Trabajo Practico de implementación: Secreto compartido con esteganografía

LEANDRO EZEQUIEL RODRÍGUEZ[1], FRANCO NICOLÁS ESTEVEZ[2], AND LUCAS AGUSTÍN VITTOR[3]

[1] *Instituto Tecnológico de Buenos Aires, Legajo 61069, Mail learodriguez@itba.edu.ar*
[2] *Instituto Tecnológico de Buenos Aires, Legajo 61435, Mail festevez@itba.edu.ar*
[3] *Instituto Tecnológico de Buenos Aires, Legajo 61435, Mail lvittor@itba.edu.ar*

Compilado 22 de junio de 2023

1. INTRODUCCIÓN

En este informe, se presentará una descripción detallada del proceso de implementación del algoritmo de Imagen Secreta Compartida, así como también se analizarán sus aplicaciones en el campo de la criptografía visual y la esteganografía.

Además, se discutirán los resultados obtenidos durante la realización del trabajo y se presentarán las conclusiones correspondientes.

2. CUESTIONES A ANALIZAR

1. **Discutir los siguientes aspectos relativos al documento/paper.**

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

   La organización formal del documento es adecuada y no es confusa. El documento sigue una estructura clara y lógica, comenzando con una introducción que establece el contexto y los objetivos del estudio, seguida de una revisión de la literatura relevante y la presentación detallada del esquema propuesto. Los autores también proporcionan resultados experimentales para respaldar su enfoque y discuten las limitaciones y posibles direcciones futuras para la investigación. En general, la organización del documento es fácil de seguir y ayuda al lector a comprender los conceptos presentados.

   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 y recuperación es clara, detallada y completa. Los autores proporcionan una explicación paso a paso de los algoritmos, incluyendo las fórmulas matemáticas necesarias para llevar a cabo los cálculos. Además, se proporcionan ejemplos numéricos para ilustrar el proceso de distribución y recuperación.

   Nos hubiese gustado a la hora de analizar el paper tener más contenido visual como diagramas o esquemas de como se iban generando las sombras y los bloques. De esa forma quizas hubieramos comprendido más rápido el algoritmo.

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

   El documento utiliza principalmente la notación matemática estándar y es clara en su presentación. No se observan cambios significativos en la notación a lo largo del documento, lo que facilita la comprensión del lector. No se detectaron errores obvios en la notación utilizada.

2. **El título del documento hace referencia a que es capaz de detectar sombras falsas (cheating detection). ¿cómo lo hace? ¿es un método eficaz?**

   El documento hace referencia a que el método es capaz de detectar sombras falsas, puesto que esconde en cada bloque de la imagen secreta, dos bytes en los que oculta un valor generado a partir de un número aleatorio $r \in \mathbb{Z}_{251}$.

   La gracia de este esquema es que el atacante, al generar una sombra falsa, no sea capaz de determinar cuales son bichos bytes (b_0 y b_1) que hacen cumplir siguiente relación bajo el mismo r

   $$(1)\ \exists\ r \in [1,251]\ /\ r * a_0 + b_0 = r * a_1 + b_1 = 0 \pmod{251}$$

   Si bien es cierto que los valores de r son pocos, y un atacante tendría $\frac{1}{251}$ chances de acertar en el r que satisfaga dicha relación, debería poder hacerlo para cada uno de los bloques, lo que hace decrecer notablemente la probabilidad de falsificar una sombra y que no sea considerado como una imagen falsa.

   Podemos ver que es un método eficaz, puesto que la probabilidad de falsificar una sombra exitosamente sería de

   $$(\frac{1}{251})^{\#B} \quad / \ \#B = bloques$$

3. **¿Qué desventajas ofrece trabajar con congruencias módulo 251?**

   Si el esquema utiliza congruencias módulo 251 (lo que nos permite armar un cuerpo finito con 251 primo) para representar los píxeles de la imagen (I), puede haber limitaciones en la resolución y la calidad de las imágenes que se pueden compartir. Esto se debe a que el rango de valores permitidos

se reduce a $[0, 250]$, lo que genera que valores superiores a 250 terminen siendo congruentes a valores muy bajos del cuerpo, creando así bytes muy oscuros en lugar de bytes muy claros.

Fig. 1. Secreto recuperado con k=3 de un conjunto de imágenes de prueba provisto por la cátedra.

Si bien esto no pareciera ser tan importante puesto que seguimos interpretando correctamente la imagen, notamos que no recuperamos el secreto original.

4. **Con este método, ¿se podrían guardar secretos de cualquier tipo (imágenes, pdf, ejecutables). ¿porqué? (relacionarlo con la pregunta 3)**

   En principio, no. Al aplicar congruencias módulo 251, se reduce la información de los datos originales a su residuo módulo 251, lo que significa que se pierde una parte de la información necesaria para representar adecuadamente los archivos completos. Esto afectaría la integridad y la capacidad de recuperar los datos originales sin pérdidas.

   Esto no es de gran importancia si el archivo secreto es una imagen, puesto que una diferencia mínima en el color de los pixeles no nos impediría interpretar el contenido. Ahora, si el archivo es un ejecutable podría verse comprometido su debido comportamiento. Al ser un conjunto de operaciones, modificar este tipo de archivos podría provocar un cambio sustancial en el comportamiento del programa en cuestión. Podría resultar en la falla absoluta del mismo, o en resultados que no son los esperados.

5. **¿En qué otro lugar o de qué otra manera podría guardarse el número de sombra?**

   El número de sombra podría guardarse en cualquier lugar del stream de bytes por fuera de la data de la imagen y de los bytes del header que son importantes para lograr un archivo .bmp bien formado.

   Sería conveniente no elegir un lugar que un usuario común pudiera modificar, como por ejemplo el título de la imagen, y así hacer imposible la recuperación del secreto. Al no tener el número de sombra asociado a determinada portadora, no sería posible reconstruir el polinomio de recuperación.

6. **¿Qué ocurriría si se permite que r, a_0, a_1, b_0 o b_1 sean 0?**

   Considerando (1), observemos lo que podría ocurrir si dichas variables tomaran el valor de 0:

   - Si $r = 0$

     $$0 * a_0 + b_0 = 0 * a_1 + b_1 = 0 \pmod{251}$$
     $$b_0 = 0 = b_1 \pmod{251}$$

     En este caso, bastaría con pedir que ambos bytes sean congruentes a 0 para pasar la verificación.

   - Si $b_0, b_1, a_1, a_0 = 0$

     En dicho caso, la ecuación (1) sería verdadera $\forall r \in \mathbb{Z}_{251}$. Lograría pasar el chequeo de falsificación de sombras sin ningún problema y el esquema se vería comprometido.

7. **Explicar la relación entre el método de esteganografía (LSB2 o LSB4), el valor k y el tamaño del secreto y las portadora**

   El método de esteganografía que se utiliza está íntimamente relacionado con el valor de k, el tamaño del secreto y la dimensión de las imágenes portadoras. Sabemos por el paper analizado que la generación de sombras a partir del secreto nos establece las siguientes relaciones

   $$(2) \quad |S_j| = \frac{|I|}{k-1} \quad / \ I = \text{Secreto}$$
   $$(3) \quad |B_t| = 2k - 2$$

   El método de esteganografía nos va a determinar cuantos bytes de la imagen portadora van a estar ofuscados con información de una sombra. Podría ocurrir que la cantidad de bytes totales de la imagen sea mayor que la cantidad de bytes escondidos en una portadora sin problema. Es importante, ya que habría que considerarlo para recuperar la información hasta esa longitud en particular. Ahora bien, veamos que no es posible utilizar cualquier método para cualquier valor de k y cualquier dimensión de imágenes, tanto secretas como portadoras.

   Supongamos que tenemos un secreto de 300x300 pixeles, en los que cada uno de ellos se ve representado por un único byte. Analicemos qué ocurre al utilizar un esquema (4,8) con LSB2 para distribuir entre portadoras con las mismas dimensiones del secreto. De (2) podemos obtener:

   $$|S_j| = \frac{300 \times 300}{4-1} = 30000$$

Sabiendo que la longitud de cada una de las sombras destinadas a una portadora tendrá una longitud de 30000 bytes, nos damos cuenta que al aplicar LSB2 cada par de bits de un byte de la sombra ira a parar a un único byte de la portadora. Es por esto que cada sombra necesitará 120000 bytes para ser ofuscada:

$$|S_{j_{LSB2}}| = 30000 \times 4 = 120000 \ \wedge \ |P_j| = 90000$$
$$|S_{j_{LSB2}}| > |P_j|$$

Queda claro entonces, que no alcanza la cantidad de bytes en una portadora para esconder una sombra utilizando el método de esteganografía LSB2 bajo este esquema. Necesitaríamos portadoras de dimensiones mayores.

Analicemos lo que ocurre bajo un esquema de (3,8) con LSB4 y con imágenes de 280x440.

$$|S_j| = \frac{280 \times 440}{3-1} = 61600 \implies |S_{j_{LSB4}}| = 61600 \times 2$$
$$|S_{j_{LSB4}}| = |P_j|$$

Por lo tanto, cada byte de la imagen portadora será utilizado para ofuscar 4 bits de la sombra. Si hubieramos utilizado el método LSB2, ocurriría lo mismo que en el caso anterior.

$$|S_j| = 61600 \implies |S_{j_{LSB2}}| = 61600 \times 4 = 246400$$
$$|S_{j_{LSB2}}| > |P_j|$$

Otra cosa interesante que podemos analizar es la relación entre (3) y la cantidad de bytes totales del secreto. Supongamos que tenemos una imagen de 300×300 y queremos utilizar un esquema de (8,8) para compartir el secreto. Rápidamente nos daremos cuenta que esto no es posible, ya que como vamos a dividir el secreto en $|B_t|$ bloques, la cantidad de bytes totales del secreto debería ser divisible por este valor y no lo es.

$$|B_t| = 2 * 8 - 2 = 14 \ \wedge \ \#B_j = \frac{90000}{14} = 6428{,}57$$

8. **Analizar cómo resultaría el algoritmo si se usaran imágenes en color (24 bits por píxel) como portadoras**

   En cuanto a la representación, las imágenes en color de 24 bits por píxel generalmente se representan utilizando el modelo RGB, donde cada píxel se compone de tres canales de color: rojo, verde y azul. Cada canal tiene una profundidad de 8 bits, lo que significa que puede tener 256 valores diferentes para cada componente de color. Para trabajar con congruencias módulo 251, sería necesario reducir cada valor de color (rojo, verde y azul) a su residuo módulo 251, lo que implicaría una pérdida de información y una reducción en la calidad de la imagen original.

   En relación al tamaño del archivo y complejidad, las imágenes en color con 24 bits por píxel tienen un tamaño considerablemente mayor en comparación con los datos binarios. Esto aumentaría la complejidad y el tiempo de procesamiento requeridos para dividir la imagen en sombras y realizar las operaciones necesarias para compartir y reconstruir los datos originales. Sin embargo, para un esquema de secreto compartido de este estilo, será indiferente el tamaño de la imagen, y podremos seguir ofuscando una imagen secreta exitosamente.

   Al utilizar métodos de esteganografía como LSB2 o LSB4, seguiremos modificando los bits menos significativos de cada byte, lo que generara un cambio imperceptible en imágenes a color.

9. **Discutir los siguientes aspectos relativos al algoritmo implementado:**

   a) **Facilidad de implementación** Si bien la implementación fue relativamente fácil, todo algoritmo en el que se deba hacer un manejo a nivel bits debe realizarse con mucho cuidado y detenimiento. Uno de los errores que más tiempo nos llevó detectar fue estar realizando el ocultamiento de cada byte en orden inverso. Es decir, en lugar de comenzar ocultando los bits más significativos del byte de la sombra, comenzábamos con los menos significativos. De esta forma, guardábamos el byte ”espejado” y hacía que la recuperación del secreto siempre fuera analizado como ”cheeting”.

   Otro aspecto que podría haber simplificado la implementación hubiera sido, dado que el trabajo fue realizado en Python, el uso de librerías como Pillow para la generación y manejo de imágenes. Eso podría haber simplificado la incorporación de sombras a los archivos .bmp existentes, o directamente generar nuevas imagenes sin más dificultad de la necesaria.

   b) **Posibilidad de extender el algoritmo o modificarlo.**

   Algo que mencionamos anteriormente y podríamos modificar en una futura versión del algoritmo es la capacidad de contemplar los casos en los que la cantidad de bytes totales de la imagen no es divisible por la longitud de los bloques que buscamos obtener. Podría agregarse un padding para localizar dicha diferencia, agregando un 1 y una cadena de 0s hasta alcanzar el tamaño deseado.

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

    Este tipo de algoritmos se suelen utilizar en situaciones en las que se requiera un consenso de k de los n participantes del esquema para poder realizar una acción o tomar una decisión. Un esquema de secreto compartido como el que implementamos, podría ser una gran herramienta para situaciones en las que cada participante cumple un rol fundamental y de no alcanzar un número mínimo de colaboradores, no se podría acceder a la información oculta. Un ejemplo válido podría ser la implementación de dicho algoritmo para la compartición de información clasificada entre agencias gubernamentales que requiere cooperación de múltiples entidades para obtener acceso a la información completa.

    La mayor ventaja que notamos al mezclar dicho esquema con imágenes como portadoras es hacer que el ocultamiento del secreto pase casi desapercibido. La esteganografía aplicada de esta forma, nos provee la capacidad de mezclar en cualquier medio digital nuestras portadoras con otras imágenes que no forman parte del esquema y solo aquel que conoce de la existencia de un esquema de secreto compartido sería capaz de ubicarla y, de tener acceso a al menos k portadoras, recuperar el secreto.

3. SOLUCIÓN DEL SECRETO DADO POR LA CÁTEDRA

Con el conjunto de datos que se nos proporcionó para verificar que nuestro algoritmo funcionara correctamente, logramos recuperar el siguiente secreto oculto:

Fig. 2. Imagen recuperada por el algoritmo implementado en la que se ve el Cristo Redentor en Brasil