Criptografía y Seguridad

Trabajo Práctico de Implementación: Secreto Compartido en Imágenes con Esteganografía

2015

Integrantes:

Javier Fraire (53023)
Ignacio Rivera (53029)
Federico Tedin (53048)

Preguntas:

Pregunta 1.a:

El documento de Thien y Lin está dividido en siete secciones principales, contando también con un resumen (abstract), agradecimientos y lista de referencias (siguiendo el formato tradicional de *paper* científico). La primera sección es la introducción, que habla en general de los objetivos y métodos usados en el documento. La sección 2 es una breve explicación del funcionamiento de un esquema (k, n) de Shamir. La sección 3 explica cómo se aplicaría el esquema de Shamir en la distribucion de imágenes, con y sin pérdida de calidad. La sección 4 muestra ejemplos del esquema aplicado en imágenes de ejemplo, y la 5 es un análisis de la seguridad del método aplicado. Finalmente, la sección 6 habla de las ventajas del esquema aplicado, y la 7 es una conclusión de los conceptos vistos en el documento.

Pregunta 1.b:

Como se describe en el documento, se utiliza una variante del esquema de Shamir para ocultar las imagenes. En el esquema original, se utiliza tan solo el primer coeficiente del polinomio construido (a_0), para ocultar exactamente un valor numérico, con el resto de los coeficientes generados al azar. En el documento que se discute, se utilizan todos los coeficientes a_0, a_1, a_2, … a_{k-1} para codificar k bytes (píxeles) respectivamente. De ésta forma, se logra ahorrar espacio, ya que utilizando el esquema original, se necesitaría que cada sombra tenga exactamente la misma cantidad de bytes que la imagen original, mientras que con el esquema modificado se necesita solo 1/k veces el tamaño de la imagen original.

El algoritmo de distribución es simple: primero se truncan los valores de los pixeles a 250, luego se permutan al azar, y finalmente se aplica el esquema de Shamir modificado. El algoritmo de recuperación sigue los pasos de forma inversa.

Pregunta 1.c:

La notación utilizada en el documento es clara, y no varía a través del documento. Se utilizan consistentemente los nombres de las variables como ‘r’, ‘n’, ‘p’, etc. a lo largo de las explicaciones.

Pregunta 2:

Para evitar el truncamiento de pixeles mayores a 250 se utiliza un método simple. Luego de la permutación, cada pixel $\mathbf{p}_i$ mayor o igual a 250 se ‘expande’ a dos pixeles: uno con el valor 250, y el otro con el valor 250 - $\mathbf{p}_i$ (en la misma posicion que el píxel original, aumentando la cuenta total de píxeles por uno). Luego, se aplica el esquema de Shamir modificado sobre el nuevo vector de píxeles. Cuando se necesita recuperar los pixeles, se deshace el esquema de Shamir, y se recorre el vector de píxeles obtenido. Cuando se encuentra un pixel $\mathbf{p}_i$ con valor igual a 250, se lo reemplaza por la suma de 250 y el valor del siguiente píxel (y el siguiente píxel es removido). Luego, se realiza la permutación inversa.

El truncamiento a valores menores que 251 se debe a que 251 es el mayor número primo que entra dentro de un byte sin signo. Si se utilizara, por ejemplo, dos bytes para

guardar cada pixel, se podría utilizar el valor 257 como módulo en el esquema de Shamir, para evitar el truncamiento de píxeles.

Preguntas 3 y 4:

Para elegir el tamaño y la forma de ocultamiento en las sombras, se utilizaron ciertos criterios específicos.

Para explicar los criterios, primero se necesitan hacer algunas aclaraciones. El esquema con el que se trabaja es el de Shamir modificado, con **k** sombras necesarias para reconstruir la imagen original, y **n** sombras generadas en total. La imagen original tiene un ancho de W píxeles, y una altura de H pixeles. Definimos el **tamaño original** de la imagen como W * H. Sin embargo, el formato BMP requiere que las filas de píxeles almacenados estén alineados a 4 bytes. Ya que cada pixel ocupa exactamente un byte, se requiere entonces que haya una cantidad de bytes divisible por 4 en cada fila. Definimos el padding de la imagen como la cantidad de bytes agregados a W que se necesitan para llegar a un múltiplo de 4. Definimos entonces el **tamaño real** de la imagen como (W + padding) * H. El tamaño real de la imagen nos informa cuántos bytes de información de la imagen se encuentran en el archivo BMP (aunque no todos los bytes tengan una representación visual). A continuación se dan algunos ejemplos:

- W = 300, H = 300: tamaño original: 90000, tamaño real: 90000
- W = 1050, H = 475: tamaño original: 498750, tamaño real: (1052 * 475) = 499700
- W = 35, H = 35: tamaño original: 1225, tamaño real: (36 * 35) = 1260

Para distribuir la imagen original en las sombras, se requiere que el tamaño **real** de las sombras (*TRs*) sea (con *TRo* siendo el tamaño real de la imagen original):

Si k >= 8 **(Caso 1)**:

$$TRs = \frac{TRo - (TRo \bmod k)}{k} \times 8, \; con\; TRo = Tamaño\; Real\; de\; imagen\; original$$

Si 8 > k >= 2 **(Caso 2)**:

$$TRs = \frac{TRo - (TRo \bmod k)}{k} \times 4, \; con\; TRo = Tamaño\; Real\; de\; imagen\; original$$

En el caso 1, se distribuyen $TRo - (TRo \bmod k)$ pixels utilizando el método de Shamir modificado (notar que la cantidad siempre es divisible por k) a las k sombras, utilizando LSB en el bit menos significativo (aunque sea redundante aclararlo, es necesario para diferenciarlo con el caso 2). En el caso de que $TRo \bmod k \neq 0,$ significa que sobraron menos de k y al menos un pixeles sin procesar. Para ocultarlos, se generan pixeles al azar hasta llegar a k, y se los guarda utilizando nuevamente el esquema de Shamir, pero utilizando LSB en el segundo bit menos significativo (para no interferir con los datos guardados previamente). Notar que esta

operación utiliza siempre exactamente 8 bytes, sin importar la cantidad de pixeles que sobraron (que siempre será menor que k).

En el caso 2, se distribuyen los $TRo - (TRo \bmod k)$ pixels utilizando el método de Shamir modificado a las k sombras, pero utilizando LSB en los dos bits menos significativos simultáneamente (esto significa que se requieren 4 bytes de sombra para almacenar un byte generado). En el caso de que sobren pixels, se aplica el mismo procedimiento que en el caso 1, pero utilizando LSB en el tercer bit menos significativo (para no pisar los datos escritos anteriormente).

Notar que cuando k = 8, se aplica en caso 1, que cumple los requerimientos del enunciado. También notar que el requerimiento de tamaño de sombra es sobre el tamaño **real** y no original, con lo cual se podrían utilizar sombras de distintos tamaños originales en un mismo esquema, por ejemplo: S1: 34x10, S2: 35x20, S3: 36x10, ya que todas resultan en el mismo tamaño real.

Pregunta 5.a:

El algoritmo final no fue fácil de implementar, ya que se intentó (y se logró) utilizar el mismo código para manejar el caso en el que k fuera igual a 8 y los casos en los que no. Se logró utilizar las mismas funciones y operaciones para manejar ambos casos.

Pregunta 5.b:

En el caso de que cada color fuera almacenado individualmente y siempre en el mismo orden (RGB, por ejemplo), sería posible adaptar el algoritmo para poder procesar imágenes a color. El mayor problema a resolver sería, nuevamente, la diferencia entre el tamaño original de la imagen (WxH), y la cantidad de bytes realmente almacenados en el archivo (real). La cuenta se volvería más compleja ya que podría cambiar el tamaño utilizado por color y/o el alineamiento.

Pregunta 6:

En cuanto a la lectura del documento, no hubo problemas para comprender lo que se detallaba. Las descripciones de los algoritmos y de los métodos fueron simples y breves, con lo cual fue fácil interpretarlo y adaptarlo a código C.

El mayor problema de implementación del algoritmo fue lidiar con el alineamiento de 4 bytes del formato BMP, que complicó la forma en la que se realizaban las operaciones de bytes. Esto se debe a que en el documento leído, los autores hablan de imagenes como simplemente una lista de bytes, cada uno con una representación visual, mientras que en formato BMP esta condición no siempre se cumple. Por esta razón se decidió trabajar con el tamaño real de la imagen y no el tamaño original, con el fin de simplificar las operaciones (y para poder recuperar las imágenes provistas por la cátedra).

Otro problema encontrado fue el caso en el que el tamaño real de la imagen no era divisible por k. Se optó por generar pixeles al azar hasta poder evaluar nuevamente el

polinomio de Shamir, y almacenar los resultados en otra ‘capa’ de la sombra (los bits menos significativos que no hayan sido utilizados).

Pregunta 7:

La primera mejora que podría ser implementada en el proyecto es la de evitar el truncamiento de píxeles. El problema que esto generaría, sin embargo, es que sería más difícil especificar un tamaño requerido de sombra, ya que no se sabe cuantos pixeles tendrán valores superiores a 250.

Otra mejora que podria ser implementada es almacenar el ancho y alto de la imagen original en las sombras. Esto se podria lograr, nuevamente, utilizando Shamir, para no tener que almacenar los valores directamente sin encriptar. Con esto sería posible recuperar el tamaño del secreto desde las sombras cuando se intente recuperar el secreto, sin tener que especificarlo por parámetros (ver README.txt).

La mejora ideal sería poder procesar imágenes a color, o incluso imágenes en otros formatos, aunque eso implicaría adaptar el código a otro esquema de datos almacenados.

Pregunta 8:

Como se menciona en el documento, resulta conveniente aplicar este tipo de algoritmo cuando no se quiere mantener toda la información oculta en un mismo lugar. Con el esquema de Shamir propuesto es posible separar una imagen en, por ejemplo, 8 imagenes sombra distintas que podrían ser distribuidas a distintos lugares de almacenamiento.

Imágenes Recuperadas:

Las imágenes a continuación fueron recuperadas a partir de los archivos provistos por la cátedra, utilizando el siguiente comando:

```
cripto -r -secret gX.bmp -k 8 -dir grupoX --verbose --no-permute
```

Donde ‘cripto’ es el nombre del ejecutable, ‘X’ un número de grupo y ‘grupoX’ un directorio con las imágenes sombra.

Grupo 1:

Grupo 2:

Grupo 3:

Grupo 4:

Grupo 5:

Grupo 6:

Grupo 7:

Grupo 8:

Grupo 9: