Trabajo Practico de Implementación
Secreto Compartido con Esteganografía

72.04 - Criptografía y Seguridad

Eugenio Damm - 57094
Ignacio Grasso - 57610
Martin Craviotto - 57756

- 2021 -

Índice

1. Introducción

La idea detrás del siguiente documento es detallar la implementación del programa desarrollado para el trabajo práctico de implementación acerca del tema de esteganografía, cuyo objetivo principal es intentar distribuir y/o hallar una imagen secreta entre un conjunto de imágenes que contienen un secreto. Para ello, es necesario primero entender que es esteganografía, y luego, entender qué algoritmo se utiliza para realizar tanto la distribución como la recolección de la imagen secreta. Comencemos por esteganografía.

El término esteganografía proviene de las palabras griegas steamos (oculto) y graphos (escritura). Es una técnica que permite enviar y entregar mensajes camuflados dentro de un objeto o contenedor, de forma que no se detecte su presencia y consigan pasar desapercibido. Si bien el concepto de esteganografia es amplio y tiene muchas aplicaciones distintas, nosotros nos vamos a centrar en el uso de imagenes de tipo BMP, en blanco y negro, con una densidad de 8 bits por pixel. En particular, vamos a utilizar el esquema de Shamir como procedimiento para realizar el ocultamiento, así como también el descubrimiento de los datos secretos.

2. Solución a las imágenes propuestas

La cátedra nos facilitó un set de imágenes con un secreto oculto, pero sin referencia de que podría ser este secreto. La única información pertinente que se nos dio fue que el valor correspondiente a las sombras requeridas era exactamente la cantidad de imágenes provistas, de modo que en nuestro caso, al recibir 5 imágenes, el parámetro correspondiente al número de sombras era nada más y nada menos que 5.

De esta forma, podemos ejecutar nuestro programa utilizando como parámetros el valor de *k = 5*, *Img-Catedra/* que es el directorio que contiene las imágenes provistas, y *recovered/out.bmp* que es donde se depositara la imagen obtenida luego de analizar las 5 imágenes del directorio.

```
→ TP-Cripto git:(main) ✗ ./TP_Cripto r recovered/out.bmp 5 Img-Catedra/
```

Una vez finalizado el algoritmo, podemos ejecutar el siguiente comando para visualizar el archivo resultado:

```
→ TP-Cripto git:(main) ✗ eog recovered/out.bmp
```

Y observamos la siguiente imagen oculta:

3. Cuestiones a analizar

3.1 Relativos al documento

El documento provisto acerca del Sistema de Imagen Secreta Compartida con Optimización de Carga resultó ser de mucha utilidad a la hora de comenzar la implementación del trabajo práctico. A lo largo del mismo encontramos una distribución del contenido muy organizada, con secciones bien definidas, permitiendo tener una lectura ordenada y poder enfocar la lectura en esas secciones.

En particular, tanto para la etapa de codificación como de decodificación, es de gran ayuda tener los esquemas detallados de la representación visual de cómo funciona el algoritmo, como también la descripción detallada que hace respecto de las asignaciones de bits y armado de bloques. De todas formas, si encontramos algunos problemas como por ejemplo la existencia de algunas fórmulas que no están en un formato correcto y faltan signos o los exponentes no están claros.

Por otro lado, como en nuestro caso particular, no optamos por utilizar el algoritmo de obtención del polinomio propuesto en el documento, sino que optamos por la implementación de un algoritmo de interpolación de Lagrange ya que era más conocido para nosotros, y en esta sección del documento es donde encontramos la mayor cantidad de errores en formulas, haciendose dificil de seguir y de querer implementar sin correr riesgos de errores.

3.2 Optimización de la carga útil

A partir de la información recolectada en el documento acerca del concepto del payload en un esquema como el que se propone, determinamos que la idea fundamental detrás del uso de un algoritmo de este estilo, es aprovechar al máximo la información contenida en las sombras, como también en los datos a ocultar. La carga útil o payload es la relación que existe entre las propiedades propias de las imágenes de camuflaje que se utilizan, conjunto con los valores de k y n del esquema de Shamir. Como se observa en los gráficos del documento, hay una relación lineal entre cuánta información útil hay contenida en las imágenes camuflaje, respecto del tamaño original de las mismas. Cuando nos referimos a información útil, nos referimos a información que podemos utilizar para ya sea esconder, o recuperar posteriormente el secreto. De este modo, cuanto mayor es el payload que obtenemos con el algoritmo, menor es el tamaño de las imágenes de camuflaje que necesitamos para lograr esconder un secreto en particular.

En la figura comparativa que se muestra en el documento, se compara el PSNR (peak signal-to-noise ratio) para 5 algoritmos distintos, entre los cuales encontramos el utilizado en esta implementación, y donde podemos ver como el propuesto tiene el valor mayor de PSNR. Ahora bien, ¿es bueno tener un valor mayor de PSNR?. ¿Qué es PSNR?.

Como dijimos, PSNR significa Peak Signal-To-Noise Ratio, osea, proporción máxima entre señal y ruido. Este dato es importante para nuestro estudio del algoritmo ya que dicho de otra forma, lo que representa este valor es, de una forma cuantitativa, que tan certero es un algoritmo para reconstruir un dato a partir de otro, ya sea un dato comprimido, un dato oculto, etc. Entonces, a mayor PSNR, mejor es el aprovechamiento de los datos ocultos en las imágenes de camuflaje, osea, mejor sería de alguna manera la utilidad del algoritmo en términos generales.

Por último, existe una relación directa entre el valor de k y el tamaño de la imagen portadora en sí, ya que k representa la cantidad mínima de imagenes camuflaje que necesito para lograr recuperar el dato secreto, entonces, necesitaré como mínimo analizar *k bytes * el tamaño de la imagen* para poder recuperarla correctamente. La máxima carga útil de información puede ser controlada por el valor k en si, y el tamaño de las imágenes de camuflaje. Si el valor de k es mayor que 4, el tamaño máximo de los datos secretos puede ser mayor que el tamaño de las imágenes de camuflaje, permitiendo una transmisión eficiente de los datos a ocultar.

3.3 Campos de Galois

En el esquema utilizado, tanto la codificación como la decodificación se realizan en función de cálculos dentro de un campo de Galois 2^8, en vez de, por ejemplo, calcular de forma modular como lo hacen otros algoritmos conocidos de esteganografía. ¿Cuál es la ventaja de utilizar un campo de Galois? En los cálculos que se realizan sobre los polinomios generados, se puede evitar realizar truncamientos de los valores dados por los píxeles en cada paso, y esto significa que puedo realizar una reconstrucción perfecta de la imagen secreta (o cualquier dato secreto de esa manera), sin tener datos aproximados o mal truncados por problemas de tipos de datos.

3.4 Polinomio generador

Si, es posible utilizar otro polinomio generador. El mismo documento aclara que se realiza a partir del polinomio equivalente a 283 en binario, mientras que la consigna nos pide utilizar el polinomio equivalente a 355 en binario, pero esto no altera el funcionamiento ni la implementación del algoritmo en sí, simplemente es cuestión de realizar las operaciones del campo de Galois respecto de otro polinomio.

Si, podría guardarse el valor del polinomio como clave ya que el

3.5 Tipos de archivos

Justamente, como mencionamos anteriormente, la utilización del campo de Galois me permite obtener los datos secretos sin pérdida de información, y siendo transparente que tipo de dato en particular se utiliza, ya que la implementación no diferencia por tipo de dato, simplemente maneja bytes y los recupera con su adecuado análisis polinómico.

3.6 Archivo de imagen completo (encabezado y cuerpo)

Considerando que para el procesamiento del algoritmo se alteran el orden de lectura de los bytes del archivo, sin incluir al encabezado, en caso de querer agregar a los datos secretos el encabezado, debería ser cuidadoso a la hora de hacer tanto la codificación como la decodificación, ya que ambas lecturas y procesamientos son distintos. Debido al hecho de que el encabezado es fijo, podría hacerse un parser distinto para el encabezado que replique los bytes codificados con otra lógica ajustada a la forma que el encabezado se transmite en el archivo, y de igual forma para cualquier archivo BMP con su encabezado ya que la estructura suele ser similar (siempre y cuando se realice en función de los datos presentes en el encabezado en sí como ancho, alto, colores, etc.)

3.7 Imágenes en color

Considerando que todo el trabajo es realizado a partir de tener 1 byte por píxel, en caso de cambiar a tener 24 bits por píxel, osea que cada píxel está formado por 3 bytes, habría que ajustar la asignación de bits y manejar todos los bytes al triple de lo actual. Por otro lado, en términos de campo de Galois, utilizar un campo 2^8 es igual de válido ya que sigue manejando valores de a byte entre 0 y 255. De esta manera, como mencionamos anteriormente, en términos generales el procedimiento es idéntico y funcionaría de igual manera.

3.8 Matrices

Creemos que en caso de querer optar por otro tipo de implementación donde se utilicen matrices que no sean de 2x2, habría que cambiar toda la lógica de asignación de bits del byte “X” (byte 0), por ejemplo si fuesen matrices de 3x3, la cantidad de bits que se deben extraer de los bytes 1-8 debería ser distinto, y mismo el bit de paridad se podría calcular distinto.

3.9 Algoritmo implementado

A nosotros el algoritmo nos pareció más complejo de entender desde un punto de vista teórico, que la implementación en sí. Tuvimos varios días de lectura y de pruebas a mano hasta lograr comprender en completitud el flujo de bytes desde la imagen a esconder hasta que estuviera distribuida, y mismo del lado inverso. Debido a que preferimos entender mejor de antemano el algoritmo antes de implementarlo, esto significó que a la hora de programar, fue considerablemente más simple hacerlo, y mismo al comenzar con la distribución en primera instancia, al momento de implementar la decodificación, ya la lógica inversa resultó ser más simple, y mucho código pudo ser reciclado.

De forma más específica, en los lugares que más complicaciones tuvimos, fue para implementar la interpolación de Lagrange y probar los resultados teniendo en cuenta los campos de Galois ya que no todas las cuentas eran posibles de corroborar de manera directa “en papel”. A su vez, la primera vez que terminamos la implementación del flujo completo donde veíamos que podíamos codificar y decodificar imágenes recuperando bien los secretos, luego cuando probamos los archivos dados por la cátedra, nos salían completamente con ruido, y era imposible ver una imagen real. Allí tuvimos que empezar a hacer debugging específico y comprendimos que estábamos leyendo las imágenes inversamente, entonces los polinomios se evaluaban distinto a lo correcto.

Este programa creemos que es extensible para que no solo corrobore que los archivos recibidos sean BMPs de 8 bits desde el encabezado, sino que podría utilizarse para cualquier archivo. A su vez, podría no trabajarse con las limitaciones de que el valor de k sea entre 4 y 6, sino que podría hacerse cualquier valor si se quisiera.

3.10 Aplicaciones del algoritmo

En situaciones reales este algoritmo podría utilizarse para transportar archivos con información sensible por ejemplo en formato pdf en imágenes de poca relevancia por mail para poder tener un nivel extra de seguridad más allá del canal seguro de comunicación que se establece. Claramente para poder ser utilizable, ambas puntas deben tener la manera de codificar y decodificar a partir de los mismos polinomios generadores, teniendo en su posesión la cantidad mínima de imágenes.

4. Bibliografía

https://es.wikipedia.org/wiki/Esquema_de_Shamir

https://web.mat.upc.edu/jorge.villar/esamcid/rep/accs/reportaccessse2.html

https://iagolast.github.io/blog/2016/11/06/campos-galois.html

https://es.wikipedia.org/wiki/Interpolaci%C3%B3n_polin%C3%B3mica_de_Lagrange