Secreto Compartido

Decisiones de Diseño

El proyecto cuenta con 3 módulos principales para la implementación del algoritmo y 2 auxiliares para la ejecución del programa.

BMP Parser

El manejo de imágenes BMP se implementó desde cero siguiendo el formato descrito por THE BMP FILE FORMAT (ualberta.ca). La implementación está pensada para imágenes de 8 bits.

Este módulo ofrece la posibilidad de leer imágenes de un path válido provisto y almacenar en una estructura específica con la información importante de la misma. Al guardar solo la información más relevante, se está descartando todo el mapa de colores como otros atributos encontrados en el header del archivo. Es por esto que, cuando se lee una imagen, se puede pedir que la lectura también devuelva el header completo. Éste será necesario si se quiere modificar una imagen previa o persistir una nueva.

Galois

En este módulo se encuentra todo lo relacionado a realizar las operaciones dentro de un campo de galois (2^8). Están implementadas las 4 operaciones principales (suma, resta, división y multiplicación) junto con la potencia. Si bien por default todas las operaciones son realizadas utilizando el polinomio: $x^8 + x^6 + x^5 + x + 1$, este puede ser cambiado por cualquier polinomio generador válido.

Además de las operaciones básicas, se cuentan con dos funciones más, específicas para el caso de uso del trabajo en cuestión. La primera es la evaluación de un polinomio con coeficientes y valor x pertenecientes al campo. La segunda consiste en una implementación del método de interpolación de Lagrange que se utiliza para el algoritmo de recuperación.

Shared Secret

En este módulo se encuentra la implementación del algoritmo en cuestión.

Primero se tiene un grupo de funciones agrupadas bajo el nombre de shades. Éstas son las encargadas de hacer todo el manejo de las matrices de 2x2 utilizadas por el algoritmo. Permiten la correcta extracción de las mismas (teniendo en cuenta que los archivos BMP almacenan la imagen de abajo hacia arriba). Además, se encarga de manejar toda la lógica relacionada a la distribución y recuperación del valor F(X) en w,v y u.

Luego, se cuenta con la implementación del algoritmo ubicada en shared_secret. En este archivo se desarrolla el procedimiento de distribución y recuperación en cuestión. Para esta implementación se decidió representar al secreto como una cadena de bytes y no limitarlo a que sea una imagen. Esto nos permite reutilizar el algoritmo con mayor facilidad en caso de querer esconder un secreto cualquiera.

Módulos utilitarios

Se cuentan con dos módulos de utilidad.

Por un lado se tiene un módulo que se encarga de todo el manejo de argumentos. Esto permite recibir y hacer una primera validación de los argumentos obligatorios como hacer el manejo de los argumentos opcionales.

Por otro lado, se tiene un módulo encargado de logeo de todo el programa. Esto permite centralizar las decisiones de qué cosas loguear y por donde loguearlas. Esta implementación es una modificación de rxi/log.c: A simple logging library implemented in C99 (github.com).

Main

Main es el encargado de centralizar todas las demás funcionalidades ya mencionadas en función de los requerimientos pedidos en el algoritmo (distribuir sólo data de la imagen y, al recuperarla, suponer que es una imagen con las mismas características que cualquiera de las sombras).

En particular, se encarga de proveer a shared_secret el secreto que se desea distribuir que, en este caso, es la data de la imagen seleccionada. Además, en el caso de la recuperación, se encarga de tomar un header cualquiera de una de las sombras y utilizarla como header del secreto recuperado.

Preguntas

1. Discutir los siguientes aspectos relativos al documento.

a. Organización formal del documento (¿es adecuada? ¿es confusa?)

La organización formal del documento es correcta, se cuenta con una introducción del tema a discutir junto con las alternativas a las que desea superar, una explicación del algoritmo de codificación y de decodificación y por último la sección de resultados. Esto hace que la lectura sea fácil de seguir y comprender. Uno de los mayores problemas del documento es que el mismo ha sido escrito de manera pobre, faltando muchos espacios entre palabras, símbolos en las ecuaciones y teniendo errores de ortografía.

b. La descripción del algoritmo de distribución y la del algoritmo de recuperación. (¿es clara? ¿es confusa? ¿es detallada? ¿es completa?)

La descripción del algoritmo de distribución es muy detallada, y cuenta con el nivel de profundidad suficiente para comprenderlo solamente leyendo el documento. El uso de figuras para complementar el texto es de gran ayuda, permite visualizar de donde se saca y donde se deja la información involucrada en el proceso.

Para el caso del algoritmo de recuperación, en cambio, ayudaría adentrarse un poco más en el procedimiento de interpolación de Lagrange y como utilizarlo con los datos que se deben recuperar de las estego-imágenes para poder obtener finalmente la imagen secreta.

c. La notación utilizada, ¿es clara? ¿cambia a lo largo del documento? ¿hay algún error?

Dentro de la notación, el caso de T e Y es un poco confuso ya que ambos símbolos representan el mismo valor (F(X)) y no parecería haber una clara razón para notarlo de dos formas distintas.

Luego, en las secciones de ecuaciones hay símbolos que faltan y dificultan la comprensión de las mismas. Por ejemplo, en esta imagen se pueden ver múltiples signos faltantes. Si bien se puede intuir que signos son, aumenta la carga cognitiva a la hora de analizar el algoritmo.

$$s_r^j = \left[(\ 1)^k\ r \left(\sum_{i=1}^{k} {}^{r+1} Y'_{i,j} \prod_{q=1, i\neq q}^{k\ r+1} \frac{X_{q,j}}{X_{i,j}\quad X_{q,j}} \right) \right] \mathrm{r}$$

En la ecuación número 7, donde se explica el cálculo de Y' que se utilizara en la interpolación aparece s_1 cuando debería ser s_i.

El lado positivo es que, cuando se explica cómo hacer el proceso de esteganografía LSB, la notación utilizada para los bits que componen a un byte y cómo se distribuyen es muy clara. Esto es importante ya que, en la parte de esteganografía es necesario saber bien qué bits se modifican y cuáles no. Por último, a la hora de calcular el Payload, la ecuación

presentada es un poco confusa, dado que muestra en el numerador un valor 4 que representa el valor k en el esquema elegido, pero esto se entiende por contexto, no porque se aclare específicamente en el documento, inclusive con este valor, se puede ver una evidente simplificación de términos.

$$Payload\ (byte) = \frac{NPixeles \times Ncolor \times 4}{4}$$

2. El título del documento hace referencia a que optimiza la carga útil ¿a qué se refiere? ¿Qué relación existe entre k y el tamaño de la portadora?

El documento define la carga útil como la capacidad de ocultamiento que posee el esquema elegido. Esta capacidad hace referencia a la cantidad de bytes máxima que puede tener el secreto que se desea distribuir. Este valor es calculado de la siguiente forma.

$$cargaUtil\ = \frac{Npixel * Ncolor * k}{4}$$

Donde Npixel es la cantidad de pixeles de la imagen y Ncolor es la cantidad de bytes por pixel. Estos 2 valores, asociados a las características de las imágenes portadoras y, el valor k vinculado al caso de uso, son los valores que se pueden modificar para aumentar o disminuir la carga útil del sistema. Luego, está el último valor, el 4 en el divisor. Este está asociado a decisiones del algoritmo y es el único al cual se le podría atribuir que optimiza la carga útil.

Este valor 4 hace referencia a la matriz de 2x2 que se toma en cada iteración del proceso de distribución, este indica la cantidad de bytes que están siendo utilizados para almacenar un único byte del secreto. Si bien no encontramos razones claras en el paper que prueben que la configuración elegida optimice la carga útil, tampoco se tiene un parámetro que permita entender en base a qué se optimiza, se puede comprender por qué podría llegar a decirse que lo es.

Yéndonos a un extremo, la carga útil máxima que podría tener una imagen es cuando se utiliza un byte para X y otro para F(X). Esto implica que la sombra solo tendrá la información necesaria para recuperar el secreto original, los pares de valor (x, F(X)) a coste de perder 1 de cada dos píxeles de la imagen portadora original. Lo cual llevaría a la incorporación de alta cantidad de ruido y que la imagen sea altamente sospechosa.

En el otro extremo podemos ubicar el caso donde, para cada par de valores (X, F(X)) se utilizan 9 bloques. Esto implicaría que para distribuir el valor de F(X) se usaría un byte para cada bit. Lo cual genera una reducción radical del ruido de la imagen, se podría llegar a decir que se la mantiene tal cual está (suponiendo que se modifica el bit menos significativo) a costa de tener una capacidad de carga muy reducida.

Para finalizar, si se quiere hablar sobre la optimización de la carga útil, sigue siendo necesario definir cuál es el parámetro de optimización. Luego de suponer que se busca una buena relación capacidad de carga contra ruido de la imagen y, analizar ambos extremos, podríamos estar de acuerdo que la elección de utilizar de a 4 bytes es una opción óptima.

Respecto a la relación entre el k y el tamaño de la imagen. Estos son los dos parámetros que se pueden configurar externamente para ajustar la carga útil en función del caso. Si bien ambos son modificables, cabe destacar que el valor k esta atado al caso de uso y modificarlo puede no ser una opción. Por lo tanto, la mejor forma de regular la carga útil será vía el tamaño de las imágenes. Ya sean imágenes con mayor resolución o mayor profundidad de color.

3. ¿Qué ventajas y qué desventajas ofrece trabajar en GF(2^8) respecto de trabajar con congruencias módulo?

El paper nos dice: “En el esquema propuesto, la codificación y decodificación se llevan a cabo dentro del campo de Galois GF(2^8), evitando así el truncamiento de los valores de los píxeles, esto permite una reconstrucción perfecta de la imagen secreta y cualquier dato secreto.”

La principal ventaja de usar GF(2^8) es que no se pierde información en la distribución. Esto se debe a que el tamaño del grupo es 256 el cual coincide con la cantidad de valores posibles que puede tomar un byte. En otros casos donde se utiliza operaciones de módulo 251, hay 5 valores que se estarían perdiendo ya que, su correspondiente se superpone con otro de los 251 valores.

La desventaja de trabajar de este modo es que las operaciones terminan siendo más complejas y para optimizarlas se debe contar con una tabla estática con los inversos de los números en el campo.

4. ¿Se puede trabajar con otro polinomio generador? ¿Podría guardarse como “clave”?

Se podría trabajar con otro polinomio generador. Este debe de cumplir ciertas condiciones como el hecho de ser irreducible, de grado 8 para este caso dado que se trabaja en ese campo de Galois y contar con cantidad de términos impar y por lo menos una constante. Ejemplos de esto podrían ser p1= 301 = $x^8 + x^5 + x^3 + x^2 + 1$ ó p2 = 285 = $x^8 + x^4 + x^3 + x^2 + 1$.

El polinomio junto al valor de k podrian ser utilizado como claves del sistema ya que ambos valores son necesarios para distribuir y recuperar el secreto. En el caso del valor k puede ser más problemático ya que, probablemente la elección del mismo esté determinada por factores vinculados al caso de uso, lo cual dificulta garantizar una distribución aleatoria.

Cabe destacar que, para que el polinomio pueda ser utilizado, este debería ser un valor configurable y no estar atado a la implementación o el algoritmo porque la seguridad debiera depender solo de la clave y de nada más.

5. Según el documento se pueden guardar secretos de todo tipo (imágenes, pdf, ejecutables). ¿por qué? (relacionarlo con la pregunta 3)

Este documento propone un esquema de secreto compartido donde el secreto a compartir es una cadena de bytes, sin ninguna característica en particular. El hecho de que se usen imágenes es para poder compararlo con otros esquema de secreto compartido que están limitados a imágenes. Por ejemplo, los esquemas basados en criptografía visual permiten recuperar la imagen original haciendo una superposición de las sombras de forma visual, revelando el secreto sin necesidad de hacer operaciones computacionales.

Entonces, teniendo en cuenta que lo que se distribuye es una cadena de bytes simple y no una imagen, esa cadena puede ser un pdf o un ejecutable. Lo único importante es, saber que parte del archivo se está distribuyendo o sino, distribuir el archivo completo junto a sus headers. Hay que tener en consideración que, en función del tipo de archivo que se está distribuyendo, puede ser tolerable o no que se detecte un error de paridad en la reconstrucción. Por ejemplo, en una imagen, si hay un error en un bloque se puede descartar el mismo, ya que su valor no condiciona a los de su entorno. Por otro lado, un error en un bloque de un ejecutable afecta a la totalidad del mismo.

6. ¿Cómo podría adaptarse la implementación realizada para poder guardar un archivo de imagen completo? (es decir guardando encabezado y pixeles)

Dado que lo único que le importa al algoritmos es recibir una cadena de bytes y su longitud, simplemente habría que asegurarse que la carga útil del esquema propuesto sea suficiente para almacenar toda la información (encabezado + pixeles). Esta es la gran ventaja de este algoritmo por sobre los otros mencionados en el paper, la facilidad de extender su uso a cualquier tipo de dato (tamaño o forma). Lo más importante es contar con la cantidad de imágenes portadoras suficiente y del tamaño apropiado.

La posibilidad de distribuir la imagen completa toma más relevancia cuando el secreto es de un tamaño distinto a las imágenes portadoras o en mayor medida, cuando es otro tipo de imagen. Las diferencias entre propiedades que las caracterizan implican que sus encabezados deban ser distintos. Si se desea utilizar el mismo, alguna de las dos imágenes no podrá ser visualizada de forma correcta.

7. Analizar cómo resultaría el algoritmo si se usaran imágenes en color (24 bits por píxel)

El principal efecto de usar imágenes de 24 bits vs 8 bits es que, para una misma cantidad de pixeles, se tiene el triple de bytes para utilizar en la distribución de información. Esto sería equivalente a utilizar imágenes que tengan el triple de cantidad de pixeles.

La principal diferencia es que, teniendo 3 bytes que definen un mismo píxel, se podrían utilizar distintas estrategias para distribuir el secreto. Por ejemplo, se podría trabajar pixel por pixel en vez de matrices de 2 * 2. Donde se toma como X uno de los colores y se distribuye F(X) en los otros dos. También se podría mantener el sistema de matrices de 2x2 iterando sobre la imagen pixel por pixel y, por cada iteración, color por color.

8. ¿se podrían tomar los bloques de otra manera, en lugar de como matrices 2x2? Explicar.

Si, perfectamente. Hay tres consideraciones a la hora de tomar los bloques.

La primera es la cantidad de bits por byte que serán utilizados para distribuir el secreto. En este caso, por cada uno de los 3 bytes utilizados para distribuir el secreto (w, v, u), se utilizan los 3 bits menos significativos de cada uno de estos. Se podrían utilizar distintos esquemas para la distribución de la información. Por ejemplo, si se utilizan matrices de 3x3, se podría tomar el bit menos significativo de cada uno de los 8 bytes restantes y de esta manera, el cambio en las sombras sería prácticamente imperceptible. El problema de un esquema como este es que se reduce la carga útil y que si se quisiese usar el bit de paridad para autenticar los bloques a la hora de recuperar el secreto, habría que tomar el segundo bit menos significativo de alguno de los bytes del bloque y esto debiera definirse de antemano. Por otro lado, en vez de agrandarse el tamaño del bloque, este podría achicarse. Por ejemplo, utilizar 2 bytes para distribuir la información en lugar de 3, teniendo que modificar los 4 bits menos significativos de cada uno de ellos e inclusive más si se desease utilizar el bit de paridad. En este caso, si bien aumenta la carga útil, los secretos distribuidos serán modificados considerablemente, y será fácil detectar la diferencia con las imágenes originales.

Otra alternativa sería no tomar los bytes en forma de matrices de 2x2 sino en su lugar, usar líneas de una determinada cantidad de bytes ya sea vertical o horizontal. Esto solo tendría como diferencia la manera en que se verían distribuidos los bytes modificados dado que así, las modificaciones serán distribuidas menos uniformemente y podría ser más fácil percibir la modificación de las imágenes.

Por último, se podría modificar la forma en la cual se toma un valor de referencia X y con la que se distribuye el fragmento del secreto F(X). El algoritmo propone tomar uno de los 4 bytes de la matriz para usar como X y distribuir F(X) en los otros 3. Una alternativa podría ser utilizar los 2 bits más significativos de los 4 bytes para definir a X y distribuir F(X) entre los 2 bits menos significativos de cada byte. Esta elección complejiza las operaciones pero permite hacer una distribución más uniforme de F(X).

Usando esta última estrategia se solventarían los problemas de utilizar una selección en forma de línea.

9. Discutir los siguientes aspectos relativos al algoritmo implementado

a. Facilidad de implementación

La facilidad de implementación del algoritmo depende de la cantidad de utilidades adicionales que haya que implementar. Si se cuenta con una librería que maneje archivos BMP, otra que realice operaciones en campos de galois con polinomios generadores arbitrarios y otra que realice interpolación de Lagrange, la implementación del algoritmo es bastante sencilla. En el caso de este trabajo práctico, la mayor parte del tiempo se dedicó a la implementación de estas funcionalidades.

Esta facilidad se debe a que el proceso está formado por pocos pasos donde ninguno tiene una alta complejidad. La robustez del sistema recae en la matemática y no en un proceso complicado o tedioso.

Para el desarrollo del algoritmo, el lenguaje de programación C fue de mucha utilidad, especialmente a la hora de trabajar con las posiciones de los bits y realizar operaciones con estos a la hora de distribuirlos en las imágenes portadoras.

b. Posibilidad de extender el algoritmo o modificarlo.

A lo largo de la implementación se nos ocurrieron algunas extensiones posibles para el algoritmo. La más interesante en nuestra opinión consiste en el agregado de padding al secreto para que este pueda ser distribuido con cualquier valor de k. Teniendo en cuenta lo mencionado anteriormente, que el secreto no tiene que tener ningún formato específico, se podría agregar padding sin problemas. Para esto es importante que tanto el algoritmo que recupera y el que distribuye estén en sintonía y sepan manejar el padding.

Otra de las opciones más interesantes consiste en permitir el uso de polinomios generadores a elección. Al igual que el padding, esta opción debe ser utilizada en sincronía al distribuir y al recuperar, de lo contrario el resultado será incorrecto.

Ambas modificaciones mencionadas fueron agregadas a nuestra implementación del algoritmo. La información necesaria para utilizarlas está en el README.

Otras modificaciones posible e interesantes son:

- Utilizar un algoritmo de interpolación diferente. Por ejemplo un sistema de ecuación resuelto con Gauss.
- Utilizar una estrategia diferente para la selección y modificación de bloques de las imágenes portadoras. Con todas las consideraciones mencionadas en la pregunta 8.

10. ¿En qué situaciones aplicarían este tipo de algoritmos?

Este algoritmo es una buena herramienta para aquellos casos donde se quiera distribuir un secreto y poder tenerlo a “simple vista”. Como el secreto distribuido está escondido utilizando esteganografía, las imágenes portadoras no llaman la atención y no son muy distintas a cualquier otra imagen que se pueda tener. Esta cualidad, si bien ayuda a mantener el secreto oculto, hace un uso deficiente del espacio.

Por otro lado este algoritmo, a diferencia de los otros mencionados en el paper, permite distribuir secretos de cualquier tipo, lo cual da la posibilidad de utilizarlo como algoritmo de secreto compartido genérico.

Este tipo de mecanismo para ocultar información podría llegar a ser muy útil si se quiere encriptar algún archivo o dato y que solo sea recuperable juntando una cierta cantidad de personas, repartiendo a cada una de ellas una cantidad específica de las sombras, quizás en base a algun rol que cumplan o permiso que posean o sino, en caso de no existir estas diferencias, uniformemente.