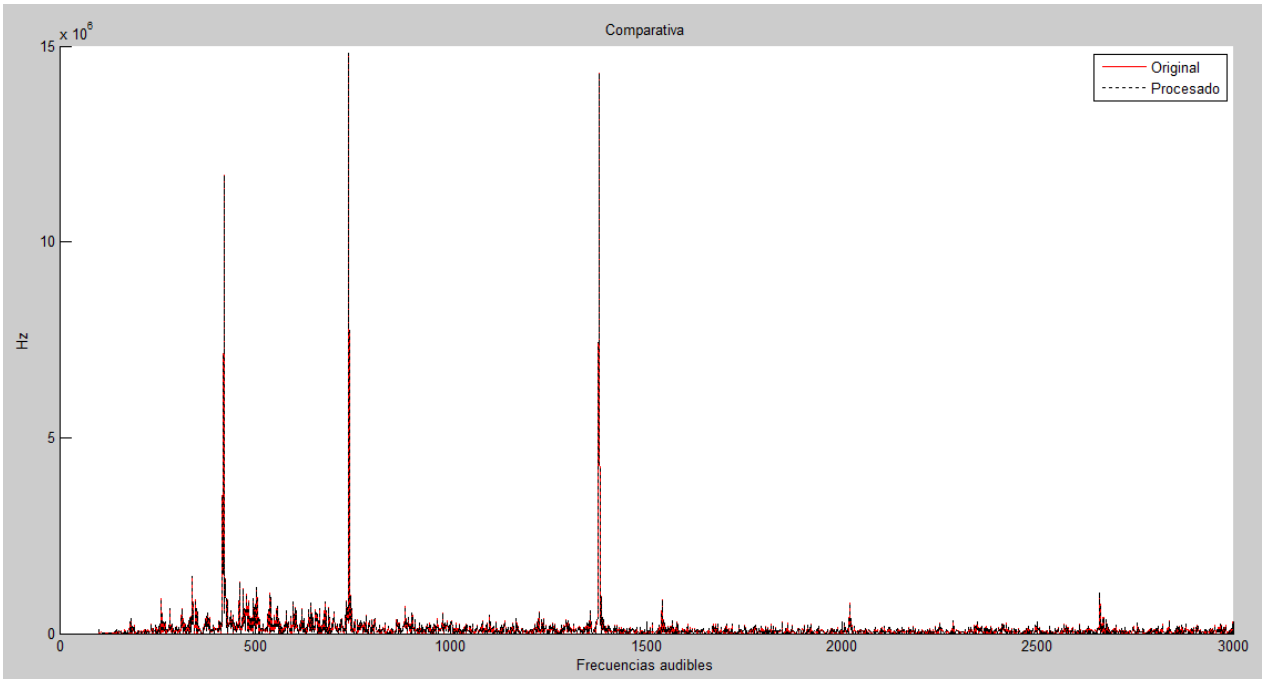

Figura 4.2.6 Comparativa de Momentos.

Podemos observar que no hay cambio alguno en la calidad del audio, Finalmente analicemos la última canción Before I Forget, sus gráficas se muestran en la Figura 4.2.7.

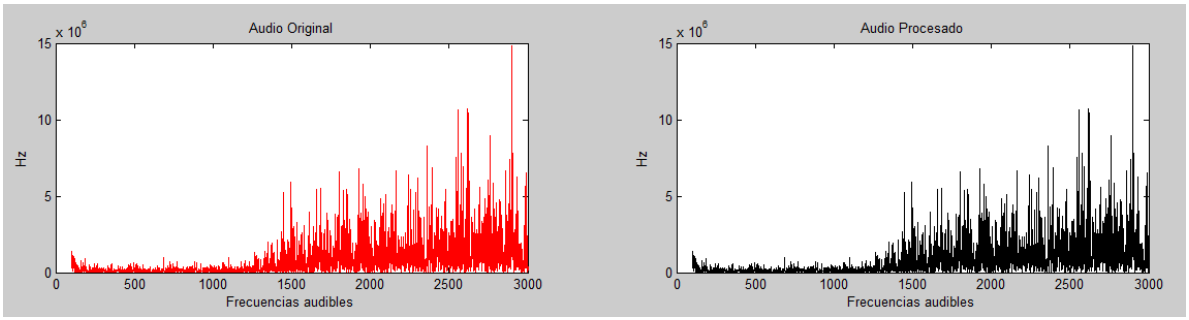

Figura 4.2.7 Audios Before I Forget.

Una vez mas no existen diferencias entre estos archivos, aún así se comprobará el resultado se muestra en la Figura 4.2.8.

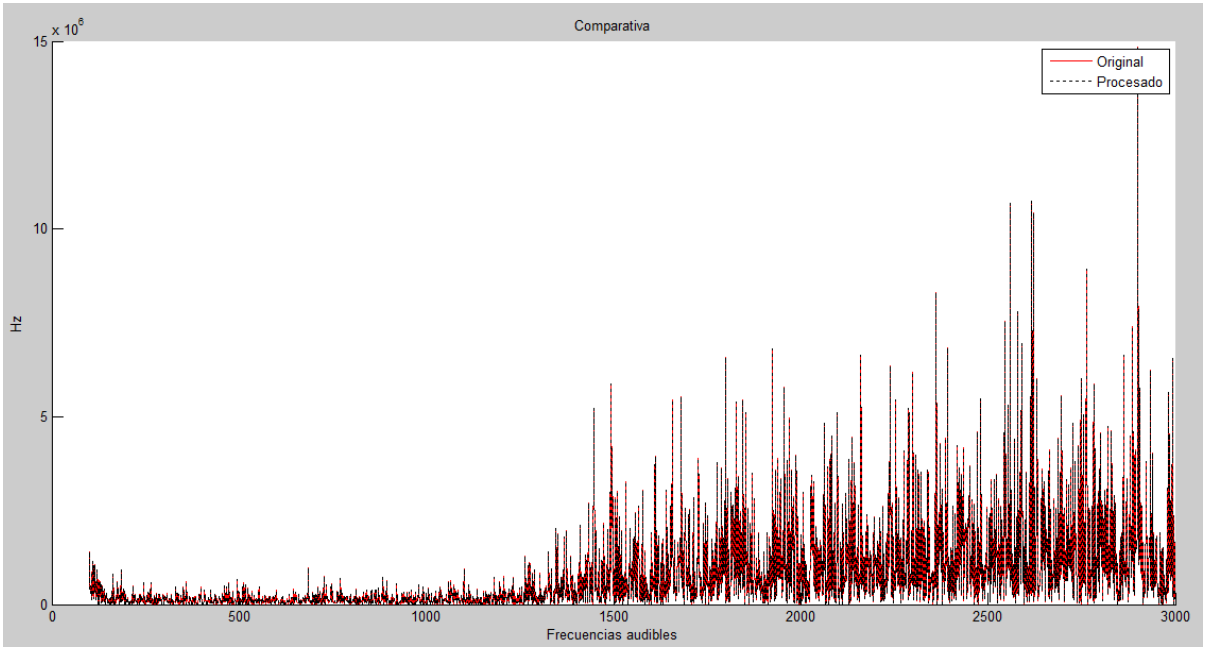

Figura 4.2.8 Comparativa Before I Forget.

Se comprueba que no hay diferencias entre estos archivos.

Sigamos con la segunda prueba, en la cual se aumentará en número de caracteres a 2000, procederemos a ocultar el texto y escuchar los audios procesados para verificar si existe alguna discrepancia, los resultados se muestran en la Figura 4.2.9.

Audio	Cant. Caracteres	Distorsión
O Sole Mio	2000	Ninguna
Momentos	2000	Ninguna
Before I Forget	2000	Ninguna

Figura 4.2.9 Tabla de resultados con 2000 caracteres.

A continuación se compararán las gráficas de los audios correspondientes para verificar de una manera más concisa si hay diferencias entre el audio original y el que contiene texto oculto. En la figura 4.2.10 se muestran las gráficas por separado del archivo O Sole Mio.

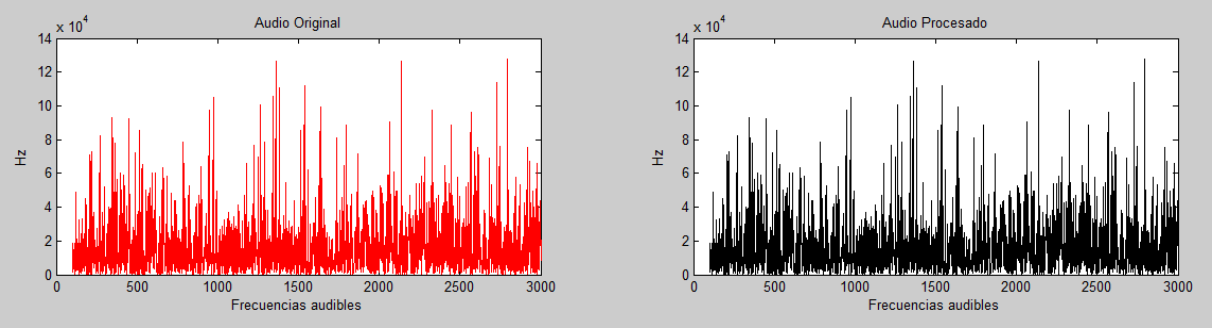

Figura 4.2.10 O Sole mio con 2000 caracteres.

A simple vista parece que el texto oculto no causó ninguna alteración, para constatarlo compararemos ambas gráficas, el resultado se observa en la Figura 4.2.11.

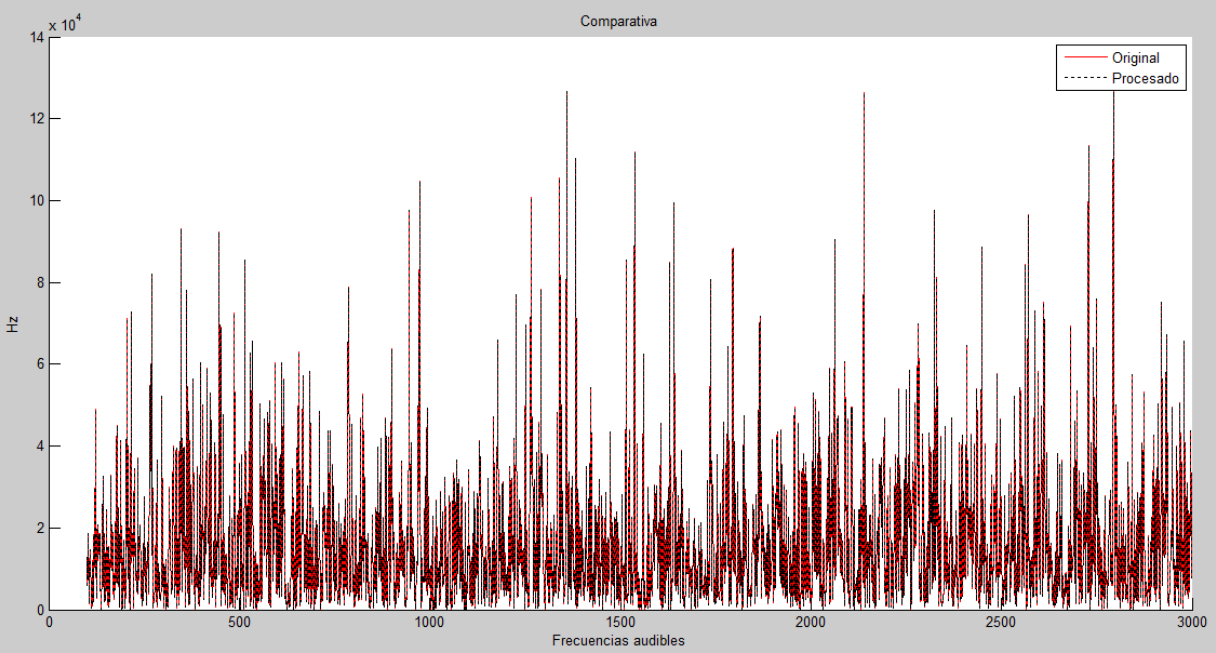

Figura 4.2.11 Comparativa O Sole Mio con 2000 caracteres.

Esta comparativa (Figura 4.5.10) nos reafirma el hecho de que no hay diferencia entre ambos archivos, prosigamos con la prueba y en esta ocasión utilizaremos el archivo "momentos". En la Figura 4.2.12 Observamos los resultados.

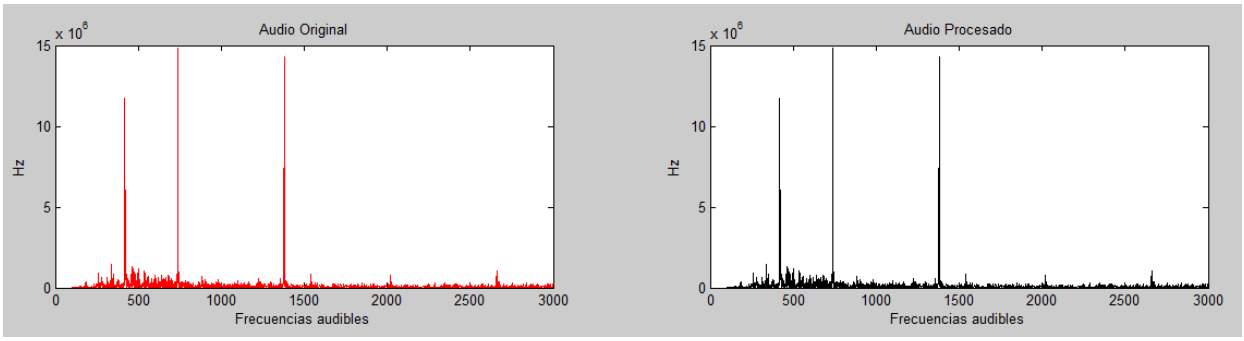

Figura 4.2.12 Momentos con 2000 caracteres.

Al parecer sigue sin haber afectaciones a la calidad de audio, verifiquémoslo comparando ambas gráficas, el resultado se muestra en la Figura 4.2.13.

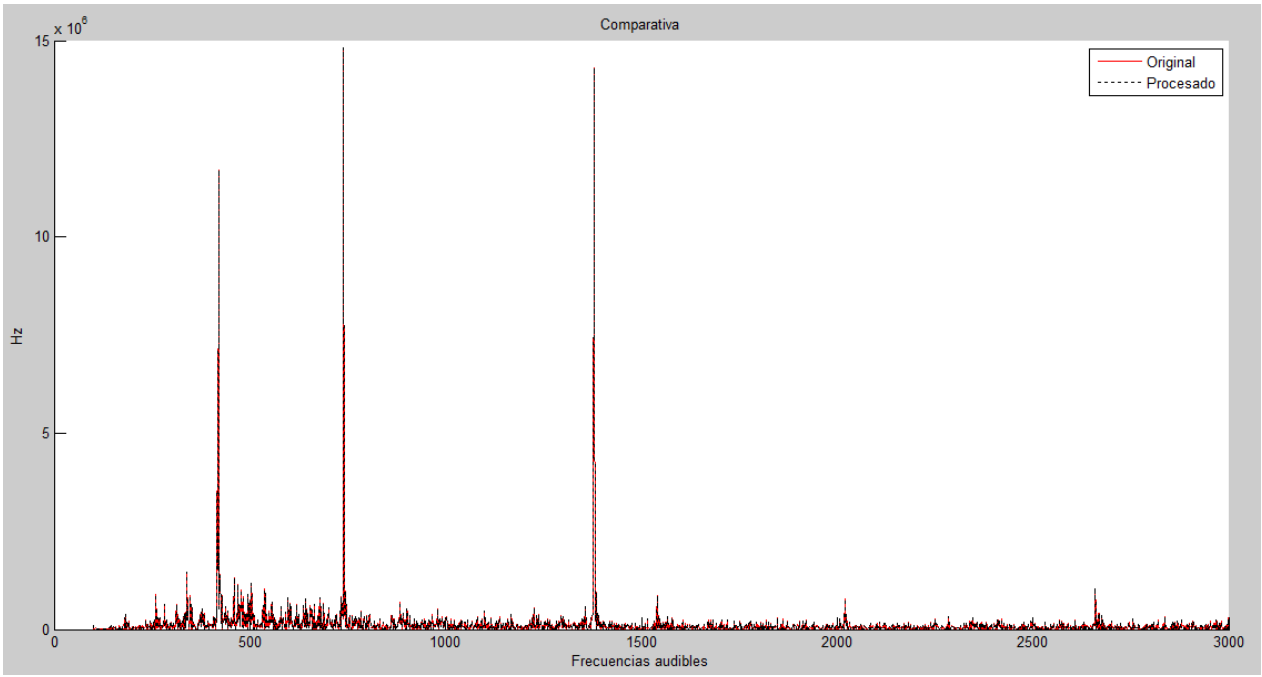

Figura 4.2.13. Comparativa de momentos con 2000 caracteres.

Como podemos observar el audio sigue sin sufrir alteración alguna, Pasemos a la parte final de esta prueba, analicemos el archivo Before I Forget, las gráficas resultantes se pueden contemplar en la Figura 4.2.14.

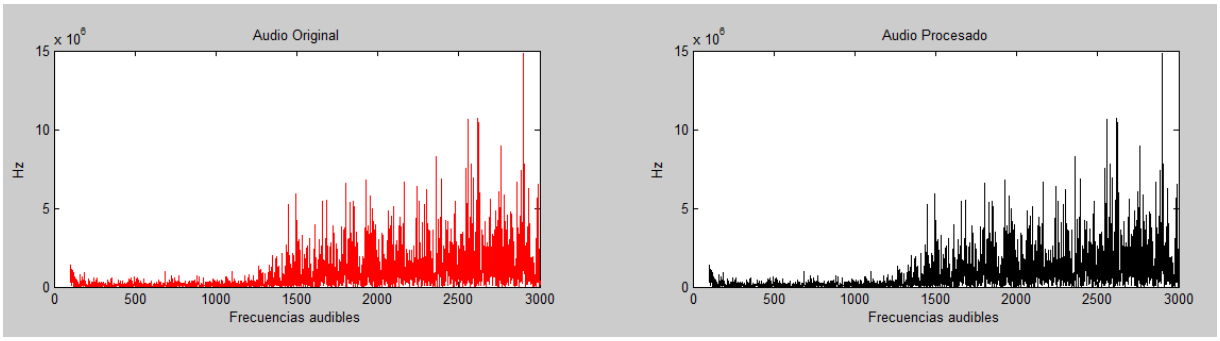

Figura 4.2.14. Before I Forget con 2000 caracteres.

Sigue sin haber cambio alguno, para verificar esto se compararán ambos gráficos, los resultados se pueden analizar en la Figura 4.2.15.

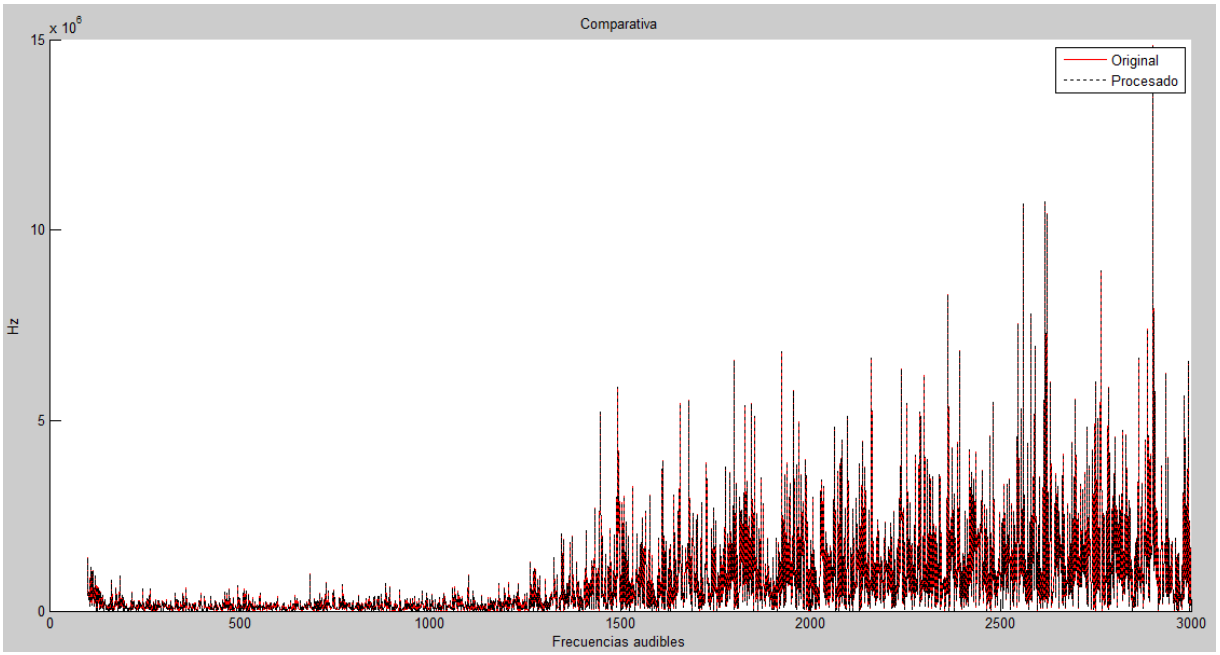

Figura 4.2.15. Comparativa de Before I Forget.

Analizando dicha Figura nos percatamos de que no existe cambio alguno. Es momento de pasar a la prueba final, en esta se ocultarán 9429 caracteres. Como ya es costumbre primero se escucharán las canciones para intentar percibir algún cambio en ellas, la tabla de resultado se muestra en la Figura 4.2.16.

Audio	Cant. Caracteres	Distorsión
O Sole Mio	9429	Ninguna
Momentos	9429	Ninguna
Before I Forget	9429	Ninguna

Figura 4.2.16 Tabla de resultados final.

Al escuchar los distintos archivos de audio no se logró encontrar alteración alguna a pesar del gran número de caracteres ocultos en ellos, sin embargo se procederá a analizar cada archivo como se ha hecho con anterioridad.

Primero se analizó la canción O Sole Mio, los resultados se observan en la Figura 4.2.17.

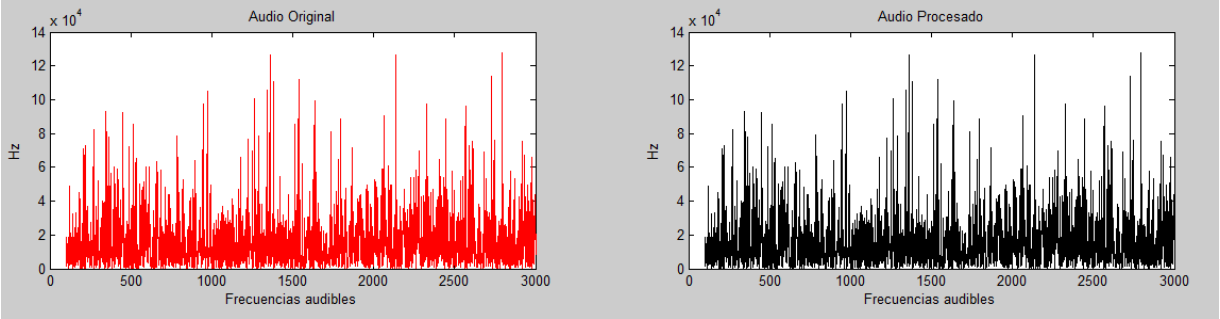

Figura 4.2.17 O Sole Mio 9429 caracteres.

Al observar ambas gráficas podemos ver que no hay diferencia entre éstas, pero para comprobarlo compararemos ambos gráficas, los resultados se pueden observar en la Figura 4.2.18.

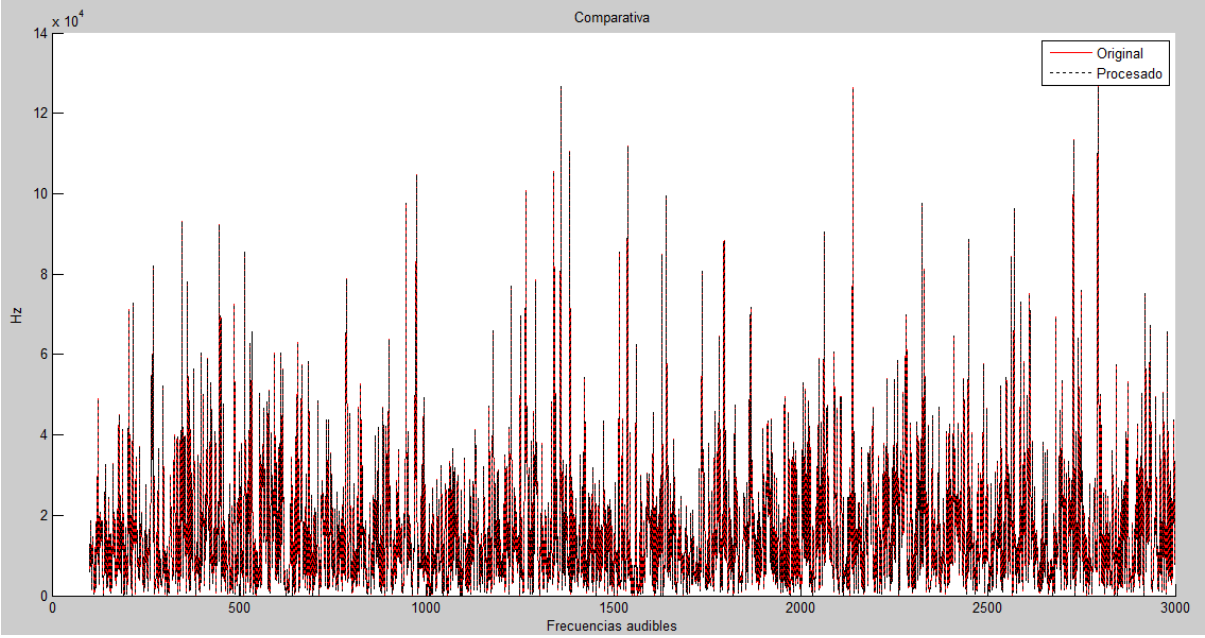

Figura 4.2.18 Comparativa O Sole Mio con 9429 caracteres.

Y a pesar de la gran cantidad de información oculta no hay distorsión alguna, a continuación se analizará la canción momentos, los gráficos se muestran en la Figura 4.2.19.

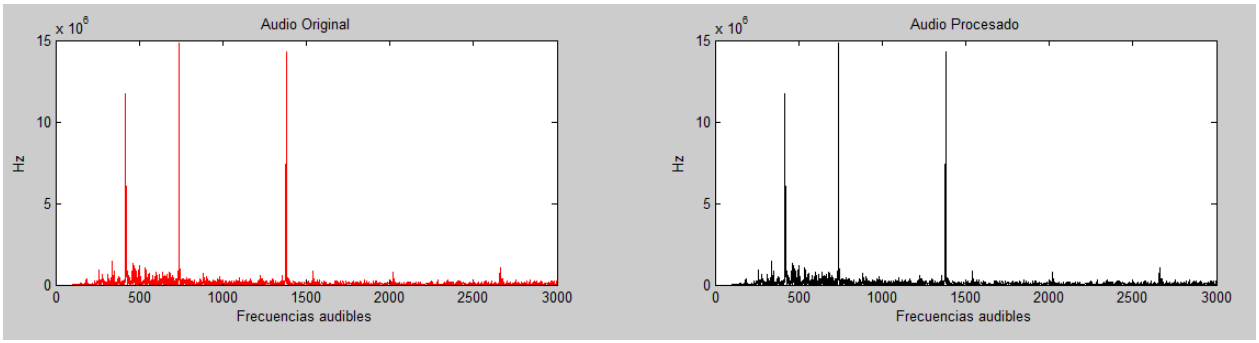

Figura 4.2.19 Momentos con 9429 caracteres.

Al parecer no existe diferencia alguna entre ambos archivos, para estar más seguros vamos a comparar ambas gráficas, el resultado se puede analizar en la Figura 4.2.20.

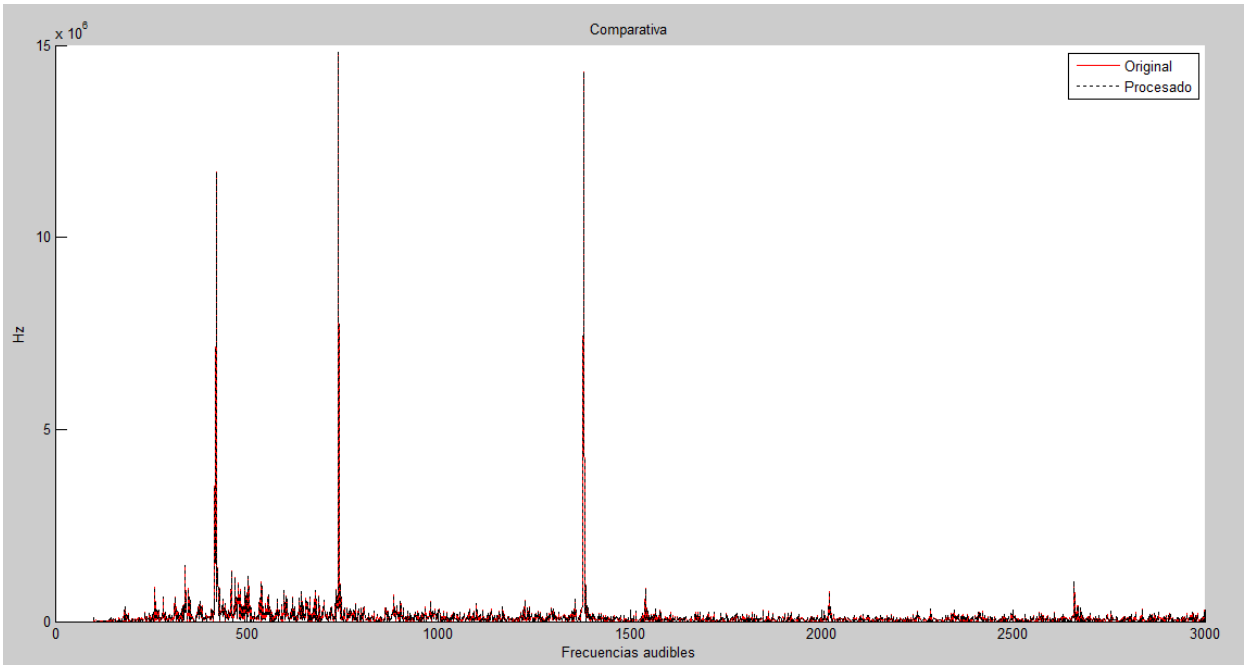

Figura 4.2.20 Comparativa momentos con 9429 caracteres.

Al observar esta comparación podemos darnos cuenta de que sigue sin existir diferencia alguna entre ambos archivos.

Finalmente analizaremos la canción Before I Forget, los gráficos resultantes se muestran en la Figura 4.2.21.

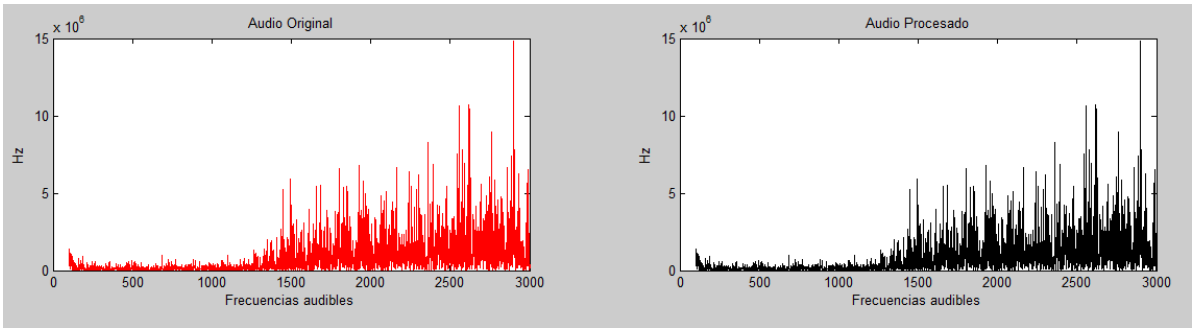

Figura 4.2.21. Before I Forget con 9429 caracteres.

En lo gráficos de la Figura 4.5.21 no se observa alguna diferencia, pero se compararan ambos para estar más seguros, los resultados se observan en la Figura 4.2.22.

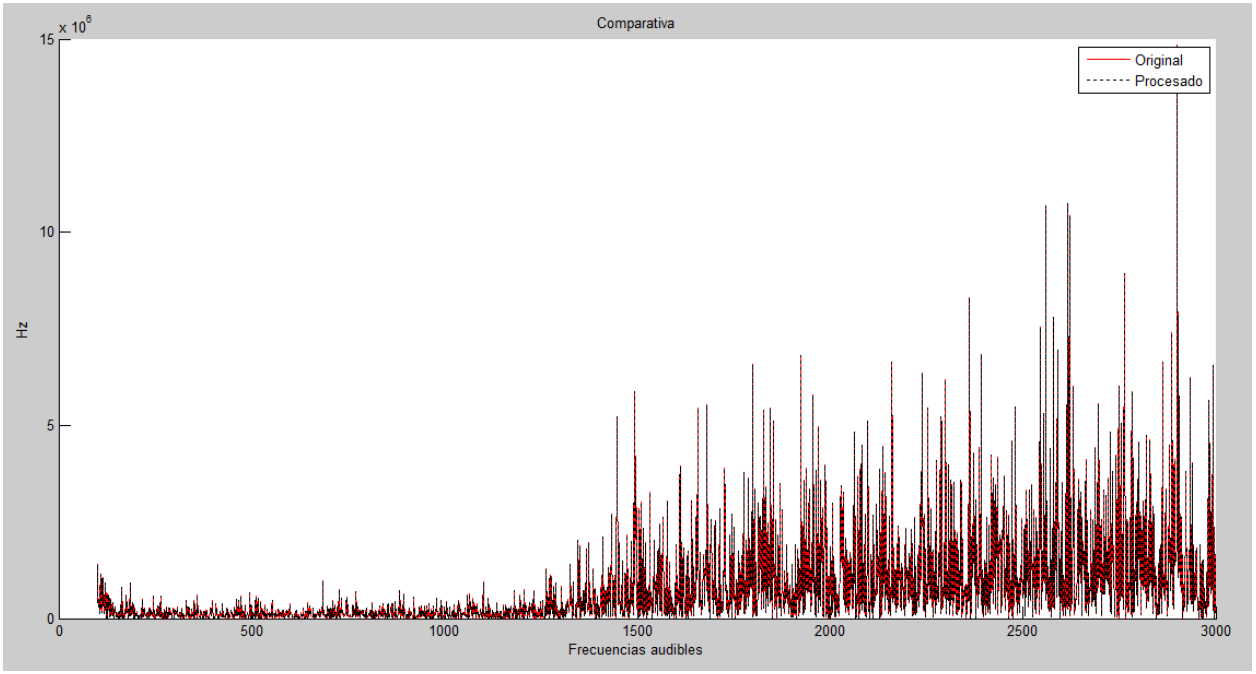

Figura 4.2.22 Comparativa Before I Forget con 9429 caracteres.

Al analizar la Figura 4.2.22 posemos observar que efectivamente no existe deferencia alguna entre ambos archivos a pesar de la gran información que se ocultó.

CAPÍTULO V

CONCLUSIONES.

Como se ha observado al ocultar texto dentro de un archivo de audio, este último no sufre alteraciones en la calidad del audio en un rango de 800 a 9429 caracteres, la única forma en la que el contenido pueda alterarse es alterando de algún modo el objeto contenedor.

Del mismo modo seleccionando tres canciones específicas podemos constatar que cualquier archivo de audio es un buen candidato para usarse como objeto contenedor.

Finalmente al extraer el texto oculto este se extrae casi de manera perfecta a excepción de aquellas letras que contienen caracteres especiales como lo son: acentos, diéresis y tildes.

La esteganografía es un método efectivo para ocultar información, el cual se ha usado a través de la historia, hay varias razones por las cuales se desea ocultar información, algunas de ellas son:

- Comunicación confidencial y almacenamientos de datos secretos.
- Protección de la alteración de datos.
- Sistema de control de acceso para la distribución de contenidos digitales.

Actualmente la esteganografía esta siendo enfocada a la protección de derechos de autor, un ejemplo de ello es lo que ocurre con i-Tunes. Cuando un usuario compra música a través de su cuenta, ésta solo puede reproducirse con los datos de la cuenta, si potro usuario intenta reproducir el audio primero tendrá que introducir el login y la contraseña del usuario que la descargo, de otra manera es imposible hacerlo.

Otro enfoque que tiene la esteganografía es la firma digital de archivos, esto con el fin de verificar la autenticidad de los mismos, al mismo tiempos estas firmas digitales pueden ser usadas para rastrear el origen de algún dispositivo electrónico, es decir de una fotografía se podría obtener el número de serie del dispositivo que la tomó, al ubicar este dispositivo se pueden saber más datos como ¿quién? y ¿en dónde? Obtuvo dicho dispositivo.

Por otra parte la esteganografía puede ser usada para espionaje industrial, utilizándola como medio para filtrar información sensible sobre empresas, productos y/o procesos.

REFERENCIAS.

1) Gunjan Nehru, Puja Dhar, *A Detailed look of Audio Steganography Techniques using LSB and Genetic Algorithm Approach*, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 2, January 2012, ISSN (Online): 1694-0814
2) Jesús Díaz Vico, *Esteganografía y Estegoanálisis: Ocultación de datos en Stream de Audio Vorbis*, Facultad De Informática, Universidad Politécnica De Madrid, Septiembre 2010
3) K.P.Adhiya, Swati A. Patil, *Hiding Text in Audio Using LSB Based Steganography*, Information and Knowledge Management, Vol 2, No.3, 2012, ISSN 2224-5758 (Paper), ISSN 2224-896X (Online)
4) Masoud Afrakhteh, *Steganography Based On Utilizing More Surrounding Pixels*, Faculty of Computer Science and Information Systems Universiti Teknologi Malaysia, March 2010
5) Da-Chun Wu, Wen-Hsiang Tsai, *A steganographic method for images by pixel-value differencing*, Pattern Recognition Letters 24 (2003) 1613–1626
6) Swati Malviya, Manish Saxena2, Dr. Anubhuti Khare, *Audio Steganography by Different Method*, International Journal of Emerging Technology and Advanced Engineering, Volume 2, Issue 7, July 2012, ISSN 2250-2459.
7) Jayaram P, Ranganatha H R, Anupama H S, *Information Hiding Using Audio Steganography – A Survey*, The International Journal of Multimedia & Its Applications (IJMA) Vol.3, No.3, August 2011.
8) Blanca Esther Carvajal Gámez, *Técnica de Inserción de Información en Video aprovechando el mismo ancho de banda*, Instituto Politecnico Nacional, Mayo 2008…
9) Bárbara Emma Sánchez Rinza, María del Rocio Guadalupe Morales Salgado, Cristian Omar Cortez Olguín, *Security system for sending information containing hidden voice data by steganography (SIOVE) using MatLab*, International Journal of Engineering and Innovative Technology (IJEIT) Volume 5, Issue 1, July 2015, ISSN: 2277-3754.