Criptografía y Seguridad *(72.44)*

TRABAJO PRÁCTICO DE IMPLEMENTACIÓN: ESTEGANOGRAFÍA

1. Objetivos

- Introducirlos en el campo de la esteganografía y sus aplicaciones.
- Experimentar con métodos de ocultamiento de información en archivos wav, analizando ventajas y desventajas de cada uno.

2. Introducción

La ***esteganografía*** (del griego στεγανοζ *steganos, encubierto u oculto* y γραπηοζ *graphos, escritura*) es la ciencia que se ocupa de la manera de **ocultar** un mensaje.

La existencia de un mensaje u objeto es ocultada dentro de otro, llamado **portador**. El objetivo es proteger información sensible, pero a diferencia de la criptografía que hace ininteligible dicha información, la esteganografía logra que la información pase completamente desapercibida al ocultar su existencia misma.

La criptografía y la esteganografía se complementan. Un mensaje cifrado mediante algoritmos criptográficos puede ser advertido por un intruso. Un mensaje cifrado que, además, ha sido ocultado mediante algún método de esteganografía, tiene un nivel de seguridad mucho mayor ya que los intrusos no pueden detectar su existencia. Y si por algún motivo un intruso detectara la existencia del mensaje, encontrarían la información cifrada.

La estenografía tiene un origen muy antiguo. Ya Heródoto en el año 440 aC narra la historia de un mensaje escrito en una tablilla que es cubierto con cera para pasar desapercibido ante el enemigo. El mensaje puede así llegar a su destino, siendo develado por sus receptores al quitar la cera.

En la era digital, el interés se renueva por sus múltiples aplicaciones, entre otras:

- Protección de derechos de autor (watermarking y fingerprinting)
- Técnicas de anonimato
- Voto electrónico

Los algoritmos de ocultamiento dependen del archivo portador, ya que la alteración del mismo de manera profunda puede despertar sospechas respecto de la existencia de un mensaje. En general se eligen como archivos portadores algún tipo de archivo multimedial: imagen, video o archivo de audio. Pero también existen otras posibilidades como archivos zip o archivos ejecutables.

El ***estegoanálisis*** se ocupa de estudiar métodos para detectar si un archivo ha sido ocultado en otro. Este campo de estudio está teniendo un desarrollo muy importante especialmente por agencias de investigación criminales debido a los alcances que la esteganografía con malos propósitos puede llegar a tener (por ejemplo, ataques terroristas, pedofilia, etc)

3. Consigna

A) Realizar un programa **stegowav** en lenguaje C que efectúe las siguientes operaciones:

1. Oculte un archivo cualquiera en un archivo .wav, mediante un método de esteganografiado elegido, con o sin password.
2. Descubra un archivo oculto en un archivo .wav que haya sido previamente esteganografiado con uno de los métodos provistos.

B) Estegoanalice un archivo .wav para determinar si tiene un archivo incrustado, con qué algoritmo y lo extraiga correctamente.

4. Archivos WAV.

El formato de archivo wav (Waveform Audio File Format) permite almacenar audio digital de una manera bastante simple, al ser un subconjunto de lo que se conoce como RIFF.

RIFF (Resource Interchange File Format) es una estructura de archivo que propone Microsoft en el documento “Multimedia Programming Interface and Data Specifications 1.0” para manejar los distintos formatos de archivo multimedia de una manera homogénea y extensible.

El bloque básico de un archivo RIFF se conoce como chunk. Un chunk, en sitaxis de C puede definirse:

```
typedef unsigned long DWORD;
typedef unsigned char BYTE;

typedef DWORD FOURCC;  //Four-character code

typedef FOURCC CKID;   //Four-character-code chunk identifier
typedef DWORD CKSIZE;  //32-bit unsigned size value

typedef struct{              //Chunk structure
      CKID  ckID;            //Chunk type identifier
      CKZIE ckSize;          //Chunk size field (size of ckData)
      BYTE  ckData[ckSize];  //Chunk data
};
```

Todo archivo RIFF comienza con un encabezado de archivo (*RIFF file header*) seguido de una secuencia de bloques de datos (*chunks*). Un archivo WAVE, por ejemplo, es un archivo en cuyo encabezado RIFF figura una identificación como archivo "WAVE" y luego consta de chunks que definen las características del archivo de audio (como por ejemplo número de canales, si está comprimido, etc) y luego los datos del sonido propiamente dichos.

Todo archivo wav tiene, obligatoriamente:

- un chunk que describe el tipo de archivo RIFF, en este caso indica que se trata de un "WAVE" file.
- Un chunk obligatorio de formato de archivo de sonido.
- Un chunk obligatorio de datos de sonido.

Ejemplo de estructura wav:

```
struct wavStr
{
    RIFF_CK riff_desc; // MANDATORY
    FMT_CK  fmt;       // Format Chunk MANDATORY
    //FACT_CK fact;    // Fact Chunk OPTIONAL
    //CUE_CK  cue;     // Cue points Chunk OPTIONAL
    //PLIST_CK   plist;// Playlist Chunk OPTIONAL
    //LIST_CK list;    // Associated data list Chunk OPTIONAL
    // more optional data…
    DATA_CK data;      // Wave Data Chunk MANDATORY
};
```

Así, el chunk de RIFF sería:

```
typedef struct{
    CKID    chunkID;    // 'RIFF'
    CKSIZE  chunkSize;  // File Size
    CKID    format;     // Format: 'WAVE'
}RIFF_CK;
```

El chunk básico de fmt sería:

```
typedef struct{
    CKID         chunkID;    //'fmt '
    CKSIZE       chunkSize;  // 16 para PCM.Size of rest of subchunk.
    /* Common fields */
    WORD         wFormatTag; // Format category,i.e.:PCM = 1 (no
compres.)
```

```
    WORD        wChannels;  // Number of channels:1, mono; 2, stereo
    DWORD       dwSamplesPerSec;    // Sampling rate: Mhz
    DWORD       dwAvgBytesPerSec;
    WORD        wBlockAlign
    WORD        wBitsPerSample; //8, 16, etc.
    WORD        extraParamSize;// If PCM, doesn't exist
    BYTE        *extraParams;//space for extra params
}FMT_CK;
```

El chunk básico de data sería:

```
typedef struct{
    CKID            chunkID;    // 'data'
    CKSIZE          chunkSize;  // Bytes of data
    BYTE            *soundData; // Sound data.
}DATA_CK;
```

Para este trabajo se considerarán archivos wav sencillos, esto es sin chunks del tipo wavl (wavelist), slnt (silent), post (playlists), sin comprimir (sólo PCM).

IMPORTANTE: Leer con cuidado los datos de los campos **obligatorios** de los archivos wav. Los campos opcionales copiarlos sin procesarlos específicamente, esto es no se vuelcan a ninguna subestructura particular, sólo se copian byte a byte.

Lo que interesa para este trabajo práctico es ocultar datos en el área de datos del sonido propiamente dicho (campo `soundData`). Las características que están almacenadas en fmt no deberían modificarse. Por eso los archivos modificados deberían poder reproducirse como cualquier .wav. Deben poder abrirse con editores de audio (por ejemplo Audacity).

IMPORTANTE: Considerar archivos wav sólo en formato PCM

5. Detalles del programa stegowav

5.1. Ocultamiento de un archivo en un .wav.

El programa debe recibir como parámetros:

- **`-embed`**

  Indica que se va a ocultar información.

- **`-in`** file

  Archivo que se va a ocultar.

- **`-p`** wavefile

  Archivo wav que será el portador.

- **`-out`** stegowavefile

  Archivo wav de salida, es decir, el archivo wavefile con la información de file incrustada.

- **`-steg <LSB1 | LSB4 | LSBE>`**

  algoritmo de esteganografiado: LSB de 1bit, LSB de 4 bits, LSB Enhanced

Y los siguientes parámetros opcionales:

- **`-a <aes128 | aes192 | aes256 | des>`**
- **`-m <ecb | cfb | ofb | cbc>`**
- **`-pass password`** (password de encripcion)

Ejemplo 1:

Esteganografiar el archivo de imagen "kitty.bmp" en el archivo portador "panamericano.wav" obteniendo un archivo "papanamericano.wav" mediante el algoritmo LSB Enhanced, con encripción DES en modo CBC con password "oculto"

```
$stegowav -embed –in "kitty.bmp" –p "panamericano.wav" -out "papanamericano.wav" –
steg LSBE –a des –m cbc -pass "oculto"
```

Ejemplo 2:

Esteganografiar el archivo de imagen "kitty.bmp" en el archivo portador "panamericano.wav" obteniendo un archivo "papanamericano.wav" mediante el algoritmo LSB Enhanced, **sin encripción**

```
$stegowav -embed –in "kitty.bmp" –p "panamericano.wav" -out "papanamericano.wav" –
steg LSBE
```

Importante:

No se puede encriptar/desencriptar sin **password**. Si este dato no está, sólo se esteganografia.

Son válidas en cambio las siguientes opciones:

- indicar **algoritmo** y **password** pero no modo: Se asume CBC por default.
- Indicar **modo** y **password** pero no algoritmo: Se asume aes128 por default.
- Indicar **sólo password:** Se asume algoritmo aes128 en modo CBC por default.

5.2. Extraer de un archivo .wav un archivo oculto.

- **-extract**

  Indica que se va a extraer información.

- **-p wavefile**

  Archivo wav portador

- **-out file**

  Archivo de salida obtenido

- **-steg <LSB1 | LSB4 | LSBE>**

  algoritmo de esteganografiado: LSB de 1bit, LSB de 4 bits, LSB Enhanced

Y los siguientes parámetros opcionales:

- **-a <aes128 | aes192 | aes256 | des>**
- **-m <ecb | cfb | ofb | cbc>**
- **-pass password (password de encripcion)**

Ejemplo:

Extraer el archivo de imagen "kitty.bmp" del archivo portador "papanamericano.wav" ocultado mediante el algoritmo LSB Enhanced, con encripción DES en modo CBC con password "oculto"

```
$stegowav –extract –p "papanamericano.wav" -out "kitty" –steg LSBE –a des –m cbc -
pass "oculto"
```

5.3. Algoritmos de Esteganografiado.

Ocultamiento sin encripción:
Antes de ocultar el archivo propiamente dicho, con cualquiera de los algoritmos, ocultar su tamaño.

Después de ocultar el tamaño y el archivo propiamente dicho, con cualquiera de los algoritmos, ocultar su extensión (“.wav”, “.png”, “.jpg”, “.txt”, “.html”, etc)
La extensión debe comenzar con ‘.’ Y terminar con ‘\0’.

Es decir, se esteganografía:

Tamaño real || datos archivo || extensión

Y el total de datos a esteganografiar es:

4 (del tamaño) + longitud archivo + e (extensión)

Siempre se sabe que los primeros 4 bytes (DWORD size) corresponden al tamaño de archivo.
Siempre se sabe que después de los n bytes del archivo viene un punto y los n ascii de la extensión, terminando en ‘\0’.

Ejemplo:

Si el tamaño es 40872 = 0x00009fa8
Y el primer byte del archivo de kitty.bmp es, en hexa, 0x89
Entonces los bytes del mensaje a ocultar son:

Msg[0] →0x00
Msg[1] →0x00
Msg[2] →0x9f
Msg[3] →0xa8
Msg[4] →0x89
….
Msg[40876] →0x2E(‘.’)
Msg[40877] →0x62(‘b’)
Msg[40878] →0x6D(’m’)
Msg[40879] →0x70(’p’)
Msg[40880] →0x00(’\0’)

Ocultamiento con encripción:
Si se eligió encriptar antes, se procede de la siguiente manera:

- Se obtiene la secuencia de bytes correspondiente a **Tamaño real || datos archivo || extensión.**
- Dicha secuencia se encripta con el algoritmo , modo y password.
- Se esteganografia el tamaño del cifrado y a continuación la secuencia cifrada.

Es decir, se esteganografía:

Tamaño cifrado || encripcion(tamaño real || datos archivo || extensión**)**

Y el total de datos a esteganografiar es:

4 (del tamaño del cifrado) + tamaño del cifrado

Para extraer, siempre se sabe que los primeros 4 bytes (DWORD size) corresponden al tamaño de cifrado. Con dicho tamaño, y algoritmo modo y password que se envian por argumento al programa se descifra.
Una vez hecha la desencripción se obtiene el tamaño real del archivo, se lee esa cantidad de bytes y se toma la extensión (hasta el ‘\0’).

Ejemplo:

Si el tamaño real de “saco.txt” es 414 bytes, y se eligio encripcion en Des Cbc.
Entonces se encripta:

414 || datos archivo || “.txt\0”

Que da un total de 4 + 414 + 5 = 423 bytes.
No es múltiplo de bloque así que requiere padding. Por lo tanto se encripta 423 + padding.
Luego, se esteganografia:

(423+padding) || encripcion(414||datos archivo||”.txt\0”)

Es decir, se ocultan **4 (del tamaño del cifrado) + (423 + padding)** bytes.

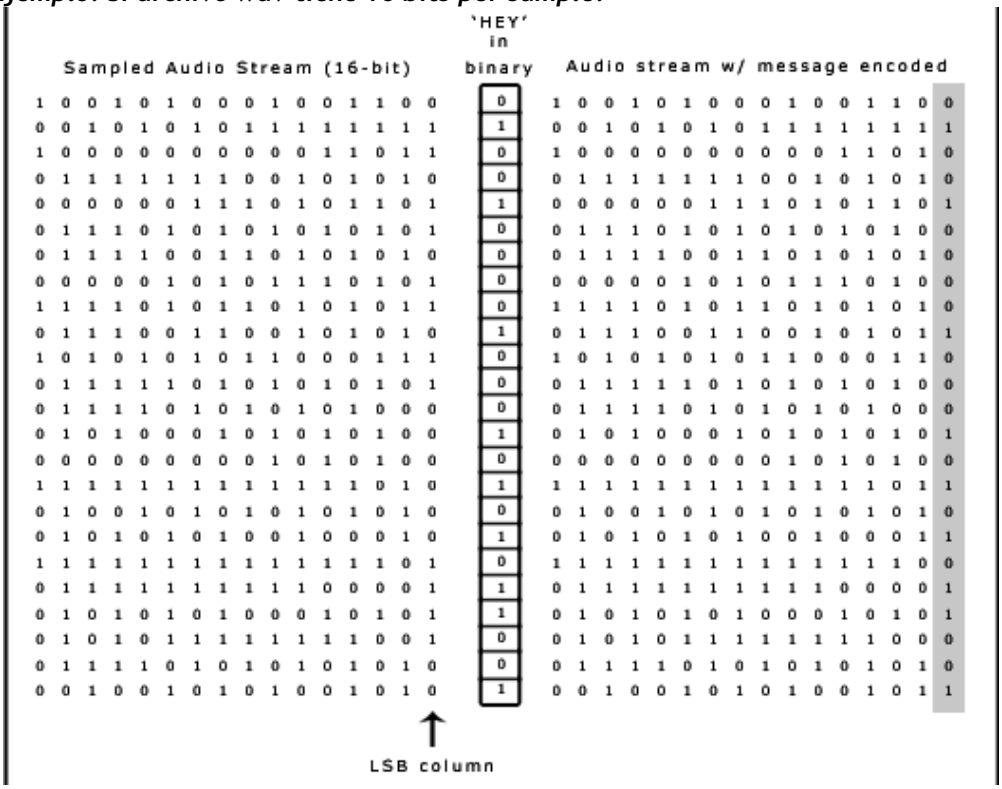
4.3.1. LSB1

Inserción en el bit menos significativo (Least Significant Bit Insertion - LSB)

Es el método más simple y consiste en insertar información sustituyendo el bit menos significativo de cada byte del archivo portador por un bit del mensaje.

En el caso de archivos de audio .wav consideraremos que cada bit del mensaje se inserta en el bit menos significativo de la **muestra** (que puede tener 1 o más bytes)

Ejemplo: Si archivo wav tiene 16 bits per sample.



Como se observa, el primer bit del mensaje a ocultar se inserta en el bit menos significativo de la primer muestra, el segundo bit del mensaje a ocultar se inserta en el bit menos significativo de la segunda muestra, etc. Se requieren entonces 8 muestras para poder guardar un byte del archivo a ocultar.

Importante: Si el archivo wav no puede albergar el archivo a ocultar completo, deberá mostrarse un mensaje de error, en el que se indique cuál es la capacidad máxima del archivo wav portador.

Ejemplo:

Si archivo wav tiene 16 bits per sample.

Si el tamaño es 40873 = 0x00009fa9

Y el primer byte del archivo de kitty.bmp es, en hexa, 0x89

Entonces los bytes del mensaje a ocultar son:

Msg[0] →0x00

Cada 0 va a ubicarse en el bit menos significativo de 8 muestras consecutivas.

Se usan hasta ahí 16 bytes del .wav. Sólo pueden modificarse los bytes 1,3,5,7,9,11,13 y 15 porque las muestras son de 2 bytes. (El primer byte sería el byte 0)

Msg[1] →0x00

Cada 0 va a ubicarse en el bit menos significativo de 8 muestras consecutivas.

Se usan hasta ahí 16 bytes del .wav. Sólo pueden modificarse los bytes 17,19,21,23,25,27,29,31 porque las muestras son de 2 bytes.

Msg[2] →0x9f

Acá se tienen que ubicar:

un 1 en el bit menos significativo del byte 33;
un 0 en el bit menos significativo del byte 35;
un 0 en el bit menos significativo del byte 37;
un 1 en el bit menos significativo del byte 39;
un 1 en el bit menos significativo del byte 41;
un 1 en el bit menos significativo del byte 43;
un 1 en el bit menos significativo del byte 45;
un 1 en el bit menos significativo del byte 47;

etc.

4.3.2. LSB4

Inserción en los cuatro bits menos significativos

Es una variante del anterior, donde 4 bits del mensaje se ocultan en los 4 bits menos significativos de la muestra del wav.

Ejemplo:

Si archivo wav tiene 16 bits per sample.
Si el tamaño es 40873 = 0x00009fa9
Y el primer byte del archivo de kitty.bmp es, en hexa, 0x89
Entonces los bytes del mensaje a ocultar son:
Msg[0] →0x00 = 00000000
0000 va a ubicarse en los 4 bits menos significativos de la primer muestra. Es decir, el byte 1.
0000 va a ubicarse en los 4 bits menos significativos de la segunda muestra. Es decir, el byte 3.
Se usan hasta ahí 2 bytes del .wav.
Msg[1] →0x00 = 00000000
0000 va a ubicarse en los 4 bits menos significativos de la tercer muestra. Es decir, el byte 5.
0000 va a ubicarse en los 4 bits menos significativos de la cuarta muestra. Es decir, el byte 7.
Msg[2] →0x9f
Acá se tienen que ubicar:
1001 en los cuatro bits menos significativos del byte 9; (Si el byte era 11101011 queda 11101001)
1111 en los cuatro bits menos significativos del byte 11;
…
etc.

Importante: Si el archivo wav no puede albergar el archivo a ocultar completo, deberá mostrarse un mensaje de error, en el que se indique cuál es la capacidad máxima del archivo wav portador.

4.3.3. LSB Enhanced

Es una variante de los anteriores propuesta por Sridevi, Damodaram y Narasimham en el documento **"Efficient method of audio steganography by modified LSB algorithm and strong encryption key with enhanced security"**.

Dicho documento se encuentra en:
http://www.jatit.org/volumes/research-papers/Vol5No6/15Vol5No6.pdf

- Tener en cuenta que la propuesta trabaja sobre bytes en lugar de muestras.
- No hace falta escribir la marca de fin de archivo que propone el documento, ya que estamos guardando siempre el tamaño del archivo en los primeros 4 bytes del mensaje.

Importante: Si el archivo wav no puede albergar el archivo a ocultar completo, deberá mostrarse un mensaje de error, en el que se indique cuál es la capacidad máxima del archivo wav portador.

5.4. Archivos .WAV y encripción.

Se hacen las mismas consideraciones que en el trabajo práctico anterior, es decir:

- Archivos .wav en formato PCM.
- Archivos .wav sin chunks del tipo wavl (wavelist), slnt(silent), plst(playlists)
- Si el archivo a ocultar requiriera padding usar el padding por defecto de openssl que es PKCS5.
- Para la generación de clave a partir de una password, asumir que la función de hash usada es md5, y que no se usa SALT.
- Para modos de feedback considerar una cantidad de 8 bits

OJO: Ahora el archivo que se encripta es el que se quiere ocultar, y se encripta completo. Es decir, aún si fuera un wav se encripta desde el byte 0 hasta el último (cabecera incluida)

6. Estegoanálisis

La cátedra proveerá de 4 archivos con información oculta. De ellos, uno de los archivos contendrá un archivo ocultado mediante LSB1, otro mediante LSB4 y otro mediante LSBE. Un cuarto archivo ocultará información de otra forma que habrá que descubrir.

En los que están esteganografiados con los métodos LSB1,LSB4, LSBE, se sigue el formato especificado previamente:

tamaño + archivo + extensión

Habrá que obtener los archivos ocultos (estegoanálisis) descubriendo de qué manera fueron ocultados.

En general, el análisis que se puede hacer es:

- por conocimiento de archivo portador
- por repetición de archivo portador
- estadístico
- de tamaño de los archivos
- etc.

IMPORTANTE:

Se recomienda, para este análisis, usar el software Audacity (o similar) y algún editor de archivos binarios (hex editor neo o similar) que permita hacer comparaciones de los archivos, como archivos de audio y como secuencia de bytes.

7. Cuestiones a analizar

Una vez realizado el programa, se resolverán las siguientes **9** cuestiones:

1. Para la implementación del programa stegowav se pide que la ocultación comience en la primer muestra del archivo de audio. ¿Sería mejor empezar en otra ubicación? ¿Por qué?
2. Esteganografiar un mismo archivo en un .wav con cada uno de los tres algoritmos, y comparar los resultados obtenidos. Hacer un cuadro comparativo de los tres algoritmos estableciendo ventajas y desventajas.
3. Para la implementación del programa stegowav se pide que la extensión del archivo se oculte después del contenido completo del archivo. ¿por qué no conviene ponerla al comienzo, después del tamaño de archivo?
4. Explicar detalladamente el procedimiento realizado para descubrir qué se había ocultado en cada archivo y de qué modo.

5. ¿Qué se encontró en cada archivo?
6. Algunos mensajes ocultos tenían, a su vez, otros mensajes ocultos. Indica cuál era ese mensaje y cómo se había ocultado.
7. Uno de los archivos ocultos era una porción de un video, donde se ve ejemplificado una manera de ocultar información ¿cuál fue el portador?
8. ¿De qué se trató el método de estenografiado que no era LSB? ¿Es un método eficaz? ¿Por qué?
9. ¿Qué mejoras o futuras extensiones harías al programa stegowav?

Se recomienda una lectura del material de consulta recomendado para hacer un análisis apropiado.

8. Organización de los grupos

El trabajo será realizado en grupos de máximo 4 integrantes.

9. Entrega

La fecha de entrega es el día **21 de junio**.

Cada grupo entregará el **ejecutable y el código en C**, junto con la **documentación** correspondiente al uso del programa y **el informe** en un mail que se enviará a mroig@itba.edu.ar **y** a rramele@itba.edu.ar

En el **informe debe figurar** la solución correspondiente al estegoanálisis de los archivos que se le entregarán oportunamente al grupo y el análisis de todas las **cuestiones planteadas en el punto 7**.

10. Criterios de Evaluación

Se considerarán los siguientes aspectos:

- funcionamiento correcto del programa (no debe haber errores de compilación ni de ejecución)
- claridad en la documentación y forma de uso del programa
- precisión, formalidad y sentido crítico en la redacción del informe
- presentación en fecha

Se penalizarán trabajos que resulten copias de otros grupos.

11. Material de consulta recomendado

Hay mucho material en internet sobre el tema. Algunos documentos que son útiles para la comprensión del tema son:

- Cummings, Jonathan y otros: Steganography and Digital Watermarking. Disponible en: http://www.cs.bham.ac.uk/~mdr/teaching/modules03/security/students/SS5/Steganography.pdf
- Isaza, Gustavo A. y otros: Análisis de técnicas esteganográficas y estegoanálisis en canales encubiertos, imágenes y archivos de sonido. En http://vector.ucaldas.edu.co/downloads/Vector1_3.pdf
- Diaz Vico, Jesús: Esteganografía y estegoanálisis: ocultación de datos en streams de audio vorbis. Capítulos 3, 4 y 5. Disponible en : http://oa.upm.es/5353/2/TESIS_MASTER_JESUS_DIAZ_VICO.pdf
- Gómez Cárdenas, Roberto: Esteganografía. Disponible en: http://homepage.cem.itesm.mx/rogomez/SlidesCripto/Estegano.pdf
- Esteganografía en audio. http://www.snotmonkey.com/work/school/405/methods.html