DESAFÍO 1

Autores: *Maria Valentina Quiroga Alzate, Emmanuel Guerra Tuberquia*

Informática II
Departamento de Ingeniería Electrónica y de Telecomunicaciones
Universidad de Antioquia

Resumen

Este documento presenta la propuesta inicial de solución a un reto de reconstrucción de imágenes digitales basado en técnicas de ingeniería inversa. A partir de una imagen final distorsionada (`P_n.bmp`) y un conjunto de archivos intermedios (`M_k.txt`, `I_m.bmp`, y `M.bmp`), se busca identificar y revertir una secuencia desconocida de transformaciones aplicadas a nivel de bits.

La estrategia se enfoca en comparar pequeñas regiones de la imagen, extraídas a partir de posiciones específicas, con los resultados parciales almacenados en los archivos de rastreo. De este modo, se busca deducir qué transformación se aplicó en cada etapa, y en qué orden, para luego aplicar las operaciones inversas correspondientes. El desarrollo será implementado en C++, respetando restricciones como el uso exclusivo de punteros y arreglos dinámicos, con un enfoque modular, iterativo y validado progresivamente.

Introducción

El presente informe tiene como objetivo exponer la propuesta inicial para abordar el Desafío 1. El reto consiste en diseñar una estrategia para revertir una serie de transformaciones aplicadas sobre una imagen digital, utilizando únicamente la imagen resultante y algunos archivos generados durante el proceso de transformación. A través del análisis, la experimentación y la deducción lógica, se busca reconstruir la imagen original aplicando técnicas de programación a bajo nivel, en lenguaje C++.

A lo largo del informe se describen las condiciones del problema, los criterios de comparación necesarios y la estrategia que se plantea implementar para resolver el desafío, así como los retos técnicos que deben considerarse durante la ejecución.

Marco teórico

Formato de imagen BMP: El formato BMP (Bitmap) es un tipo de archivo gráfico utilizado en sistemas Windows para almacenar imágenes digitales sin compresión. Las imágenes BMP son fáciles de manipular a bajo nivel debido a su estructura simple: contienen encabezados que describen dimensiones y profundidad de color, seguidos por los datos de los píxeles. En una imagen de 24 bits, cada píxel se representa por tres bytes, correspondientes a los valores de color en los canales azul (B), verde (G) y rojo (R), en ese orden. El acceso directo a los bytes que componen la imagen permite aplicar transformaciones a nivel de bits, lo cual es importante para tareas como cifrado, ocultamiento de información o compresión personalizada [1].

Representación RGB y organización interna: En el modelo de color RGB, cada píxel está formado por tres componentes: rojo, verde y azul, cada uno con valores entre 0 y 255. Internamente, en archivos BMP de 24 bits, estos valores se almacenan en orden inverso (BGR).

Además, las filas de píxeles se almacenan desde la parte inferior de la imagen hacia arriba, lo que significa que la primera fila en memoria representa la parte inferior de la imagen. Manipular estos valores a nivel de bits permite distorsionar o transformar la imagen de maneras no perceptibles a simple vista, lo que es útil en contextos de seguridad y criptografía básica [2].

Enmascaramiento: El enmascaramiento es una técnica utilizada para alterar una sección específica de la imagen después de cada transformación. Consiste en sumar, píxel a píxel, los valores de una porción de la imagen transformada con una máscara `M`, que es una pequeña matriz de píxeles RGB. Esta suma se aplica comenzando desde una posición aleatoria determinada por una semilla s.

La fórmula general es:

$$S(k) = ID(k + s) + M(k)$$

Donde $S(k)$ es el resultado almacenado en los archivos de rastreo (`.txt`), y representa el único rastro visible de las transformaciones previas. Al conocer M y $S(k)$, se puede intentar deducir $I_D(k+s)$, donde s es la semilla (la imagen antes del enmascaramiento).

Ingeniería inversa de imágenes: La ingeniería inversa en el contexto de imágenes consiste en analizar una imagen transformada y, a partir de datos parciales o manipulados, reconstruir la imagen original. Este proceso implica identificar el orden y tipo de transformaciones aplicadas, validar cada paso mediante los resultados del enmascaramiento y revertir las operaciones en el orden correcto [4].

Estrategia propuesta para la solución

El objetivo del desafío es recuperar la imagen original `I_0.bmp` a partir de la imagen final transformada `P_n.bmp`, conociendo una secuencia de transformaciones que fue aplicada, las cuales están registradas parcialmente en los archivos `M_n-i.txt`, y usando una imagen intermedia aleatoria `I_m.bmp`, además de una máscara `M.bmp`. Como no se conoce qué operaciones específicas se aplicaron en cada paso, se plantea una estrategia de análisis y reversión bloque por bloque, transformación por transformación, evaluando posibles combinaciones y validando con los datos disponibles.

Archivos disponibles

- `P_n.bmp`: imagen final ruidosa, resultado de aplicar n transformaciones.
- `M_n-i.txt`: registros intermedios (uno por cada transformación aplicada).
- `I_m.bmp`: imagen intermedia que puede estar relacionada con las operaciones XOR.
- `M.bmp`: máscara usada en cada etapa del proceso para enmascarar información.
- `I_0.bmp`: imagen original (objetivo final).

1. Establecer el conjunto de operaciones posibles

Se parte de un conjunto finito de transformaciones binarias, que incluyen:

- XOR con bloques derivados de `I_m.bmp` y/o `M.bmp`
- Rotaciones a nivel de bits (izquierda y derecha)
- Desplazamientos a nivel de bits (izquierda y derecha)

Estas se combinan en distintas formas, dando lugar a un total de 32 posibles combinaciones.

2. Comparar bloques con `M_n.txt`

Para cada transformación posible:

1. Aplicar la transformación inversa a `P_n` para obtener una imagen candidata P_{n-1}.
2. En P_{n-1}, extraer un cuadro pequeño en la posición (k, s) (según lo que se sepa de `M_n.txt`).
3. Aplicar la máscara `M.bmp` a ese bloque para simular el enmascaramiento.
4. Comparar el resultado con el contenido de `M_n.txt`.
5. Si coincide, es muy probable que esa sea la combinación correcta para obtener P_{n-1} desde P_n.

3. Ir hacia atrás

Una vez identificada la combinación correcta:

1. Guardar qué transformación se usó para ir de P_n a P_{n-1}.
2. Repetir el proceso con P_{n-1} para buscar P_{n-2}, usando el archivo `M_n-1.txt`, y así sucesivamente.

Aplicación de operaciones

Para cada bloque de `M_n.txt` en cada combinación, se aplican todas las transformaciones posibles (XOR, rotaciones, etc.).

Comparación con las `P`

Se compara el resultado de cada transformación con el bloque equivalente de `P_n` usando una diferencia.

Selección del mejor candidato

Si en algún momento la diferencia entre bloques es cero, se asume que se ha encontrado la transformación exacta, por lo que se sale inmediatamente del ciclo de comparación y se identifica esa operación como la aplicada. Si no hay coincidencia exacta, se guarda la transformación que genere la menor diferencia posible, como mejor candidata.

Conclusiones

Hasta ahora, hemos concluido que es posible reconstruir la imagen original a partir de una versión transformada y con datos limitados. Mediante operaciones como XOR, rotaciones y desplazamientos, junto con la comparación bloque por bloque y el uso de la máscara, hemos logrado avanzar en la identificación de las transformaciones aplicadas. Aunque el proceso requiere prueba y error, la estrategia adoptada ha demostrado ser efectiva. Sin embargo, también notamos que la eficiencia del algoritmo puede verse afectada por la cantidad de combinaciones posibles, por lo que optimizar el análisis se vuelve clave para mantener un buen rendimiento sin perder precisión.

Bibliografía

[1] Microsoft, "Bitmap Storage," *Microsoft Learn*, [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/gdi/bitmap-storage.

[2] The Color Blog, "¿Qué es el modelo de color RGB?," *The Color Blog*, [Online]. Available: https://thecolor.blog/es/rgb/.

[3] Formlabs, "¿Qué es la ingeniería inversa y cómo funciona?," *Formlabs Blog*, [Online]. Available: https://formlabs.com/latam/blog/reverse-engineering/?srsltid=AfmBOooxR7n-FlcFTEICfp95JDeZE_PMk4uG65yRMg