72.44 Criptografía y Seguridad
TPI-Esteganografía
Primer cuatrimestre 2022

Integrantes :

- Julián Nicolás Sicardi - Legajo : 60347
- Ana Cruz - Legajo : 60476
- Paula Oseroff - Legajo: 58415

Cuestiones a analizar

Pregunta 1

Para entender mejor la funcionalidad del método de esteganografía LSBI leímos el paper "An improved inverted LSB image steganography" provisto por la cátedra.

El documento sigue la organización habitual de los papers científicos, con una parte inicial llamada abstracto que cuenta a grandes rasgos y con poca profundidad los temas que van a ser tratados en el paper en sí. Luego, en el cuerpo del paper, tenemos una división por secciones: Introducción, Implementación, Resultados y Análisis y por último Conclusión.

En cuanto a la implementación del algoritmo, hubiese sido preferible una descripción más detallada de esta per se. Nuestra percepción fue que en la sección de implementación se habla más de una idea teórica, sin mostrar nada de pseudocódigo que, como programadores, era lo que esperábamos bajo el título mencionado. Fuera de eso, la idea de ambos esquemas está clara como para poder extrapolarlas y llevarlas a una implementación funcional.

La notación utilizada a lo largo del paper nos resultó suficientemente clara. Pudimos notar un pequeñisimo error en el tercer párrafo de la sección "First scheme" en implementación: menciona que hay cuatro posibles combinaciones de los dos bits previos al bit que debe cambiarse, y aparece dos veces "10", cuando una de las dos debería decir "01". También nos pareció que al paper le faltaba aclarar sugerencias sobre donde guardar los 4 u 8 bits de patrones que indican las inversiones presentes según su criterio, pero es un detalle menor.

Pregunta 2

En una primera instancia se buscó comparar la diferencia en tiempos y perturbación visual obtenidos con los diferentes algoritmos de esteganografía.

Para esto, se tomaron 3 variantes a analizar: una primera donde se toma la imagen lado.bmp (observar imágenes en la carpeta analysis/test_no_enc_common/resources) junto a un texto tipo Lorem Ipsum pero que contiene párrafos del guión de la película Back To The Future, una segunda donde se toma una imagen bmp completamente negra (observar texto en la carpeta analysis/test_no_enc_common/resources) ocultando también el guión de película y una tercera donde se hace uso de la misma imagen completamente negra pero ocultando un archivo compuesto únicamente por el carácter ASCII que en binario se representa 1111 1111 (todos los bits prendidos).

El objetivo de la primera prueba es observar que sucede en una situación cotidiana. Aquí se toma una imagen usual en la que también se puede observar un poco de compresión o pixelado, junto a un texto que tiene sentido y no es una serie de caracteres aleatorios.

La segunda prueba busca observar una situación un poco más extrema. Se oculta en un archivo completamente uniforme (todos los pixeles de color negro o, lo que es equivalente, 0000 0000 0000 0000 0000 0000) un texto que podría ser razonable ocultar. Aquí vamos a poder observar de manera mucho más detallada que tan intrusivos son los métodos de ocultamiento.

Por último, la tercera prueba refleja el peor de los casos: ocultar en un archivo completamente negro un texto que solo contiene el caracter 1111 1111. LSB1 y LSB4 van a siempre cambiar el último bit y los últimos 4 bits respectivamente.

Para la primera prueba, se obtuvo la siguiente tabla comparativa de tiempos:

	Real		Usuario		Sistema	
Oculta	Media	Desvío	Media	Desvío	Media	Desvío
eLSB1	0.027	0.001	0.025	0.003	0.003	0.003
eLSB4	0.032	0.001	0.029	0.003	0.003	0.003
eLSBI	0.105	0.002	0.102	0.004	0.003	0.003
xLSB1	0.047	0.001	0.044	0.003	0.003	0.003
xLSB4	0.039	0.002	0.037	0.003	0.002	0.002
xLSBI	0.046	0.002	0.044	0.003	0.002	0.003

Tabla 1: *Tiempos de los algoritmos de ocultamiento LSB sin encriptar para un archivo cotidiano y un texto cotidiano para ocultar. 1000 corridas.*

Para referencia, se deben entender que el método con una x antepuesta es el método utilizado para extracción y el método con una e antepuesta es para ocultamiento.

Se puede generalizar lo ocurrido para esta prueba con las otras dos observando las Tablas 2 y 3 en el Anexo. Lo que ocurre es una hipótesis que se puede realizar desde una primera instancia: a mayor cantidad de bits que se deben procesar, mayor es el tiempo para ocultar lo necesario. Por otro lado, el caso de LSBI que debe moverse y modificar pixeles que ya había modificado anteriormente es causante de esa gran diferencia en tiempos con LSB1 y LSB4.

Es importante observar que LSB1 oculta menos bits por byte pero LSB4 termina ocultando en menor cantidad de bytes, en otras palabras, la cantidad de bits ocultados es la misma. Sin embargo, LSB1 toma menos tiempo en poder ocultar el archivo.

Una decisión tomada que también influye en los tiempos obtenidos es la de seguir procesando los bytes una vez consumido todo el archivo a ocultar. Esto ayuda a equiparar los tiempos aunque no de manera notoria. Una diferencia de

tiempos substancial puede resultar en estar brindando información a posibles atacantes de qué algoritmo se está utilizando para ocultar generando un canal oculto del tipo temporal.

En el caso de la extracción, podemos observar que es marginal la diferencia temporal entre LSB1 y LSBI, lo cual es algo positivo, sin embargo es mucho menor el tiempo empleado para LSB4. Esto tiene una clara correlación con que se requieren menos bytes para completar un mensaje oculto con LSB4.

Una conclusión teniendo en cuenta únicamente la variable temporal es que LSB4 requiere menos bytes y se puede rescatar un mensaje en menos tiempo. LSBI, por otro lado, conlleva mayor complejidad para ocultar pero a la hora de rescatar el archivo oculto es marginal el tiempo que toma respecto a los otros métodos. Deberá evaluarse si se requiere velocidad de ocultamiento o velocidad de rescate según el caso.

Luego podemos observar las imágenes obtenidas del ocultamiento en cada prueba.

En la primera prueba, podemos observar los mayores cambios en la zona blanca de la imagen (observar imágenes en la carpeta analysis/test_no_enc_common/resources). Con LSB4 (observar imágenes en la carpeta analysis/test_no_enc_common/results) es la que se puede observar una mayor cantidad de ruido a un nivel que difícilmente se crea que es pixelado. En otras palabras, hay un flujo de información claro y discernible a nivel del ojo humano.

Por otro lado, con LSB1 y LSBI (observar imágenes en la carpeta analysis/test_no_enc_common/results) puede tranquilamente confundirse con el pixelado de la imagen o que sea una imagen de mala calidad.

En las pruebas 2 y 3 por otro lado (observar imágenes en la carpeta analysis/test_no_enc_extreme y analysis/test_no_enc_extreme_2), si se coloca la imagen original lado a lado de las imágenes que tienen un archivo oculto, se puede observar las diferencias en los píxeles. Cómo ya se mencionó, estas pruebas son un caso extremo y no lo usual.

A partir de lo ya mencionado anteriormente, se puede realizar la siguiente tabla comparativa:

	LSB1	LSB4	LSBI
Ventajas	• La opción más rápida para el ocultamiento • Buena para disimular contenido oculto (solo se oculta 1 bit por byte).	• Rápida para el ocultamiento • Se usan 4 bits para almacenar información (se puede almacenar mucha información) • La más rápida para la encripción	• Tiempos casi idénticos a LSB1 para recuperar los archivos • Muy buena para disimular contenido oculto (en caso de que LSB1 sea evidente, LSBI puede ayudar)
Desventajas	• No es la más rápida para recuperar archivos • Sólo se utiliza 1 bit por cada byte para almacenar información (se puede almacenar menos información). • Al siempre ocultar información en el mismo bit puede ser fácil identificar cuando se usó este método.	• Introduce mucho ruido a la imágen. Va más allá de lo que podría ser una imagen pixelada. Es evidente que hay algo oculto.	• Lenta para el ocultamiento • Tiene capacidad para almacenar información en el mismo orden que LSB1

Para los 3 métodos el caso extremo no fue muy alentador pero la idea de estos métodos es aprovechar el ruido visual que ya de por sí tienen las imágenes. Se deben evitar imágenes con zonas uniformes de un solo color.

Pregunta 3

Dados los archivos definitivos de la cátedra, lo primero que hicimos fue analizar estos con un editor de hexadecimales, para ver si había algún tipo de

patrón o mensaje oculto adicional dentro de estos. En el archivo *lima.bmp* encontramos un mensaje en texto plano sobre el final de este que decía “la password es camaleón” (ver *Figura 1* en Anexo).

Nuestra primera hipótesis era que este archivo era el encriptado, entonces se intentó probar con varios métodos y modos, variando también el algoritmo de estenografiado. Nuestras pruebas no tuvieron éxito, y comenzamos a analizar los demás archivos probando primero la extracción sin encriptar. Así pudimos extraer del archivo *hugo.bmp* un archivo pdf embebido, utilizando el algoritmo de LSBI. Este nuevo archivo contenía el mensaje “al .png cambiarle la extensión por .zip y descomprimir”.

Comenzamos entonces a pensar que uno de los archivos restantes estaba encriptado, mientras que los otros nos darían pistas para resolverlo. Optamos por analizar el archivo *titanic.bmp* y logramos extraer una imagen png (ver *Figura 2* en Anexo), mediante el algoritmo de LSB4. Recordando el mensaje anterior, cambiamos la extensión del archivo para transformarlo en un zip y encontramos que al descomprimirlo se revelaba un archivo .txt con instrucciones de como leer el tablero de la figura (ver *Figura 3* en Anexo). Pasando todas las banderas a 1, todos los demás números a 0 y concatenando los bits 01 al principio, pudimos obtener que el tablero decía “des ecb”.

Finalmente, extrajimos del archivo *secreto1.bmp* un video oculto, utilizando LSB1 con encripción DES, modo ECB y la password camaleón. Con esto finalizamos nuestro análisis de los archivos.

Pregunta 4

El archivo oculto dentro de *titanic.bmp* contenía una imagen png, pero al cambiar la extensión de esta a zip pudimos ver que en realidad contenía un archivo txt. Este ocultamiento se hizo concatenando el archivo zip al final de la imagen, como se puede ver utilizando un editor hexadecimal donde se ve el nombre del archivo txt (ver *Figura 4* en Anexo). Además se puede ver como el archivo zip tiene el mismo peso que la imagen pero el txt solo pesa 1kb, por lo cual el comprimido tiene más tamaño que el original.

Pregunta 5

El archivo oculto dentro de *secreto1.bmp* era un video con extensión wmv. En este se muestra una escena de una serie policial donde se analiza un mail con un archivo adjunto donde se señala que este “es más grande de lo normal” y que “el método de compresión debió fallar”. A nuestro parecer,el método utilizado debe ser uno similar al mencionado en la pregunta 4.

Pregunta 6

El archivo *lima.bmp* no contenía un archivo oculto utilizando ninguno de los algoritmos implementados, sino que simplemente ocultaba el texto “la password es camaleón” al final del archivo bmp. Este método no es muy eficaz pues si el mensaje tuviese una extensión más grande, arruinaría la imagen dejando ver que hay algo embebido en esta (algo que la esteganografía busca justamente evitar). Además el simple uso de un editor hexadecimal permite ver el mensaje oculto.

Pregunta 7

Según el paper, las dos propuestas de Akhtar, Khan y Johri son más seguras que un esquema de LSB plano. Adicionalmente, como en estos esquemas hacemos inversiones basadas en la cantidad de bits que fueron invertidos, podemos mejorar la "peak signal-to-noise ratio" o PSNR. Esto significa que vamos a tener una diferencia considerablemente menor entre la imagen original y la imagen alterada que si utilizaramos métodos tradicionales de LSB.

Pregunta 8

El paper deja a criterio del desarrollador como indicar luego que patrones se invierten y cuáles no. Una opción sería escribirlos con LSB1 al final (el primer método indicado por la cátedra). Este sin embargo resulta ser poco práctico, dado que no se sabe en principio cuantos bits leer del archivo portador, y debería por lo tanto indicarse con un carácter especial al final (como un \0). El segundo método (el implementado) es ocultar los 4 bits al principio, mucho más práctico de leer y permite además leer e invertir bits en paralelo, obteniendo el mensaje tal cual fue escrito sobre el portador (considerando el primer esquema). Otra opcion podria ser ocultarlo con LSB4 sea al principio o al final del archivo, reduciendo la cantidad de bytes necesarios (paso de 4 a 1), pero esto sería poco práctico de parsear y debería indicarse de alguna forma que no se encuentran con LSB1.

Pregunta 9

La idea del segundo esquema se basa en que el recipiente del mensaje a ocultar ya tenga en su poder la imagen que se va a utilizar para el ocultamiento. Con esto en cuenta, se puede intentar mejorar todavía más la calidad de la imagen luego de la esteganografía.

A diferencia del primer esquema, además de hacer la separación en "clases" acordes a los bits dos y tres, queremos hacer una separación adicional en el análisis entre los bits en la primera posición para distinguir si originalmente eran 0 ó 1. De esta manera, tendremos dos categorías anidadas: Las previamente mencionadas de si los bits dos y tres son 00, 01, 10 ó 11, y para cada una de esas categorías ver que

sucede con el bit uno: si era 0 y fue modificado a 1, si era 0 y quedó en 0, si era 1 y fue modificado a 0 o si era 1 y quedó en 1.

Teniendo esto en cuenta, vamos a analizar cuántos fueron cambiados (categoría A y C respectivamente) y cuántos quedaron iguales (categorías B y D). Si la cantidad en la categoría A es más grande que la cantidad en B (es decir, de los bits que eran 0 hubieron más que fueron modificados a 1 que los que quedaron con su valor inicial 0) entonces voy a invertir esos bits para que mayoría de ellos queden con sus valores iniciales. Esta misma lógica se aplica para los casos de las categorías C y D. De esta manera, voy a tener un beneficio de píxeles que no sufrieron cambios de tamaño min(A,B) + min(C, D).

Este beneficio en pixeles va a ser mayor al beneficio que puede proveernos el esquema 1, ya que en el primer esquema podría estar pasando que muchos unos fueron modificados pero pocos ceros, y eso haría que ambos se vean invertidos. En cambio si hago una división para discernir entre los ceros y los unos que fueron modificados y en base a eso hacer la inversión de valores, puedo mejorar todavía más la calidad de la imagen final. En el peor de los casos no habría ventaja sobre el primer esquema, pero presenta una buena ventaja en los otros casos.

Pregunta 10

La desventaja del segundo esquema es que, por el hecho de que uno de los bits que se usa para evaluar si es necesaria hacer una inversión o no está siendo alterado, necesitamos que el recipiente del mensaje tenga la imagen original. Esto agrega una complejidad adicional. Adicionalmente, necesitamos que el archivo tenga al menos 4 bytes adicionales a los necesarios en el esquema 1 para poder guardar cuales de las 8 combinaciones de los bits uno dos y tres fueron invertidas. Esta complejidad no es mayor pero vale la pena mencionarla.

En cuanto a inconvenientes en la implementación se puede mencionar que ahora la extracción implicaria que para el caso de LSBI se debe hacer una lectura en paralelo del portador modificado y el portador original, lo cual es en sí poco práctico y un gran cambio en comparación a los otros algoritmos LSB, dificultando la generalización del código.

Pregunta 11

La implementación del primer esquema de LSBI, una vez comprendido el método, presentó alguna serie de dificultades. En primer lugar, era muy difícil generalizar este algoritmo con los otros, teniendo que hacer chequeos y pasos adicionales. Un error frecuente en desarrollo fue que al realizar el conteo de bits modificados por cada patrón (00 01 10 11), se comenzaba recorriendo el cuerpo del archivo bmp sin saltear los primeros 4 bytes, destinados a albergar los bits de patrones. Otra dificultad fue en la extracción, ya que ahora dependiendo del algoritmo se podría encontrar en los primeros bytes del cuerpo del portador sea el

tamaño (encriptado o real) sea los 4 bits de patrones. Habiendo mencionado esos casos, se utilizaron las utilidades ya creadas de escritura LSB1 para embeber con LSBI, invirtiendo los bits que fuesen necesarios según el análisis previo.

Pregunta 12

Una extensión predecible sería implementar el segundo esquema de LSBI, dando la oportunidad al usuario de elegir cual utilizar. Otra podría ser utilizar otro tipo de imágenes portadoras, como puede ser un archivo png (con las dificultades que conlleva eso).

Otra cuestión que sería deseable es analizar el programa para detectar posibles vulnerabilidades. Durante el desarrollo se realizó un cierto nivel de programación con la intención de evitar vulnerabilidad con negligente costo en la performance del programa (cómo reiniciar las variables locales y las estructuras donde fueron guardados el archivo portador y el archivo oculto poniendolas en 0). También se hizo uso de métodos cómo strncopy para evitar buffer overflows. De todas maneras, es necesario ampliar este esquema de seguridad a todas las funciones que lo requieran y también equiparar los tiempos para que no dependan ni del archivo que se desee ocultar ni la forma de la que se lo desee ocultar evitando flujos de información.

Anexo

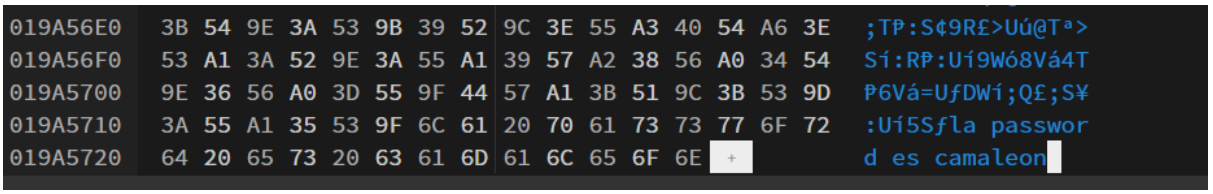

Figura 1: mensaje oculto en lima.bmp

Figura 2: archivo oculto en titanic.bmp

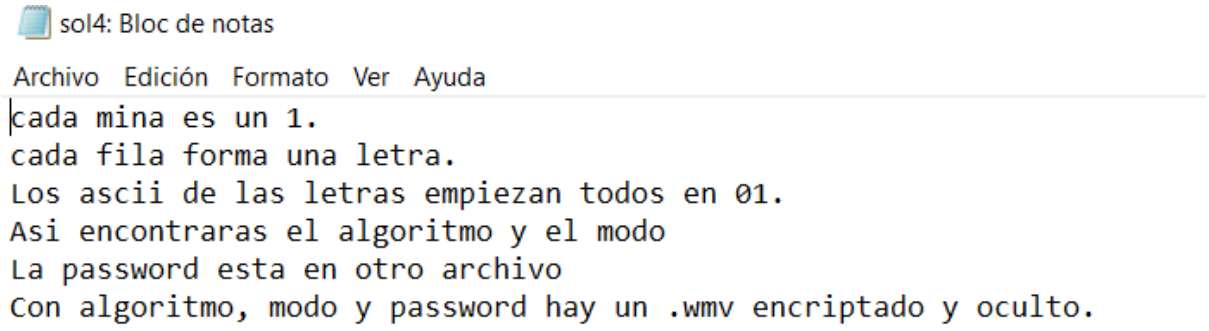

Figura 3: archivo oculto en zip creado con Figura 2

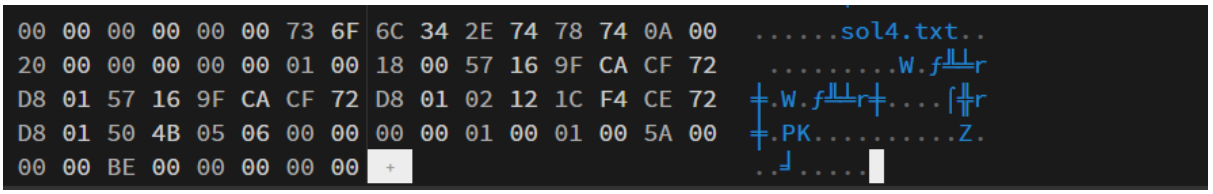

Figura 4: nombre de archivo txt comprimido en png

	Real		Usuario		Sistema	
Oculta	Media	Desvío	Media	Desvío	Media	Desvío
eLSB1	0.027	0.001	0.024	0.003	0.003	0.003
eLSB4	0.03	0.001	0.028	0.003	0.002	0.003
eLSBI	0.091	0.002	0.088	0.003	0.003	0.003
xLSB1	0.041	0.001	0.039	0.003	0.002	0.002
xLSB4	0.038	0.001	0.036	0.003	0.002	0.003
xLSBI	0.041	0.001	0.039	0.002	0.002	0.002

Tabla 2: *Tiempos de los algoritmos de ocultamiento LSB sin encriptar para un archivo bmp con todos sus píxeles en 0 y un texto cotidiano para ocultar. 1000 corridas.*

	Real		Usuario		Sistema	
Oculta	Media	Desvío	Media	Desvío	Media	Desvío
eLSB1	0.026	0.002	0.024	0.001	0.002	0.003
eLSB4	0.031	0.003	0.028	0.005	0.003	0.003
eLSBI	0.093	0.001	0.09	0.003	0.003	0.003
xLSB1	0.041	0.002	0.039	0.003	0.003	0.002
xLSB4	0.04	0.001	0.037	0.003	0.003	0.004
xLSBI	0.041	0.001	0.04	0.002	0.001	0.002

Tabla 3: *Tiempos de los algoritmos de ocultamiento LSB sin encriptar para un archivo bmp con todos sus píxeles en 0 y un texto con todos caracteres 1111 1111 para ocultar. 1000 corridas.*