2024

Informe Grupo-01

Criptografía y Seguridad
Esteganografía

Integrantes:

Agustín Zakalik - 62068

Santino Augusto Ranucci - 62092

Juan Ramiro Castro - 62321

Maximo Gustavo Rojas - 62353

Introducción

Con este trabajo práctico, aprendimos e implementamos 3 distintos sistemas de esteganografiado utilizando el lenguaje de desarrollo C. También, se utilizó la librería existente de openssl para realizar un sistema de encriptación que soporta 4 distintos mecanismos de encadenado y cifrado para agregar una capa extra de confidencialidad a la información escondida.

Uno de los sistemas de esteganografiado, es el investigado y escrito por Mohammed Abdul Majeed y Rossilawati Sulaiman en el paper de investigación: *"An Improved LSB Image Steganography Technique Using Bit-Inverse In 24 Bit Colour Image"*. El objetivo de este algorítmo es esconder la información entre el 2do y 3er bit menos significativo de cada byte de color de una imagen mediante un análisis de los patrones que presentan estos dos bits.

El programa implementado tiene como objetivo:

1. Esconder la información de un archivo en una imagen del tipo .bmp mediante el método LSB1, LSB4 y LSBI.
2. Encriptar la información que se desea esconder mediante AES128, AES192, AES256 y DES utilizando los distintos tipos de encadenamiento CBC, CFB, OFB y ECB mediante el uso de una contraseña con salt fijo.

Aspectos relativos al paper

Organización Formal

El paper tiene una buena organización, comenzando con un abstracto que habla de una manera genérica de que es lo que se trata el paper junto a sus objetivos.

Luego, continúa con una introducción de qué es el estegoanálisis, su origen y su diferencia con la criptografía. Habla de cómo la combinación de ambas técnicas es usada a lo largo y ancho del mundo junto a su simpleza y fama.

A continuación, comienza a detallar el algortítmo en si y su implementación, hablando rápidamente de la estrategia del bit menos significativo en un byte de color. Detalla específicamente los distintos casos que pueden ocurrir junto con un pseudocódigo del algoritmo en el cual nos basamos para nuestra solución.

A la vez, habla de los resultados y simulaciones hechas sobre 4 imágenes utilizando histogramas y tablas que comparan entre utilizar el estándar LSB y el método propuesto. Se llega a la conclusión que ambos son extremadamente similares y el cambio en la intensidad y paleta de colores es casi insignificante. Otro método utilizado para analizar el algoritmo, es el de Peak Signal-to-Noise Ratio en el cual cuanto más elevado el número, una mayor calidad en el esteganografía existe.

Finalmente, el paper cierra con una discusión y conclusión que establece las ventajas y mejoras sobre un método estándar de esteganografiado con LSB.

Descripción del Algoritmo

La idea general del algoritmo es fácil de entender siempre y cuando se tenga un conocimiento básico de cómo funciona LSB1. La explicación de por que se decide usar los colores azul y verde pero no el rojo para esteganografía es clara, así como la argumentación a favor de implementar el algoritmo de inversión de bits.

Sin embargo, hubo algunos puntos que identificamos que se podrían mejorar en el paper, como se puede ver en el siguiente extracto.

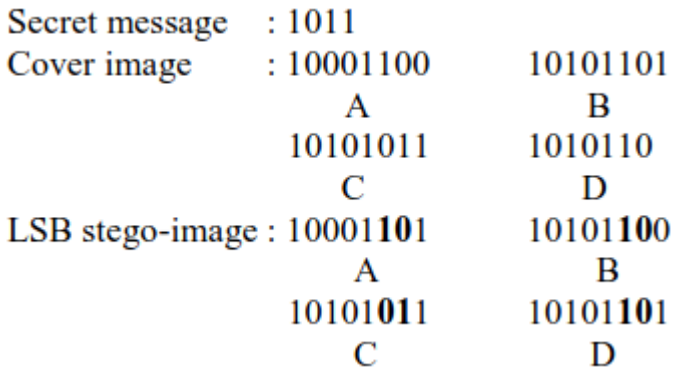
Notación y Claridad

Para empezar, el pseudocódigo no nos pareció nada claro. Tomemos como ejemplo el siguiente extracto:

```
pic = cover image
msg = secret message
For i = 1 to n
      Get char from msg
      For each 2 bits
```

rápidamente surge la pregunta: que es n? La longitud de pic o la de msg? Este es un simple ejemplo de algo que podemos inferir por contexto cuando se analiza el resto del pseudocódigo, pero que complica el entendimiento del mismo.

Por otro lado, reconocimos un pequeño error. Como se puede ver en la siguiente imagen, a “D” le falta el último bit, lo que puede dificultar un poco el entendimiento.



Además, en ningún lugar se especifica dónde se deben esconder los 4 bits que aclaran si se invirtieron los patrones 00, 01, 10 y 11. Tal vez hubiera sido útil una aclaración diciendo que esa parte la dejan a decisión del programador.

Finalmente, en el paper se hace alusión a que LSBI es más “seguro” que LSB1, como se puede ver en el siguiente extracto:

“The proposed method introduces two additional levels of security to the standard LSB steganograph. The first level is that because only the green and blue colours are used, instead of three colors red, green, and blue in the standard LSB, the red colour will act as noise data, and thus increases the complexity of an attacker, when he/she

tries to retrieve the secret message. The second level exploits the new bit inversion technique that reverses the bits of the image pixels after applying the standard LSB”
Sin embargo, esto sin dudas se trata de seguridad por oscuridad. Como aprendimos en la materia, la seguridad de un algoritmo no puede residir en mantener secreto su código, porque esto viola el principio de diseño abierto. Por lo tanto, cuando el paper afirma que se “aumenta la seguridad”, es un error.

Comparación de algoritmos

Decidimos tratar de esconder información primero en el archivo BMP dado como portador de ejemplo por la cátedra, y luego con un archivo BMP en blanco perfecto para maximizar las diferencias visuales, mientras que usamos como secreto un archivo random generado mediante el uso de `/dev/urandom`. Notamos que para esconder un archivo en `lado.bmp`, el BMP provisto por la cátedra, tuvimos que disminuir cada vez más el tamaño del archivo secreto generado para poder embeberlo con LSB1 y LSBI

	Ventajas	Desventajas
LSB1	Menor cambio a imagen original	Requiere archivos portadores muy grandes, o esconder poca información
LSB4	Puede esconder mucha más información en el mismo archivo	En partes con poco cambio, como un fondo blanco perfecto, es fácil notar la diferencia donde hay y no hay información
LSBI	Mejor fidelidad a la imagen original	Requiere archivos portadores incluso más grandes, pues no utiliza un tercio de los píxeles disponibles Requiere más tiempo de procesamiento (negligible)

Análisis de Archivos

El primer paso fue a fuerza bruta intentar extraer de cada archivo con los tres métodos de estenografía: Así conseguimos la foto de minesweeper utilizando LSB4 en `loimposible.bmp`, y luego con LSBI sobre `buenosaires.bmp`. Utilizando LSB1 en `maverick00.bmp` conseguimos un archivo que no podía ser abierto, e intentando abrirlo con `open extracted_maverick00_lsb1` lo marcaba como un stream de bytes. Con `bogota.bmp` ninguna de las formas resultó en un archivo válido, y debuggeando podíamos ver que el size que supuestamente estaba al principio del payload oculto no era un número válido, así que supusimos que tenía información oculta de otra forma. La siguiente acción a tomar fue correr `strings bogota.bmp`, por si había algo oculto en vista plana, y efectivamente al final de todo el archivo estaba oculto el string “la password es camaleon”.

Abriendo el archivo extraído de loimpossible, encontramos un PNG de un juego de minesweeper. Luego, abriendo lo extraído de buenosaires encontramos un PDF que nos indicaba que cambiemos la extensión del .png a .zip y luego de hacer eso y extraerlo, nos encontramos con un archivo `sol1.txt`, que nos indicaba cómo proceder para poder extraer el archivo escondido de maverick00.

El archivo extraído de `loimpossible.bmp` es al mismo tiempo un PNG de un juego de Minesweeper, que a su vez es un mensaje encodeado utilizando las celdas vacías como 0 y aquellas que tienen bombas como 1, y también es un archivo ZIP valido. Esto es posible dado que el principio del archivo es un PNG valido, y luego el archivo ZIP tiene al final de todo un índice indicando donde se encuentra cada pedazo de información, permitiendo que al appendear los dos archivos en uno en el orden de PNG y luego ZIP, funcione como ambos al mismo tiempo, pues PNG ignora los bytes extra cuando termina la imagen original.

La porción de la película muestra como sobre un gran telar se guarda un mensaje secreto, encodeado en binario. Mediante los hilos, viendo su tejido, cuando un hilo pasa por sobre dos hilos al mismo tiempo se toma como un 1, y cuando pasa por debajo se tomaba como un 0. Mientras que es eficaz en el sentido de que dependiendo del tamaño del telar, la probabilidad de que alguien lo note disminuye (en la película los veían mediante una lupa) también no es eficaz en otros aspectos. Si el hilo se llega a cortar se pierde todo el mensaje. Además, aunque se tenga el medio y se conozca la clave, es muy fácil cometer un error de transcripción ya que se podría saltear un hilo y hacer que el todo lo recuperado no tenga sentido. Finalmente la película oculta “un nombre” que representa un objetivo en el telar que se muestra.

El Algoritmo Propuesto

Inicialmente, la propuesta de Majeed y Sulaiman es una mejora respecto de LSB común ya que mantiene el cambio en la intensidad de colores de la imagen base casi idéntica, por lo que es mucho más difícil de darse cuenta que oculta un mensaje estenografiado a simple vista.

El análisis de “Peak Signal-to-Noise Ratio” realizado en el paper nos permite comprobar que la calidad de la imagen portadora se mantiene mucho mejor en comparación a LSB1.

Finalmente, manteniendo la intensidad de colores, como vemos en el paper, también dificulta darse cuenta que hay un mensaje estenografiado.

Implementación y Mejoras

En la implementación se optó por guardar los patrones invertidos antes del mensaje ya que podría ser guardado como parte del header en vez de en los mismos bytes de colores. Esto es posible debido a que el tamaño del header es dinámico gracias al valor del offset guardado en el byte 10 del header. Esto permitiría reducir aún más la notoriedad de modificaciones a la vista del contenido, aunque sería más fácil reconocerlo mediante el análisis del binario.

Por el lado de la implementación del algoritmo, el pseudocódigo es bastante extenso y confuso, teniendo una explicación más corta y menos clara. Otra dificultad es que tuvo que asegurarse de evitar los píxeles correspondientes al rojo, lo cual requiere mantener un estado global mientras se recorre el archivo para saber a qué color pertenece cada byte en específico en distintas funciones.

Relacionado a nuestro trabajo, una de las principales mejoras sería la calidad de código. No modularizamos bien las funciones de extracción y terminamos con bastante comportamiento repetido. Durante el inicio, pensamos realizar una división lógica del embed y extract para así separar el código y comportamiento para ambos modos. En vez de tener un parser y diferentes funciones, podríamos haber realizado una para cada uno.

Otra mejora del trabajo es optimizar la implementación de LSBI. Debido a que el paper original no proporciona una implementación directa, tratamos de separar en distintos pasos la escondida del archivo lo más posible. Al realizarlo de esta manera, terminamos recorriendo más veces de las necesarias el archivo para hacer el proceso de inversión.

Conclusión

Durante el desarrollo del trabajo práctico, aprendimos diversas técnicas de estenografía y logramos comprenderlas en absoluto tras programarlas en C. Asimismo, tuvimos la oportunidad de implementar un algoritmo siguiendo directamente las instrucciones de un paper académico, cosa que no habíamos hecho antes en la carrera.

Otro aprendizaje importante que tuvimos fue el de la librería openssl, que nos permitió comprender cómo utilizar distintos algoritmos de encriptación, con diferentes formas de encadenamiento vistas en clase en C. Consideramos que fue una experiencia muy enriquecedora, dado que siempre es útil bajar a la práctica los conceptos teóricos que se ven en clase. Esto asegura que si en nuestras carreras profesionales necesitamos usarlos, no tendremos problemas al hacerlo.

En conclusión, aprovechamos este trabajo para poder ganar un mayor entendimiento de conceptos de criptografía con los cuales posiblemente nos crucemos en nuestro futuro profesional, de una manera entretenida e interesante.