Ocultar datos en archivos de sonido

Juan Antonio Cano Salado Borja Moreno Fernández
Pascual Javier Ruiz Benítez

31 de mayo de 2011

Índice

Índice **2**

1. Resumen **4**

2. Introducción **5**
2.1. Esteganografía 5
2.2. Audio digital 7
2.2.1. El formato WAV 7

3. Descripción del problema **12**
3.1. Privacidad 12
3.2. Protección de copyright 12

4. Soluciones presentadas **15**
4.1. Codificación en el bit menos significativo (LSB coding) 15
4.2. Codificación en paridad (Parity coding) 17
4.3. Codificación en fase (Phase coding) 19
4.4. Espectro ensanchado (Spread spectrum) 21
4.5. Ocultamiento en eco (Echo hiding) 23

5. Implementación realizada **26**
5.1. Manual de uso 26
5.1.1. Ocultación en LSB 27
5.1.2. Extracción en LSB 28
5.1.3. Ocultación en Parity Coding 28
5.1.4. Extracción en Parity Coding 30
5.2. Mejoras y ampliaciones 30
5.3. Otras implementaciones 31

6. Problemas abiertos **33**

7. Conclusiones **34**

8. Tabla de tiempos **35**

Bibliografía **36**

1. Resumen

Este trabajo aborda la ocultación de datos en archivos de sonido o, lo que es lo mismo, la esteganografía de audio. El objetivo de esta disciplina consiste básicamente en ocultar información (de cualquier tipo) en archivos de sonido, de forma que los cambios llevados a cabo en el archivo original resulten imperceptibles para una persona.

Comenzaremos estudiando los principios básicos de la esteganografía y el audio digital, con especial énfasis en uno de los formatos de audio más populares: WAV. Hecho esto, presentaremos el problema de la esteganografía de audio y sus aplicaciones más destacadas. A continuación, analizaremos algunos de los métodos más comúnmente utilizados en esteganografía de audio. Completaremos esta información con unas conclusiones y una exposición de los problemas que quedan abiertos. Finalmente, incluiremos una descripción de la implementación realizada y proporcionaremos un manual de instalación y manejo de la aplicación desarrollada.

2. Introducción

2.1. Esteganografía

La esteganografía es la disciplina en la que se estudian y aplican técnicas que permiten el ocultamiento de mensajes u objetos dentro de otros, llamados portadores, de modo que no se perciba su existencia.

Los orígenes de la esteganografía datan de la antigua Grecia. Heródoto, famoso historiador griego, informa del uso de la esteganografía en informes de Grecia a Persia. El método consistía en afeitar la cabeza de un esclavo y tatuar un mensaje en la cabeza rapada. Dicho mensaje quedaba oculto cuando el pelo del esclavo volvía a crecer. Para leer el mensaje sólo era necesario volver a afeitarle la cabeza al esclavo. La idea clave en la que residía la seguridad del método era que nadie sospechase de la existencia misma del mensaje.

La esteganografía ha sido usada en numerosas ocasiones a lo largo de la historia: tintas invisibles, acrósticos o mensajes microscópicos son sólo algunas de sus formas. Recientemente, en plena era digital, la estaganografía ha adquirido gran importancia como tecnología utilizada en el campo de la seguridad informática. Es posible ocultar mensajes secretos en correos electrónicos, imágenes, audio e incluso vídeo.

Diversos grupos han mostrado tener un gran interés en las aplicaciones de la esteganografía. Algunos, interesados en la protección de derechos de autor. Otros, preocupados por proteger la privacidad de sus mensajes. Muchos gobiernos temen que la esteganografía podría convertirse en una herramienta de gran utilidad para criminales y grupos terroristas. En cualquier caso, parece claro que existen multitud de aplicaciones y usos para la esteganografía, y la mayoría de los expertos están de acuerdo en que será un tema de gran interés durante los próximos años.

La figura 1 ilustra un ejemplo simple de un problema en el que el uso de la esteganografía puede ser de utilidad. Dos prisioneros en una cárcel necesitan comunicarse para ultimar los detalles de un plan de fuga, pero tienen un gran problema: el guardia de la prisión tiene acceso a toda la correspondencia entre los prisioneros. La criptografía por sí sola no es una solución. Un mensaje cifrado levantaría todo tipo de sospechas. Los prisioneros deben idear un sistema que les permita pasar mensajes de apariencia inocente, con información oculta que sólo ellos puedan entender. Esto es exactamente lo que persigue la esteganografía.

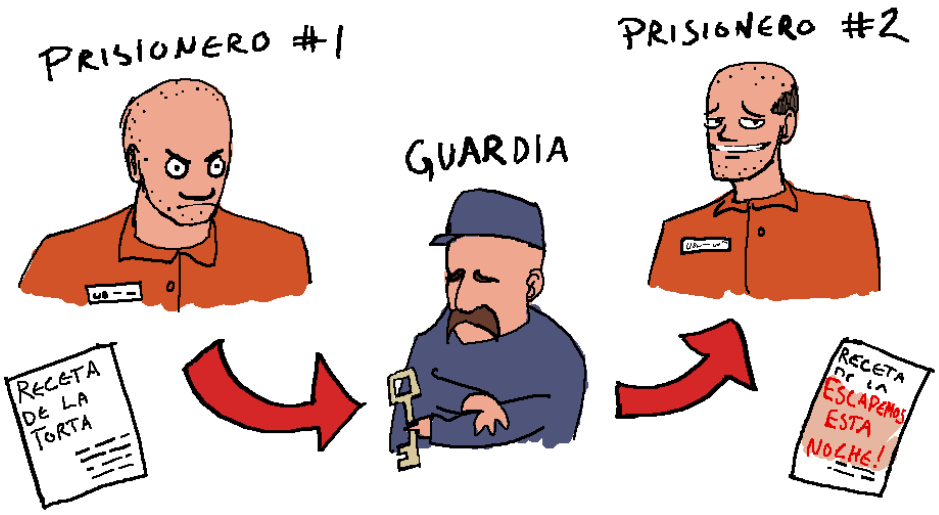

Figura 1: Problema de los prisioneros

La esteganografía puede además combinarse con la criptografía para crear sistemas más seguros. Así, distinguimos:

- Esteganografía pura

  La fortaleza del sistema recae en los algoritmos de ocultación y extracción de la información, que solo el emisor y el receptor del mensaje deberían conocer.

- Esteganografía de clave privada

  Fruto de la combinación de esteganografía pura con criptosistemas simétricos. Se asume que un atacante podría conocer los algoritmos de ocultación y extracción de la información. Por este motivo, el mensaje se cifra utilizando un cifrado simétrico antes de ocultarlo. De esta manera, incluso si el atacante intercepta la transmisión y logra extraer la información aún tendrá que enfrentarse al criptoanálisis del criptosistema utilizado.

- Esteganografía de clave pública

Basada en unir esteganografía pura y criptosistemas de clave pública. De esta manera, el emisor y el receptor evitan tener que compartir una clave privada.

2.2. Audio digital

El audio digital se diferencia del sonido analógico tradicional en que es una señal discreta en lugar de una señal continua. Esta señal discreta es creada llevando a cabo un proceso de muestreo y cuantización de una señal analógica continua. La frecuencia de muestro varía según los propósitos. Por ejemplo, la frecuencia de muestreo estándar para un CD de audio digital es de alrededor de 44kHz. La figura 2 ilustra el proceso de muestro y cuantización para producir una señal de audio digital a partir de una señal continua de audio analógico. En dicha figura se ha exagerado la naturaleza discreta de la señal digital. Sin embargo, las frecuencias de muestreo usuales permiten que las diferencias entre el audio digital y la señal analógica original sean mínimas.

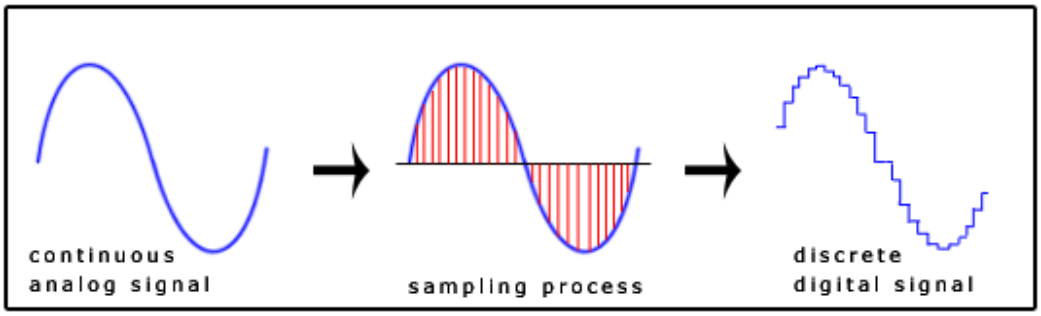

Figura 2: Audio digital

Los archivos de audio digital se almacenan en el ordenador como una secuencia de ceros y unos. Con las herramientas apropiadas, es posible alterar individualmente los bits que componen estos archivos. Esto hará posible llevar a cabo modificaciones a la secuencia binaria que produzcan cambios en la señal de audio que no sean perceptibles al oído humano.

2.2.1. El formato WAV

WAV (o WAVE), apócope de WAVEform audio file format, es un formato de audio digital normalmente sin compresión de datos desarrollado y pro-

piedad de Microsoft y de IBM que se utiliza para almacenar sonidos en el PC. Admite archivos mono y estéreo a diversas resoluciones y velocidades de muestreo. Su extensión es .wav.

Es una variante del formato RIFF (Resource Interchange File Format, formato de fichero para intercambio de recursos). El formato toma en cuenta algunas peculiaridades de la CPU Intel, y es el formato principal usado por Windows. A pesar de que el formato WAV es compatible con casi cualquier códec de audio, se utiliza principalmente con el formato PCM (no comprimido) y, al no tener pérdida de calidad, es adecuado para uso profesional. Para tener calidad CD de audio se necesita que el sonido se grabe a 44100 Hz y a 16 bits. Por cada minuto de grabación de sonido se consumen unos 10 megabytes de espacio en disco. Una de sus grandes limitaciones es que solo se pueden grabar archivos de 4 gigabytes como máximo, lo cual equivale aproximadamente a 6,6 horas en calidad de CD de audio. Es una limitación propia del formato, independientemente de que el sistema operativo donde se utilice sea MS Windows u otro distinto, y se debe a que en la cabecera del fichero se indica la longitud del mismo con un número entero de 32 bits.

En Internet no es popular, fundamentalmente porque los archivos sin compresión son muy grandes. Son más frecuentes los formatos comprimidos con pérdida, como el MP3 o el Ogg Vorbis. Como éstos son más pequeños, la transferencia a través de Internet es mucho más rápida. Además, existen códecs de compresión sin pérdida más eficaces, como Apple Lossless o FLAC.

Estudiaremos este formato debido a su sencillez. Será además el formato empleado por la aplicación desarrollada (véase sección "Implementación realizada").

Los archivos RIFF comienzan con una cabecera seguida de una secuencia de chunks de datos (fragmentos de información). Los archivos WAVE son archivos RIFF con un único chunk llamado "WAVE" que se divide en dos sub-chunks:

- fmt

  Especifica el formato de los datos.

- data

  Contiene la información de sonido dividida en samples (porciones de audio).

La figura 3 muestra gráficamente la organización de un archivo WAV de audio digital. Podemos observar la división en tres partes claramente diferenciadas (RIFF, fmt y data) así como los campos que componen cada una de ellas (incluyendo tamaño, nombre, posición relativa en el archivo, etc.). En conjunto, la cabecera del archivo (compuesta por RIFF, fmt, y parte de data) tiene un tamaño de 44 bytes. Tras esta cabecera se encontraría la información de sonido real. A continuación ofrecemos una pequeña descripción de cada uno de los campos que componen esta cabecera.

- ChunkID

  Contiene las letras "RIFF" en formato ASCII.

- ChunkSize

  36 + SubChunk2Size, o, más en concreto: 4 + (8 + SubChunk1Size) + (8 + SubChunk2Size)

  Es el tamaño del archivo a partir de este punto. Es decir, el tamaño total del archivo menos los 8 bytes ya tratados (campos ChunkID y ChunkSize).

- Format

  Contiene las letras "WAVE".

- Subchunk1ID

  Contiene las letras "fmt ".

- Subchunk1Size

  16 en caso de usar PCM. Es el tamaño del resto de la sección "fmt".

- AudioFormat

  1 para PCM. Otros valores para otros tipos de compresión.

- NumChannels

  Número de canales. Mono = 1, Stereo = 2, etc.

- SampleRate

  Frecuencia de muestreo. 8000, 44100, etc.

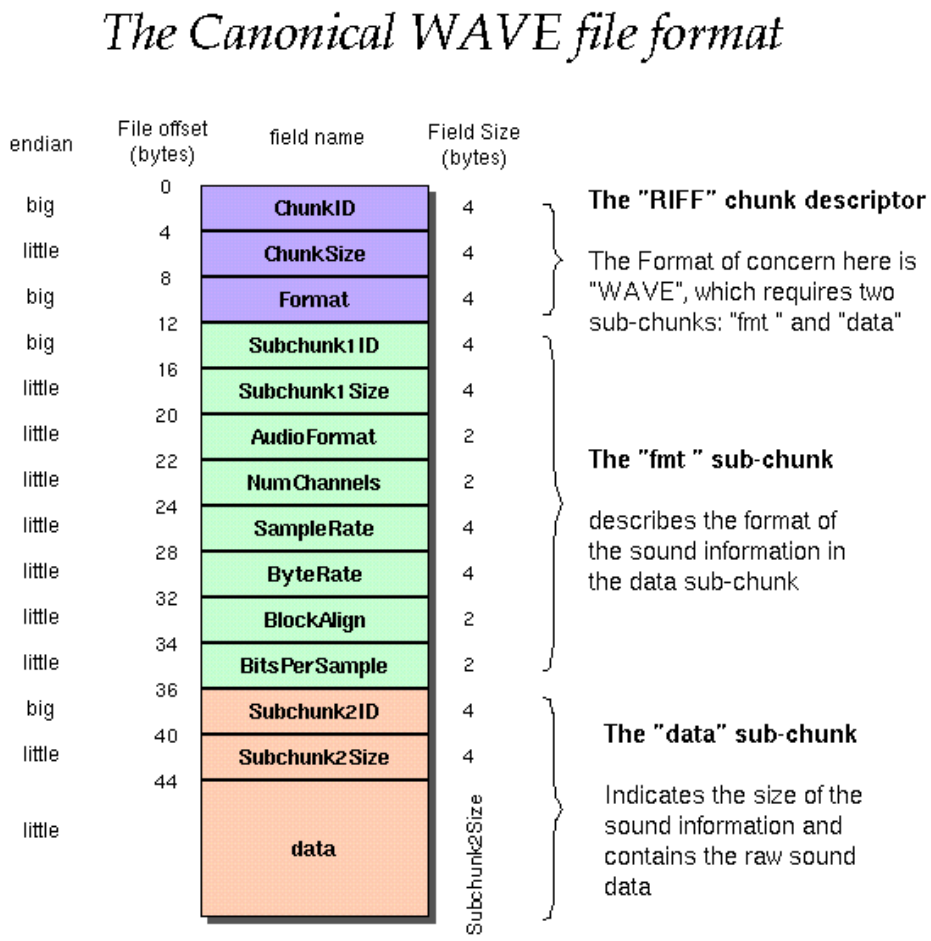

Figura 3: Formato WAV

- ByteRate

  SampleRate * NumChannels * BitsPerSample/8

- BlockAlign

  NumChannels * BitsPerSample/8

  El número de bytes para cada sample incluyendo todos los canales.

- BitsPerSample

  8 bits = 8, 16 bits = 16, etc.

- Subchunk2ID

  Contiene las letras “data”.

- Subchunk2Size

  NumSamples * NumChannels * BitsPerSample/8

  Número de bytes de datos.

- Data

  Los datos de sonido, divididos en samples.

3. Descripción del problema

En un sistema de estaganografía de audio, se ocultan mensajes secretos en audio digital. El mensaje secreto se introduce alterando ligeramente la secuencia binaria del archivo de audio. Existen aplicaciones de esteganografía de audio que permiten ocultar mensajes en archivos de sonido en formato WAV, AU e incluso MP3.

Ocultar mensajes secretos en archivos de sonido suele ser más difícil que ocultar mensajes en archivos de imagen o vídeo. Esto se debe a la mayor sensibilidad que presenta en las personas el sistema auditivo frente al sistema visual. Existen un buen número de métodos y algoritmos para llevar a la práctica estos conceptos, algunos de ellos sencillos y otros más complejos (basados en técnicas avanzadas de procesado digital de señales). Estos métodos serán analizados en detalle en la sección "Soluciones presentadas".

Podemos clasificar las aplicaciones de la esteganografía de audio en dos campos fundamentales: privacidad y protección de copyright.

3.1. Privacidad

La esteganografía, y en particular la esteganografía de audio, tiene un campo de aplicación muy evidente en el envío de mensajes secretos. En este caso, el objetivo es proteger las comunicaciones entre un emisor y un receptor, de manera que una tercera parte (un enemigo) no pueda tener acceso a la información. El hecho de que la esteganografía pueda ser combinada con técnicas de criptografía la convierte en una herramienta muy potente. Así, la esteganografía añade una primera "barrera de defensa". El atacante tendrá que ser capaz de detectar la mera existencia de la información, para posteriormente enfrentarse al criptoanálisis de la misma.

3.2. Protección de copyright

El rápido desarrollo del software y la bajada de precios de los dispositivos digitales han hecho posible que consumidores de todo el mundo puedan hoy en día crear e intercambiar datos multimedia con gran facilidad. Internet de banda ancha proporciona transmisiones de datos a gran velocidad con una tasa de errores muy baja. Esto ha facilitado que cualquiera pueda distribuir archivos multimedia de gran tamaño y hacer copias exactas de los mismos. La capacidad de llevar a cabo estas copias perfectas dificulta la protección

de los derechos de propiedad intelectual. La posibilidad de crear y distribuir un número ilimitado de copias de sus obras causa un daño económico considerable para los autores.

Los métodos tradicionales de protección de copyright ya no son suficientes. Los métodos de protección hardware no son una opción, ya que el mercado se está alejando cada vez más de los dispositivos físicos. Los métodos básicos de protección software basados en la inclusión de información en la cabecera de los archivos multimedia tampoco son de utilidad, ya que la cabecera de los archivos puede ser fácilmente sustituida por otra, sin que los datos multimedia sufran ningún tipo de deterioro.

Algunos proveedores buscaron la solución en la criptografía. Su idea era distribuir los archivos cifrados, con una clave diferente para cada cliente. De esta manera sólo el cliente que hubiese pagado su licencia podría descifrar los contenidos del archivo. El problema es que una vez que un cliente legítimo ha descifrado su archivo, nada le impide llevar a cabo copias del mismo y distribuirlas. Por tanto, el sistema sólo sirve para garantizar la comunicación segura entre el proveedor y el cliente, pero no pone ningún impedimento a la difusión un número ilimitado de copias por parte de dicho cliente.

Las marcas de agua digitales son una nueva alternativa para tratar de proteger el copyright. Se definen como la comunicación de datos relacionados con una señal portadora, de manera imperceptible, robusta y segura. En el caso de archivos de audio, la señal portadora es el archivo de sonido original. La información a comunicar de forma imperceptible, robusta y segura sería una firma digital que informara acerca de quién es el "dueño" de dicho archivo. Pese a ser transparente a la percepción humana, esta firma digital debería poder ser leída con las herramientas adecuadas.

Una diferencia fundamental entre la esteganografía general y la esteganografía de marcas de agua es que en el caso de las marcas de agua, la información que se inserta tiene que ver con la señal portadora (por ejemplo, indica quién es su propietario). La diferencia es de concepto, las técnicas empleadas son las mismas.

En resumen, el uso de esteganografía de audio en protección de derechos de propiedad intelectual se basa en la capacidad de ocultar información en los archivos multimedia. Esta información debe resultar imperceptible: el cliente no debe apreciar ningún tipo de pérdida de calidad en su archivo multimedia. Los datos insertados informan acerca de quién es el legítimo propietario de esa copia de dicho archivo. De esta manera, en caso de producirse su copia y distribución masiva, sería posible localizar a su propietario original. Esta

información debe además ser robusta y segura (no debería ser posible modificarla ni eliminarla fácilmente). Para ello es fundamental que se incluya en los mismos datos digitales, y no en la cabecera de los archivos.

4. Soluciones presentadas

En esta sección presentamos algunos de los métodos más comúnmente utilizados en esteganografía de audio. Pueden encontrarse implementaciones de estos métodos en la Web. Algunos de los métodos más avanzados requieren conocimiento previo de técnicas de procesado de señales, análisis de Fourier y otras áreas matemáticas. Se ha preferido utilizar diagramas y pseudocódigo en lugar de fórmulas matemáticas exactas para intentar hacer la teoría más accesible a lectores con conocimientos básicos de esteganografía.

4.1. Codificación en el bit menos significativo (LSB coding)

Es el método más simple de inclusión de información en un archivo de audio digital. Se basa en sustituir el bit menos significativo de cada sample (porción de sonido) con un bit del mensaje a ocultar. Este método permite ocultar grandes cantidades de información. La figura 4 ilustra cómo se oculta el mensaje "HEY" en un archivo de audio compuesto por samples de 16 bits.

Algunas implementaciones de este método utilizan los dos (o más) bits menos significativos de cada sample para ocultar información. Esto hace que la cantidad de información que se puede insertar aumente, pero también eleva la cantidad de ruido que se introduce en la señal de audio resultante. Por ello, es importante considerar el tipo de señal de audio antes de decidir el número de bits a utilizar por cada sample. Por ejemplo, un archivo de sonido grabado en una estación de metro con gran cantidad de ruido de fondo podría soportar la inclusión de dos o más bits de información en cada sample, ya que el ruido de fondo enmascararía el ruido introducido en el proceso de ocultación de información. Por contra, un archivo de sonido que contenga un solo de piano será mucho más sensible a estas modificaciones, por lo que el uso de más de un bit por sample podría hacer que los cambios resultaran perceptibles al oído humano.

Para poder extraer un mensaje secreto codificado utilizando LSB, el receptor necesita tener acceso a la secuencia de samples utilizada en el proceso de inserción de información. Normalmente, la longitud del mensaje secreto a ocultar es mucho menor que el número total de samples del archivo de sonido. Una decisión a tomar consiste por tanto en cómo elegir qué subconjunto de samples del archivo original se usarán para contener información. El emisor y el receptor deben ponerse de acuerdo en esta cuestión. Una técnica

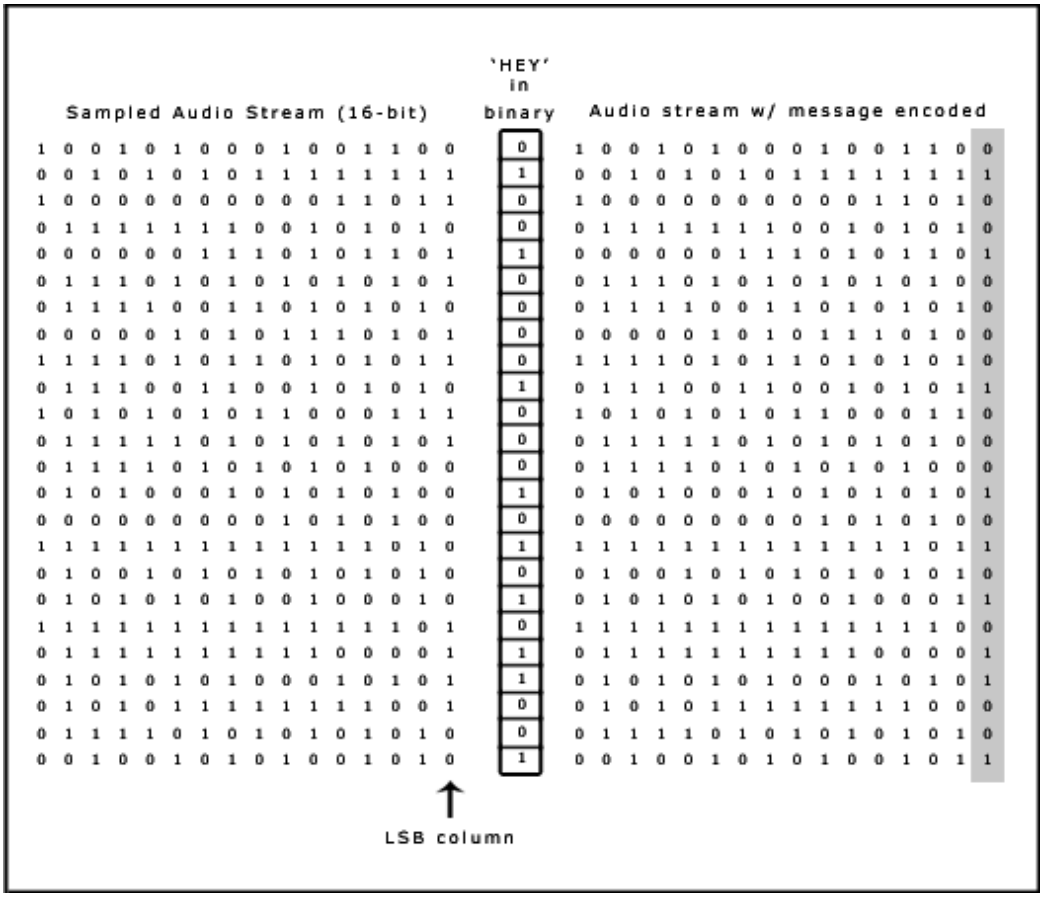

Figura 4: LSB Coding

trivial consiste en comenzar en el inicio del archivo de sonido y llevar a cabo la inserción de información en los primeros samples, hasta que el mensaje haya sido completamente introducido. A partir de ahí, el resto de samples quedarían inalterados. El problema de esta técnica radica en que la primera parte del archivo de sonido tendrá propiedades estadísticas diferentes a las del resto del archivo, que no fue modificado. Una solución a este problema es añadir al mensaje secreto una secuencia de bits aleatoria de manera que la longitud total del mensaje sea igual al número total de samples. El inconveniente es que el proceso de inserción modificaría ahora muchos más samples de los que la transmisión del mensaje original requería, lo cual hace que sea más sencillo detectar la existencia del mensaje oculto.

Una solución más sofisticada se basa en usar un generador de números pseudoaleatorios para distribuir el mensaje entre los samples del archivo de sonido. Para poder hacer esto, el emisor y el receptor deben ponerse de acuerdo en la semilla que utilizarán para generar la secuencia pseudoaleatoria de samples a utilizar. De esta manera el receptor puede reconstruir la secuencia y averiguar cuáles son los samples que contienen el mensaje oculto. Hay que tener cuidado con que el generador no produzca el mismo sample dos veces. Si esto ocurriese, se modificaría el bit menos significativo de un sample cuyo bit menos significativo ya había sido modificado, produciéndose una colisión. Una manera de evitar esto es almacenar en una lista los samples que ya han sido modificados, de manera que si el generador indica que se vuelva a utilizar un sample ya empleado, se ignora. Otra alternativa consiste en calcular una permutación pseudoaleatoria del conjunto de samples del archivo de audio, y seleccionar los primeros. Con esta técnica es imposible que se utilice el mismo sample más de una vez.

4.2. Codificación en paridad (Parity coding)

En lugar de dividir la señal en samples individuales, este método divide la señal en regiones disjuntas. Una región es un conjunto de samples, de un tamaño determinado. Todas las regiones tendrán el mismo tamaño, excepto quizás la última, si el número de samples del archivo no es múltiplo del número de samples por región. El método oculta un bit del mensaje secreto en el bit de paridad de cada región. El bit de paridad de una región se define como el resultado de aplicar la operación lógica XOR a todos los bits de todos los samples de la región. Si el bit de paridad de la región seleccionada (aquella en la que se ha decidido ocultar un bit del mensaje) no coincide con

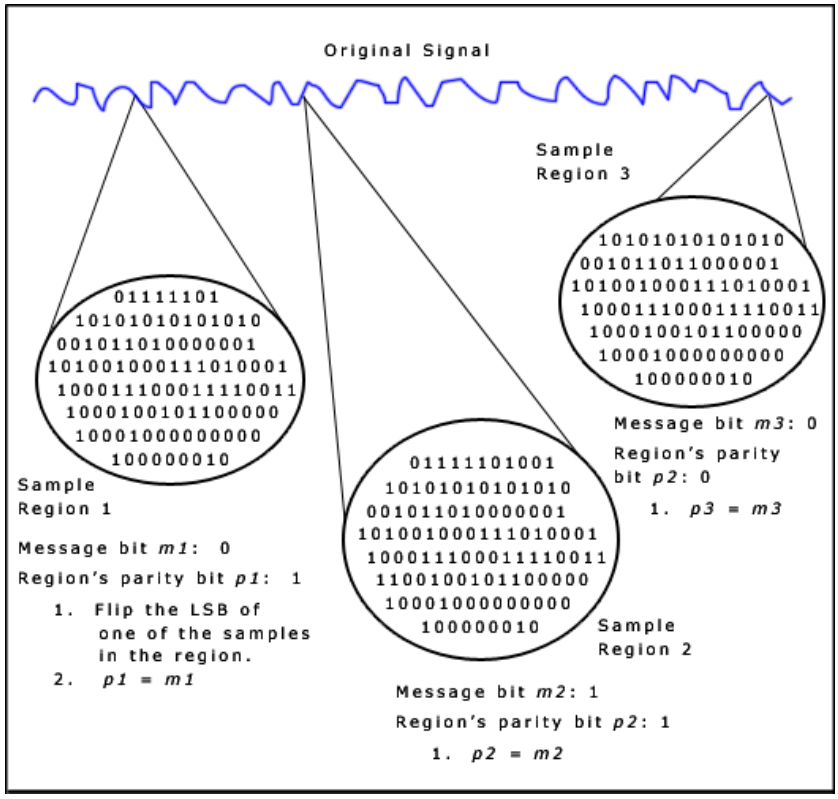

Figura 5: Parity Coding

el bit secreto a incrustar, se modifica el bit menos significativo de uno de los samples de la región (no importa cual). Si coincide, no es necesario llevar a cabo ninguna modificación. De esta manera, el emisor tiene una serie de samples entre los que elegir a la hora de ocultar cada bit.

La figura 5 ilustra cómo se ocultan, usando este método, los tres primeros bits del mensaje "HEY" (010, ya que en formato ASCII la 'H' se representa por el número 72 en decimal, 01001000 en binario).

Para poder llevar a cabo el proceso de extracción de la información el receptor debe conocer el tamaño de las regiones así como el orden en el que éstas fueron utilizadas a la hora de ocultar la información. Como en el método anterior, puede optarse por utilizar las primeras regiones del archivo de sonido de manera secuencial, o bien por usar un generador pseudoaleatorio que, a partir de una semilla, determine el orden en que deben escogerse las regiones en las que ocultar información. En caso de utilizar esta segunda

alternativa, la semilla y el generador pseudoaleatorio deben ser compartidos por emisor y receptor, evidentemente.

El proceso de obtención de la información oculta se limita a calcular los bits de paridad de las regiones empleadas durante el proceso de inserción del mensaje. La concatenación de estos bits de paridad dará como resultado el mensaje original. Como puede observarse, el receptor no necesita conocer en ningún momento qué samples concretos se han visto modificados con respecto al archivo de sonido original.

Los métodos de codificación en el bit menos significativo y codificación de paridad presentan dos desventajas fundamentales. En primer lugar, ambos introducen ruido en la señal original, y el oído humano es muy sensible al ruido, a menudo siendo capaz de detectar incluso las cantidades más insignificantes. El método de codificación de paridad es ligeramente mejor en este aspecto, ya que asegura que sólo se modificará como máximo un bit en cada región. La otra gran desventaja que presentan estos métodos es que no son robustos. Pequeñas modificaciones en el archivo de sonido que contiene el mensaje secreto podrían destruir dicho mensaje. La robustez de estos métodos puede aumentar en cierta medida si se utilizan técnicas de redundancia, ocultando cada bit del mensaje secreto en dos o más localizaciones del archivo de audio. El problema que presenta esta solución es que aumenta la cantidad de bits a incrustar en el archivo de sonido original, lo cual reduce el tamaño máximo del mensaje que se puede ocultar en una señal de audio determinada a la vez que se incrementa la probabilidad de que las modificaciones realizadas resulten perceptibles.

4.3. Codificación en fase (Phase coding)

A diferencia de los métodos anteriores, el método de codificación en fase no está basado en la inserción de ruido y por tanto no comparte los inconvenientes previamente señalados. La idea central en codificación en fase es que las componentes de fase del sonido no resultan tan perceptibles al oído humano como el ruido. Por ello, en lugar de introducir perturbaciones, esta técnica codifica los bits del mensaje como desplazamientos de fase en el espectro de fases de la señal digital.

El procedimiento es el siguiente:

1. La señal de sonido original se divide en segmentos cuya longitud es igual al tamaño del mensaje a ocultar.

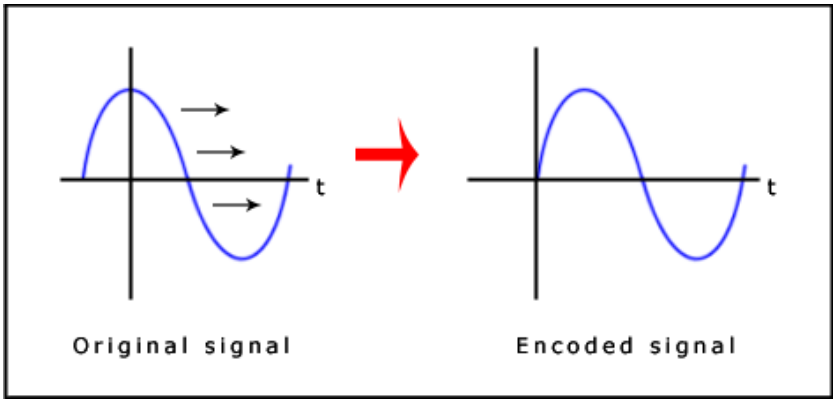

Figura 6: Phase Coding

2. Se aplica la transformada discreta de Fourier a cada segmento para crear una matriz de las fases y magnitudes de la transformada de Fourier.

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

4. Desplazamientos de fase entre segmentos consecutivos serían fácilmente detectables. En otras palabras, las fases absolutas de los segmentos pueden cambiarse pero las diferencias de fase relativas entre segmentos adyacentes deben preservarse. Por este motivo, el mensaje secreto sólo se inserta en el vector de fases del primer segmento de la señal, de manera que la nueva fase será:

$$\text{nueva\_fase} = \begin{cases} \pi/2 & \text{si} \quad \text{bit del mensaje} = 0 \\ -\pi/2 & \text{si} \quad \text{bit del mensaje} = 1 \end{cases}$$

5. Se crea una nueva matriz de fases usando las nuevas fases del primer segmento y respetando las diferencias de fase originales.

6. Usando la nueva matriz de fases y la matriz de magnitudes original, se reconstruye la señal de sonido aplicando la transformada discreta de Fourier inversa, y después concatenando los segmentos.

Para extraer el mensaje secreto del archivo de sonido, el receptor debe conocer la longitud del segmento. Entonces puede usar la transformada discreta de Fourier para obtener las fases y extraer la información.

Una desventaja del método de codificación en fase es su baja tasa de transferencia de información, debida al hecho de que el mensaje secreto sólo se codifica en el primer segmento de la señal. Este inconveniente podría mitigarse aumentando la longitud de los segmentos de la señal. Sin embargo, esto haría que las relaciones de fase entre cada componente de frecuencia cambiase de manera más drástica, haciendo más sencillo detectar la existencia del mensaje oculto. Por tanto, el método de codificación en fase se usa cuando sólo es necesario transmitir una pequeña cantidad de datos.

4.4. Espectro ensanchado (Spread spectrum)

En el contexto de la esteganografía de audio, el método básico de ensanchado del espectro intenta expandir la información secreta lo máximo posible en el espectro de frecuencia de la señal de audio. Esto es análogo a un sistema de codificación en el bit menos significativo en el que el mensaje se ocultara en bits elegidos aleatoriamente de entre todos los bits del archivo de sonido. Sin embargo, a diferencia de lo que ocurre en LSB coding, el método de espectro ensanchado distribuye los bits del mensaje secreto en el espectro de frecuencia del archivo de sonido, usando un código que es independiente de la señal. Como resultado, la señal final ocupa un ancho de banda mayor al originalmente requerido para la transmisión.

Se pueden usar dos versiones del método de espectro ensanchado en esteganografía de audio: el esquema de secuencia directa y el esquema de salto de frecuencia. En secuencia directa, el mensaje secreto se distribuye utilizando una constante llamada "chip rate" y luego se modula con una señal pseudoaleatoria. Después se entrelaza con la señal original. En salto de frecuencia, el espectro de frecuencia del archivo de audio es alterado para que salte rápidamente entre frecuencias.

La teoría matemática en la que se basa este método es bastante complicada y va más allá de los objetivos de este trabajo. El diagrama de la figura 7 ilustra el diseño de un sistema de esteganografía de audio basado en espectro ensanchado utilizando el esquema de secuencia directa.

El método de espectro ensanchado ofrece una tasa de transmisión de datos moderada y un nivel de robustez alto (más alto que el obtenido con cualquiera de los métodos anteriores). Sin embargo, comparte una desventaja con los métodos de codificación en el bit menos significativo y codificación en paridad: introduce ruido en el archivo de sonido.

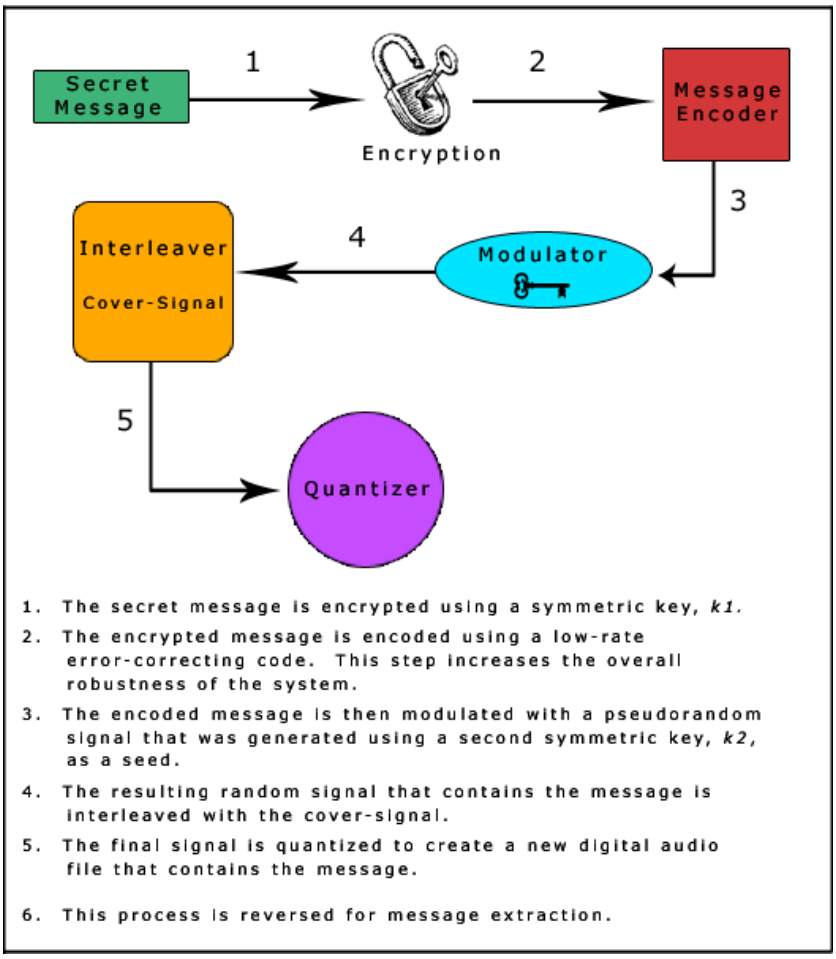

Figura 7: Spread Spectrum

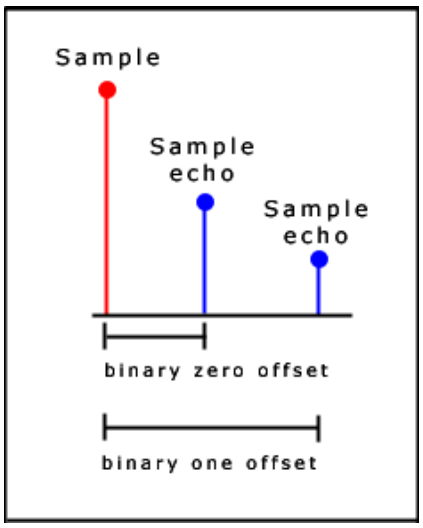

Figura 8: Offsets del eco

4.5. Ocultamiento en eco (Echo hiding)

En este método la información es insertada en el archivo de sonido introduciendo un eco en la señal discreta. Como el método de espectro ensanchado, ofrece una tasa de transmisión de datos alta y proporciona una robustez elevada.

Para ocultar la información con éxito, se utilizan tres parámetros del eco: amplitud, decadencia y offset (retraso) de la señal original. Todos estos parámetros se establecen en valores inferiores al umbral de perceptibilidad que marca el oído humano. El offset varía según el bit del mensaje original a insertar. Un valor de offset representa el uno binario y un segundo valor de offset representa el cero binario.

Si sólo se produjera un eco a partir de la señal original, sólo podría insertarse un bit de información. Por ello, la señal original se divide en bloques antes de que comience el proceso de inserción de información. Una vez que el proceso termina, los bloques son concatenados para crear la señal de audio final.

A continuación describiremos una versión sencilla del método de ocultamiento de eco. Dividiremos la señal completa en bloques, aunque en circunstancias normales debería dejarse un número aleatorio de samples sin usar entre cada par de bloques, para reducir la probabilidad de detección.

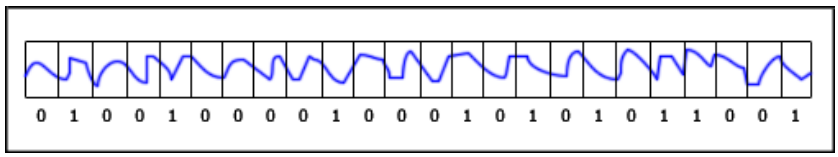

Figura 9: Señal dividida en bloques y bit asociado a cada bloque

En primer lugar la señal se divide en bloques, y a cada bloque se le asigna un uno o un cero en función del mensaje secreto a ocultar. En nuestro ejemplo, el mensaje será la secuencia binaria asociada a 'HEY'(véase figura 9).

A continuación se usa el siguiente algoritmo (se muestra pseudocódigo) para insertar la información necesaria en cada bloque:

```
init(Block blocks[]) {
    for (int i=0; i < blocks.length; i++) {
        if (blocks[i].echoValue() == 0)
            blocks[i] = offset0(blocks[i]);
        else
            blocks[i] = offset1(blocks[i]);
    }
}
Block offset0(Block block) {
    return (block + (block - OFFSET_0));
}

Block offset1(Block block) {
    return (block + (block - OFFSET_1));
}
```

Por último, los bloques se recombinan para producir la señal final.

Usar la implementación descrita normalmente resultará en una señal con una mezcla de ecos considerable, lo que incrementa el riesgo de detección. Describimos ahora una segunda implementación que intenta solventar este problema. Primero se crea una señal de eco de la señal original completa utilizando el offset asociado al cero binario. Luego generamos una segunda señal de eco asociada a la señal original global, usando el valor de offset asociado al uno binario. La señal "uno" sólo contiene unos, y la señal "cero" sólo contiene ceros. Para combinar las dos señales de eco y conseguir la señal

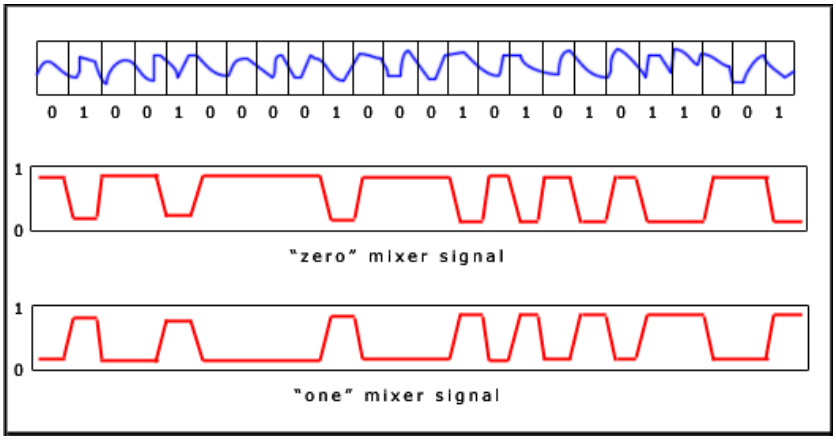

Figura 10: Señales de mezcla

final, se usan dos señales de mezcla. Las señales de mezcla tienen un valor de uno o cero, según el bit a insertar en el bloque. En nuestro ejemplo, para ocultar el mensaje "HEY" obtendríamos las señales de mezcla mostradas en la figura 10.

La señal de eco "uno" se multiplica por la señal de mezcla "uno" y la señal de eco "cero" se multiplica por la señal de mezcla "cero". Hecho esto, las dos señales resultantes se suman para obtener la señal final. La señal final obtenida utilizando este procedimiento es menos abrupta que la obtenida usando el primer procedimiento descrito. Esto se debe a que las dos señales de mezcla son complementarias y a que se usan transiciones lineales suaves.

El diagrama de la figura 11 resume esta segunda implementación del método de ocultamiento en eco.

Para extraer el mensaje secreto, el receptor debe ser capaz de dividir la señal en el mismo conjunto de bloques usado durante el proceso de inserción de la información. Hecho esto, puede usarse la función de autocorrelación del cepstrum de la señal (el cepstrum es la transformada de Fourier del espectro de frecuencias de la señal) para extraer el mensaje, ya que ésta muestra un pico en cada offset de eco en el tiempo, lo que permite reconstruir el mensaje.

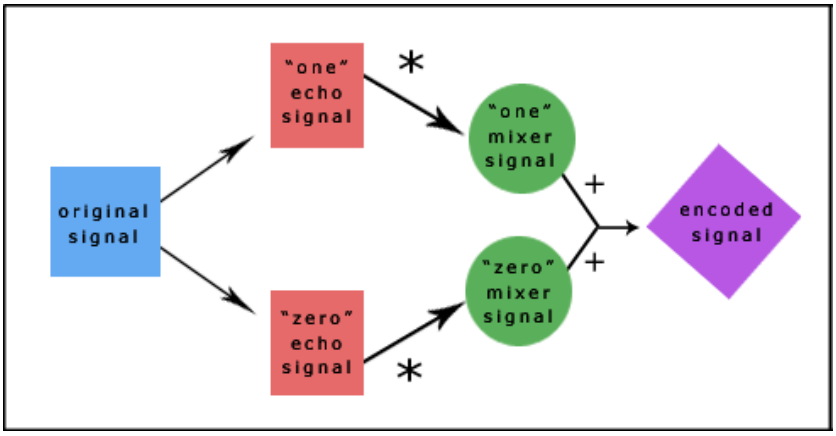

Figura 11: Ocultamiento en eco: segundo procedimiento

5. Implementación realizada

Parte del tiempo empleado en la realización de este trabajo ha sido dedicado al desarrollo de una aplicación que pusiera en práctica los conceptos previamente analizados de manera teórica. Si bien no se nos pedía explícitamente que desarrolláramos una aplicación, hemos pensado que sería muy interesante poder aplicar algunas de las ideas tratadas y observar los resultados obtenidos experimentalmente. Para el desarrollo de esta aplicación se ha utilizado el lenguaje de programación C#.

La aplicación se distribuye en formato ejecutable (.exe), por lo que no es necesaria instalación. Está preparada para ser ejecutada en entornos con sistema operativo Windows y la versión 3.5 (o posterior) del .NET Framework.

5.1. Manual de uso

Tras arrancar, la aplicación muestra una ventana vacía. Para poner en marcha el programa, es necesario en primer lugar cargar un archivo de audio en formato WAV. Para ello se usa la opción *Abrir* del menú desplegable *Archivo*. La figura 12 ilustra la apariencia del programa una vez cargado el archivo de audio correspondiente.

En la región superior izquierda, se muestra un menú desplegable que permite seleccionar el algoritmo de esteganografía de audio a aplicar. Hay dos opciones: codificación en el bit menos significativo (*LSB*) y codificación en paridad (*Parity Coding*). Bajo el menú desplegable, un campo de texto

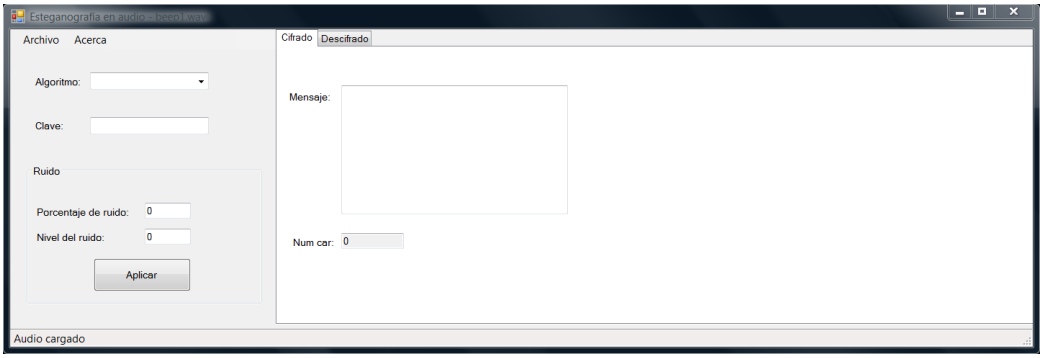

Figura 12: Apariencia de la aplicación tras cargar un archivo de audio

nos permite introducir la clave con la que se cifrará el mensaje oculto. El algoritmo de cifrado empleado será XOR.

También en la región izquierda encontramos una zona denominada *Ruido*. Dos campos de texto nos permiten seleccionar el *Porcentaje de ruido* (porcentaje de samples que se verán afectados por el ruido, de 0 a 100) y el *Nivel de ruido* (número de bits afectados dentro de cada uno de estos samples, de 0 al número de bits por sample del archivo cargado). El botón *Aplicar* lleva a cabo la inserción de ruido en el archivo de audio cargado. La señal de audio resultante se graba al archivo de audio cargado, sobreescribiendo su contenido original. La utilidad de esta funcionalidad de adición de ruido está en permitir aplicar ruido a archivos de audio que contengan un mensaje oculto, y observar cómo afecta este ruido en la correcta extracción del mensaje.

La región derecha de la ventana se encuentra dividida en dos pestañas, dedicadas a los procesos de *Ocultación* y *Extracción* respectivamente. Es necesario seleccionar un algoritmo de esteganografía para que se carguen las opciones específicas a dicho algoritmo en esta región de la ventana. Comentamos a continuación las diferentes opciones presentes en cada una de las combinaciones posibles.

5.1.1. Ocultación en LSB

La figura 13 muestra la apariencia de la aplicación tras aplicar el algoritmo LSB para la ocultación de un mensaje en un archivo de sonido. En este caso, se ha empleado como clave la palabra “sesamo”, para cifrar el mensaje “esto es un mensaje secreto”. Se ha decidido utilizar 2 bits por sample, lo que permite ocultar un mensaje de 189 caracteres para el archivo de sonido

cargado. La longitud del mensaje (calculada automáticamente) es de 26 caracteres, por lo que el mensaje podrá ser incrustado en la señal de audio sin ningún problema. El campo *Semilla* sirve para establecer la semilla a utilizar en el generador de números aleatorios que genera la secuencia de samples en los que se ocultará la información. El botón *Aplicar* lleva a cabo la inserción del mensaje secreto, y genera un archivo de salida en la carpeta en la que se encontrase el archivo de entrada, añadiéndole el sufijo "out.wav".

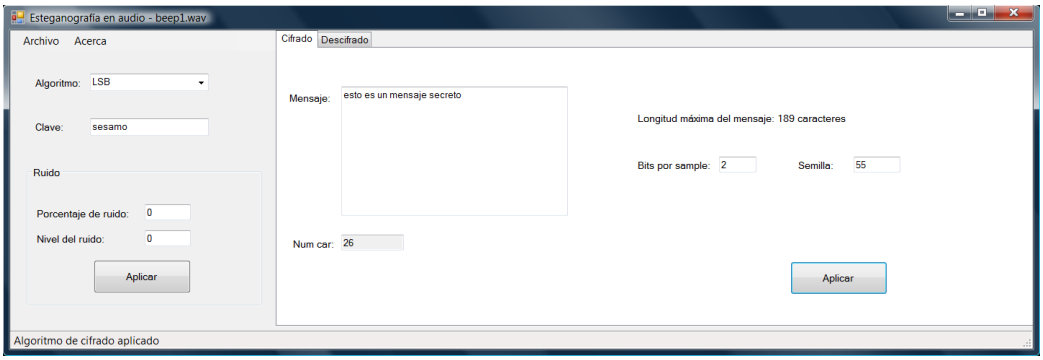

Figura 13: Apariencia de la aplicación tras ocultar un mensaje utilizando LSB

5.1.2. Extracción en LSB

La figura 14 muestra la apariencia de la aplicación tras aplicar el algoritmo LSB para extraer el mensaje insertado en el apartado anterior. Para la correcta extracción del mensaje, es fundamental que los campos *Clave*, *Bits por sample* y *Semilla* tomen los mismos valores que se emplearon para ocultar el mensaje. El campo *Num car* indica el número de caracteres que queremos extraer del archivo de sonido. Como en principio el receptor de un mensaje oculto no conoce la longitud del mensaje secreto, habrá que ajustar este valor manualmente. En el ejemplo, se han extraído 10 caracteres, lo que hace que no se extraiga el contenido completo del mensaje secreto. El botón *Aplicar* es el encargado de poner en marcha el proceso de extracción.

5.1.3. Ocultación en Parity Coding

La figura 15 muestra la apariencia de la aplicación tras aplicar parity coding para ocultar un mensaje en un archivo de sonido. Esta vez se ha

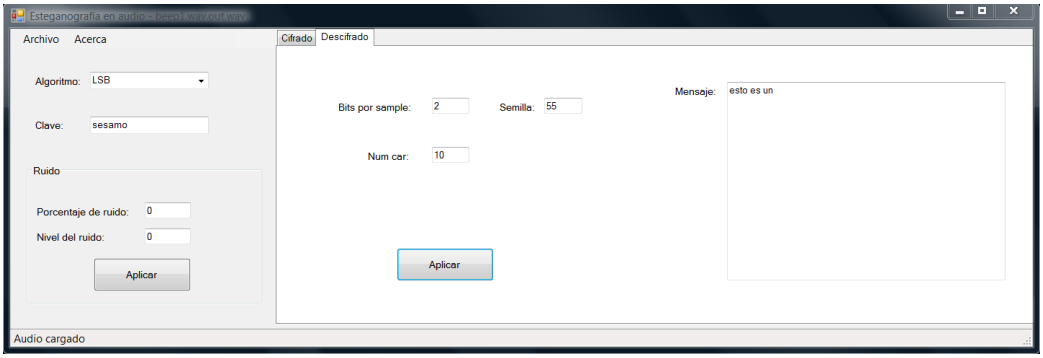

Figura 14: Apariencia de la aplicación tras extraer un mensaje utilizando LSB

empleado la clave "key" para ocultar el mensaje "otro mensaje que queremos ocultar". Se han utilizado regiones de 32 samples y una semilla de 69 para el generador aleatorio. La longitud del mensaje a ocultar es de 33 caracteres, muy inferior a los 594 que el archivo de sonido permite insertar. El botón *Aplicar* pone en marcha el proceso de inserción del mensaje secreto, y genera un archivo de salida en la carpeta en la que se encontrase el archivo de entrada, añadiéndole el sufijo "out.wav".

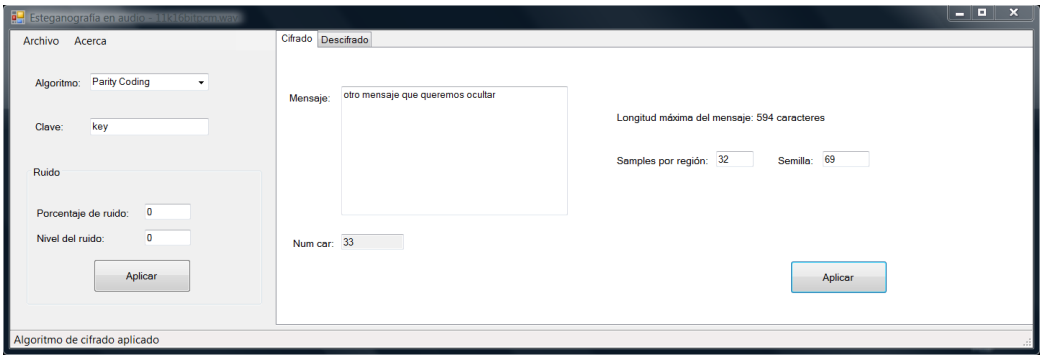

Figura 15: Apariencia de la aplicación tras ocultar un mensaje utilizando Parity Coding

5.1.4. Extracción en Parity Coding

La figura 16 muestra la apariencia de la aplicación tras aplicar parity coding para extraer el mensaje insertado en el apartado anterior. Para la correcta extracción del mensaje, es fundamental que los campos *Samples por región* y *Semilla* tomen los mismos valores que se emplearon para ocultar el mensaje. El campo *Num car* indica el número de caracteres que queremos extraer del archivo de sonido. Como en principio el receptor de un mensaje oculto no conoce la longitud del mensaje secreto, habrá que ajustar este valor manualmente. En el ejemplo, se han extraído 40 caracteres, lo que hace que se extraiga el contenido completo del mensaje secreto y algunos caracteres adicionales. El botón *Aplicar* es el encargado de poner en marcha el proceso de extracción.

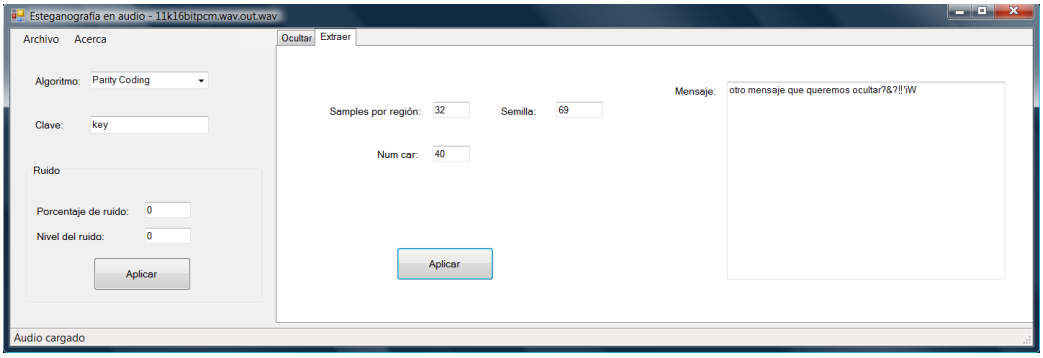

Figura 16: Apariencia de la aplicación tras extraer un mensaje utilizando parity coding

5.2. Mejoras y ampliaciones

Se ha desarrollado una aplicación con interfaz gráfica y un marcado carácter didáctico. Debido al poco tiempo del que se disponía para la elaboración de este trabajo, no se han desarrollado aspectos que habrían dado lugar a una aplicación mucho más completa. A continuación listamos las mejoras y ampliaciones más importantes que podrían llevarse a cabo:

- Soporte para múltiples formatos de audio

  El programa desarrollado sólo ofrece la capacidad de ocultar datos en archivos de audio en formato WAV. Existe un gran número de formatos

de audio digital, entre los que destacamos AIFF, AU, MP3, OGG y WMA.

- Inclusión de más métodos

  La aplicación desarrollada ofrece dos algoritmos de ocultación de datos: codificación en el bit menos significativo y codificación en paridad. Como se estudió en la sección "Soluciones presentadas", existen más métodos. Se decidió implementar estos algoritmos por ser los más sencillos de entender y de programar desde cero. Para la implementación de los restantes métodos (que trabajan en el dominio de la frecuencia) sería necesario el uso de librerías de C# para procesado digital de señales. En este sentido, se intentó implementar el algoritmo de codificación en fase (para lo cual se hizo uso de la librería de código abierto Math.NET). Finalmente, no fue posible completar dicha funcionalidad por falta de tiempo y por la poca experiencia que los alumnos tienen en las técnicas matemáticas necesarias (que no forman parte de los contenidos de la asignatura).

- Ocultación de datos de diferente índole

  El programa elaborado permite ocultar texto en archivos de audio. El texto a ocultar debe ser escrito en un campo de texto en la ventana de la aplicación. Una mejora consistiría en permitir al usuario elegir un archivo (que podría ser de texto, o de cualquier otro tipo) que se ocultaría en el archivo de audio elegido. El procedimiento aplicado sería el mismo, puesto que cualquier archivo puede interpretarse como una secuencia de bits.

5.3. Otras implementaciones

Si bien no es un campo tan desarrollado como la esteganografía en imágenes, existen varias aplicaciones (comerciales o gratuitas) de esteganografía en archivos de audio. Exponemos a continuación algunas de las más conocidas:

- SilentEye (http://www.silenteye.org/)

  Gratuita. Oculta archivos de cualquier tipo en archivos de audio en formato WAV.

- Steghide (http://steghide.sourceforge.net/)

  Gratuito. Oculta datos en archivos de imagen o audio. Los formatos de audio soportados son WAV y AU.

- MP3Stego (http://www.petitcolas.net/fabien/steganography/mp3stego/index.html)

  Gratuito. Oculta archivos de texto dentro de archivos de audio en formato MP3.

- Invisible Secrets (http://www.invisiblesecrets.com/)

  Comercial. Oculta archivos en archivos de imagen o audio. El único formato de audio soportado es WAV.

6. Problemas abiertos

La principal dificultad en el campo de la ocultación de información en audio digital es comúnmente ilustrada utilizando el llamado "triángulo mágico" (figura 17). La inaudibilidad o imperceptibilidad, la robustez frente a ataques y la tasa de transferencia de datos son las tres esquinas del triángulo mágico. Estos tres requisitos son contradictorios. No es posible obtener todas estas características simultáneamente. Por ello, deberá estudiarse cada situación particular para elegir el método más apropiado a aplicar en cada caso.

Por ejemplo, para mejorar la robustez suele recurrirse a ocultar el mensaje varias veces en el archivo de audio. Si se oculta el mensaje dos veces, el efecto sería similar a ocultar un mensaje de longitud igual al doble de la original. Evidentemente, esto reduce la tasa de transferencia de datos. En concreto, el tamaño máximo de mensaje que se podrá incluir será la mitad. Igualmente, esto reduce la imperceptibilidad, puesto que para ocultar el mismo mensaje será necesario modificar el doble de bits, incrementando las posibilidades de que el mensaje oculto pueda ser detectado.

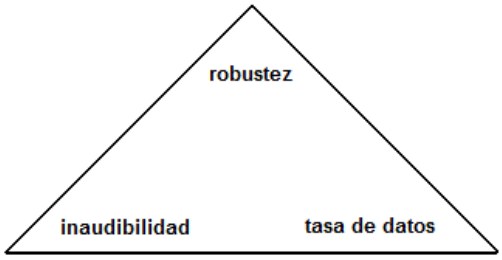

Figura 17: Triángulo mágico

Otros problemas interesantes son:

- ¿Cuál es la máxima tasa de transferencia de datos que se puede conseguir con cada método, respetando los límites que marca la necesidad de imperceptibilidad?
- ¿Cómo puede evaluarse la robustez frente a ataques (intencionados o no intencionados) de cada uno de estos métodos?

7. Conclusiones

La esteganografía de audio es muy flexible y esto es lo que la hace tan interesante. Los cinco métodos analizados dan a los usuarios gran capacidad de decisión y hacen la tecnología accesible a un gran número de personas. A la hora de escoger qué método de esteganografía de audio emplear, debe estudiarse la importancia de factores como la tasa de transferencia de datos, el ancho de banda, la robustez y la perceptibilidad, para elegir el método que mejor se ajuste a las necesidades. Por ejemplo, dos personas que sólo necesitan enviarse un mensaje secreto ocasionalmente podrían usar el método de codificación en el bit menos significativo, fácil de implementar. Por contra, una gran multinacional con la intención de proteger su propiedad intelectual de "piratas digitales" probablemente consideraría métodos más sofisticados como codificación en fase, espectro ensanchado u ocultamiento de eco. Otro de los aspectos de la esteganografía de audio que la hace tan atractiva es su capacidad para combinarse con las tecnologías de criptografía ya existentes. Los usuarios no tienen que elegir cuál de los dos métodos utilizar, la información puede ir cifrada y oculta simultáneamente.

En resumen, con el crecimiento de áreas como protección de copyright, protección de privacidad y vigilancia, creemos que la esteganografía seguirá aumentando en importancia. La esteganografía de audio, en particular, se centra en problemas que se originaron con la aparición del audio digital (y su espectacular auge con el formato MP3), el software P2P, y la necesidad de un esquema de comunicaciones seguro que pueda mantener el secreto de la información transmitida incluso si ésta debe atravesar canales inseguros.

8. Tabla de tiempos

Fecha	Inicio	Fin	Duración	Miembro del grupo	Actividad
02-05-2011	15:30	19:30	4h	Juan, Borja, Pascual	Estudio referencias
02-05-2011	19:00	21:00	2h	Pascual	Estudio referencias
04-05-2011	17:00	20:00	3h	Borja	Estudio referencias
06-05-2011	12:00	14:00	2h	Juan, Borja, Pascual	Estudio referencias
07-05-2011	10:00	14:00	4h	Juan, Borja, Pascual	Estudio referencias
09-05-2011	19:00	22:00	3h	Juan	Implementación
10-05-2011	15:30	20:30	3h	Juan, Borja, Pascual	Implementación
11-05-2011	17:30	20:30	3h	Pascual	Presentación
13-05-2011	11:00	14:00	3h	Borja	Memoria
16-05-2011	10:00	14:00	4h	Pascual	Presentación
16-05-2011	16:00	18:00	2h	Borja	Estudio referencias
16-05-2011	17:30	20:30	3h	Borja	Memoria
16-05-2011	19:00	23:00	4h	Juan	Implementación
17-05-2011	10:30	12:30	2h	Borja	Memoria
20-05-2011	16:00	18:00	2h	Pascual	Presentación
21-05-2011	10:00	14:00	4h	Borja	Memoria
21-05-2011	15:00	19:00	4h	Juan	Implementación
22-05-2011	15:00	17:00	2h	Pascual	Presentación
23-05-2011	19:00	21:00	2h	Borja	Memoria
25-05-2011	17:00	20:00	3h	Borja	Memoria
26-05-2011	17:30	20:30	3h	Juan	Implementación
26-05-2011	17:30	20:30	3h	Borja, Pascual	Presentación
28-05-2011	10:00	13:00	3h	Juan, Pascual	Implementación
29-05-2011	11:00	14:00	3h	Juan	Implementación
30-05-2011	15:00	17:00	2h	Borja, Juan	Implementación
30-05-2011	19:00	22:00	3h	Borja	Memoria

Cuadro 1: Tabla de tiempos

Bibliografía

[1] Cvejic, Nedeljko (2004). *Algorithms for audio watermarking and steganography*. Oulu University Press. ISBN 951-42-7384-2. *http://herkules.oulu.fi/isbn9514273842/*

[2] Gibson, Tyler (2003). *Methods of audio steganography*. *http://www.snotmonkey.com/work/school/405/*

[3] John, Corinna (2004). *Hiding Data in Wave Audio Files*. *http://www.codeproject.com/KB/security/steganodotnet8.aspx*

[4] Wilson, Scott (2003). *WAVE PCM soundfile format*. *https://ccrma.stanford.edu/courses/422/projects/WaveFormat/*