Criptografía y Seguridad

TP02:
SECRETO COMPARTIDO EN IMÁGENES CON ESTEGANOGRAFÍA

Grupo 1:

- Diego de Rochebouët
- José Vitali
- Leandro Llorca
- Juan Marcos Bellini

ITBA
INSTITUTO TECNOLOGICO DE BUENOS AIRES
•UNIVERSIDAD PRIVADA•

Índice

I. Introducción

El objetivo de este trabajo práctico fue ocultar una imagen con información sensible en otras imágenes procurando que esto pase desapercibido. La idea es que a un atacante le sea difícil identificar una imagen que guarda parte de un secreto para revelar otra imagen. Aún así, si un atacante logra identificar imágenes de este tipo, necesita una cantidad mínima de imágenes para descubrir el secreto, y a menos que las obtenga no dispondrá de ningún tipo de información acerca de la imagen secreta.

II. Análisis

Investigación de Luang-Shyr Wu y Tsung-Ming

La implementación realizada se basó fuertemente en la investigación de Luang-Shyr Wu y Tsung-Ming, la cual fue analizada antes de comenzar con nuestra implementación.
El paper consultado está compuesto por una introducción, donde se discute el problema a atacar, una explicación del método de Shamir, que es utilizado para ocultar la imagen, una descripción de un método muy similar desarrollado por Thien y Lin, y finalmente, la descripción del método propuesto en el paper, seguido de resultados obtenidos al aplicarlo.

A continuación se hace una descripción detallada de los algoritmos de distribución y de recuperación.

Algoritmo de distribución:

- Realizar una operación XOR con los bytes de la imagen a ocultar y una tabla R pseudoaleatoria. Se llama Q a la imagen resultante.
- Asignar como 1 a la sección j a procesar.
- Tomar secuencialmente r píxeles no procesados de Q para formar la sección j. Llamar a estos píxeles a0, a1, a2,..., ar-1.
- fj(x)=(a0+ a1x+ a2x^2+…+ ar-1x^r-1) mod 257.
- Generar n pixeles para las sombras evaluando fj(1), fj(2), fj(3),…, fj(n).
- Si fj(x) = 256 para algún x evaluado, se toma el primer pixel del polinomio distinto de cero, se decrementa su valor en 1 y se repite el paso anterior.
- Asignar secuencialmente los píxeles generados a las sombras.
- incrementar j.
- Repetir los pasos desde el paso 3 hasta que todos los píxeles de la imagen a ocultar hayan sido procesados.

Algoritmo de recuperación:

- Asignar como 1 a la sección j a procesar.
- Tomar un píxel sin procesar de cada sombra.

- Usar todos los píxeles fj(1), fj(2), fj(3),…, fj(r) y método de Lagrange para obtener los coeficientes a0, a1, a2 ,…, ar-1 del polinomio. Estos son los píxeles correspondientes a la sección j de la imagen Q.
- incrementar j.
- Repetir los pasos 2 a 4 hasta que todos los píxeles de las sombras hayan sido procesados.
- Aplicar la operación XOR con Q y la tabla pseudoaleatoria para obtener la imagen secreta.

Algoritmo implementado

Diferencia con algoritmo analizado

La diferencia fundamental con el algoritmo previamente analizado es que en la implementación realizada el secreto se distribuye en imágenes previamente existentes, en vez de que se creen imágenes con el propósito de que las mismas sirvan de sombras. La idea es que por cada byte que se quiera ocultar en una sombra el mismo se guarde modificando el bit menos significativo de 8 bytes consecutivos. Lo que esta modificación proporciona es que a un atacante le será mucho más difícil identificar una foto que tenga un secreto guardado.

Fidelidad de la imagen recuperada

La imagen recuperada es muy parecida a la original, pero se ven algunos pixeles notablemente cambiados. Esto se debe al hecho de que como se usa módulo 257 en el polinomio de shamir, al aplicar el polinomio a los distintos números de sombras y obtener 256, se decide cambiar alguno de los términos. Esto se hace como ya fue explicado mirando todos los términos desde a0 hasta ak-1, descontando en 1 el primero que no sea 0. Al modificar los términos, como estos términos representan los bytes de la imagen Q obtenida después del XOR, se está provocando que cuando se quiera recuperar la imagen se vaya a obtener una imagen Q distinta y por ende un imagen secreta también distinta.
Estas colisiones tienen una chance de 1 en 257, por ende los píxeles modificados no van a ser muchos, y entre estos píxeles modificados tan solo algunos van a ser notablemente diferentes que los de la imagen original. Sin embargo, la diferencia está.

Criterio de selección de imágenes portadoras

En el caso de que k sea igual a 8, se utilizó el criterio de la cátedra, que dicta que el secreto y las sombras deben tener el mismo ancho y alto. Para el caso de k distinto de 8, se analizaron varias posibilidades, ya que se buscaba ser lo más flexible posible, mientras que al mismo tiempo, poder tener los datos necesarios para reconstruir el archivo - no sólo la imagen, sino que la metadata también.

Una propuesta a la hora de elegir imágenes portadoras en el caso que k sea distinto de 8 fue nada más dejar usar imágenes que sean cuadradas. El problema que surgió es que la cantidad de bits de las sombras es 8*T/k, siendo T el tamaño de la imagen original. Lo que pasa es que si la imagen es cuadrada, se tiene que dar que 8*T/k se le pueda aplicar la raíz cuadrada. Esto nada más se cumple para algunos valores y entonces no sería una solución efectiva.
Otra opción pensada fue que las imágenes portadoras fueran del mismo ancho que la imagen a ocultar, y tener un largo mayor o menor acorde a lo que se necesite ocultar. También, el offset debe ser el mismo para todas las sombras y la altura de cada sombra debe ser igual a 8 veces la altura del secreto dividido k, y a eso se le suma uno (por un tema del padding). La razón por la cual se elige esto es porque se necesita datos de la imagen secreta que se puedan deducir a partir de las sombras. Con cualquier sombra se puede obtener los datos necesarios para que el archivo bmp sea legible:

- Offset del secreto = offset de la sombra
- Ancho del secreto = ancho de la sombra
- Alto del secreto = (alto de la sombra)*k/8
- Tamaño fila = piso[(ancho_secreto*8 + 31)/32]*4
- Tamaño de la imagen secreta = tamaño_fila*alto_secreto
- Tamaño del archivo = (offset de la sombra + tamaño_imagen)
  + (offset de la sombra + tamaño_imagen)%4

Finalmente, se decidió implementar la segunda opción.

Problemas en la implementación

A la hora de implementar este algoritmo no se presentaron muchos inconvenientes. Se tuvo algunos problemas con respecto a la aplicación del método de shamir para la desencriptación pero fuera de eso se encontró que en general el algoritmo tenía una alta simplicidad.

Extensión a imágenes de 24 bits

Si se tuviesen que esconder los pixeles de imágenes de color que tienen 3 bytes (24 bits por pixel), lo que se podría hacer es esconder cada color del RGB de forma independiente. Escondiendo los bytes del rojo en el último bit de los bytes que representan el color rojo, los de verde en los de verde y los de azul en azul . Se tiene entonces 3 problemas paralelos en los cuales tenemos que esconder una cantidad de bytes igual a la cantidad de pixeles de la imagen original, en una cantidad de pixeles igual a la cantidad de pixeles que tienen las sombras. Esto es equivalente a lo que ya se resolvió. Es entonces un algoritmo que puede entonces extenderse muy fácilmente a imágenes de 24 bits.

Dificultades encontradas

Al comenzar con la implementación, hubo errores a la hora de hacer operaciones modulares, lo que generaba imágenes erróneas. Eventualmente estos errores fueron reparados y se obtuvo un algoritmo que funciona como corresponde.
La gran dificultad encontrada fue determinar qué sistema utilizar cuando k es diferente de ocho. Sobre todo, lidiar con los ceros que el archivo bmp pone automáticamente para asegurarse que cada fila sea múltiplo de cuatro, y que el archivo también sea múltiplo de cuatro. Por ejemplo, al probar con una imagen sencilla de 1 píxel de alto x 2 pixeles de ancho, con k = 2, se esperaba que se guarden dos secretos (cada uno en un archivo) pero al final se guardaban seis, tres en cada archivo. Esto era porque aparte de los dos píxeles que tenía la imagen, se le agregaron dos píxeles más con valor cero para hacer que la fila sea múltiplo de cuatro, y después se le agregaron otros dos bytes más para hacer que el archivo sea múltiplo de cuatro.
Se hicieron muchas pruebas con diferentes tamaños de imágenes para probar como el archivo bmp era generado, y cómo se podía utilizar acorde al problema. Se sabía que, por ejemplo, al utilizar k igual a dos, para esconder dos pixeles de información necesitaba ocho píxeles en cada archivo. Se vio que la relación de cuantos bytes se necesitaban en una sombra a partir de un secreto era igual a la cantidad de bytes de la imagen del secreto, multiplicado por ocho y dividido por k. Y aparte de eso, está el tema del padding, que con widths diferentes se rellenan diferente cantidad de ceros por fila, y el archivo puede terminar no siendo múltiplo de cuatro, y se rellenen más ceros. Si se restringe todo a unos ciertos parámetros de imagen, se puede manejar esto más fácilmente, pero el sistema deja de ser tan usable (una posibilidad era solo trabajar con imágenes cuyo ancho era 4 píxeles). Así que la dificultad fue restringir lo mínimo posible mientras que tenga lo necesario para obtener el archivo original, así el sistema es más utilizable.

Posible modificación al algoritmo

Una posible modificación del algoritmo, sería tratar de minimizar los pixeles modificados por colisiones al encriptar. Para hacer esto lo que se podria hacer seria que, en vez de modificar el primer término, lo que se haga es probar modificar el término que haga que la diferencia con la imagen original sea lo menor posible. Entonces en vez de modificar siempre primero el término a0, haciendo que en algunos casos los pixeles cambien mucho, sino se podria probar ver como quedaría si se cambian los otros términos, de manera separada, y analizar de entre todos los posibles términos a cambiar, de entre todos lo que generan un número distinto a 256 al aplicar el polinomio, el que menor diferencia en el color de píxel con el original provoque. De esta manera, se podrían minimizar la diferencia entre la imagen original y la cambiada, aunque este algoritmo sería más pesado porque se tendrían que analizar muchos más polinomios de shamir.

Ejemplo de aplicación

Una situación en donde este tipo de algoritmos resultaría útil es si se quiere compartir una imagen con información sensible, pero no solo se quiere ocultar el contenido de la imagen, sino que también se quiere ocultar el hecho de que haya una imagen ocultada. Se recibió por parte de la cátedra un conjunto de imágenes que guardaban una imagen secreta. A simple vista estas imágenes parecían completamente normales, pero al realizar la recuperación de la imagen original se obtuvo la siguiente imagen que resultaba ser el lugar secreto donde se reunirían los personajes:

El hecho de que no se note que hay información secreta en esas imágenes puede ser útil si se quiere tener la imagen almacenada sin que posibles observadores ajenos noten la relevancia de esa imagen o el hecho de que lleva internamente un secreto.
Otra posibilidad podría ser que el canal de comunicación por el cual la información es enviada esté siendo vigilado, entonces una imagen encriptada llamaria la atencion. Pero de esta manera, información que a simple vista parece completamente inocente puede ser enviada para que otro usuario la desencripte y la lea.
Por ejemplo, un integrante del grupo extremista ISIS quiere compartir una serie de órdenes secretas a todos los integrantes del grupo. Como muchos de ellos se encuentran en redes altamente vigiladas y no quieren levantar sospecha alguna, ni siquiera de sus conexiones, lo que hacen es compartir sus mensajes secretos ocultandolos en conjuntos de otras imágenes. Estas imágenes no son imágenes aleatorias, pero sino imágenes cómicas conocidas como “memes” que comparten en las redes sociales más populares del momento. Estas imágenes son entonces son entonces leídas en la red social por todos los integrantes, que fingen estar usando la red como cualquier otro usuario normal que se pasa su día leyendo estas imagenes cómicas. Gracias a la alta viralidad que tienen estas imágenes en el internet, se puede lograr que la información llegue a todos los miembros sin que tengan que pasarselas entre ellos. Los extremistas pueden entonces compartir información sin hablar entre ellos, sin siquiera saber dónde están, y sobre todo sin levantar sospecha. El plan perfecto.
También, se probó el uso del algoritmo para valores de k distintos de 8. Se probó guardar una imagen secreta en otras dos. Las imágenes portadoras son las del avestruz (600px x 1593px), y el secreto es la de anonymous (600px x 398) [notar que 1593 = 398*4+1].

III. Conclusión

Se pudo ver cómo un algoritmo relativamente simple a la hora de la implementación permite esconder una imagen en otras aportando un alto grado de confidencialidad a quien quiere distribuir esa información en forma segura. Se vio como una pequeña modificación al algoritmo propuesto por Luang-Shyr Wu y Tsung-Ming aportó una mayor seguridad ya que al usar fotos ya existentes para guardar un secreto no se levantan sospechas de que la información esté siendo distribuida.