**Informe Preliminar Desafio I**

Marx Eusdav Lopez Montero, Daniel Antonio Londoño Godoy

Facultad De Ingeniería, Universidad De Antioquia

Informatica II: 2598521

12 de abril de 2025

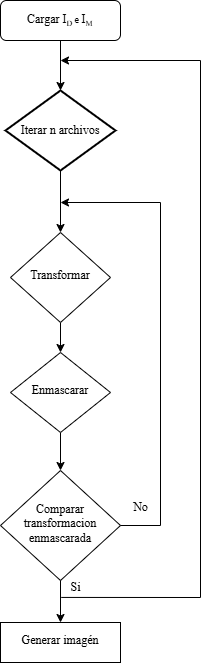
**Introducción**

Este informe presenta la propuesta inicial para enfrentar el reto de ingeniería inversa que se planteó en el curso de Informática II. La idea es diseñar e implementar un programa en C++ que permita la reconstrucción de una imagen BMP original, pero partiendo solo de su versión final transformada y un conjunto de archivos que nos ofrecen pistas sobre el proceso que se usó.

Esta reconstrucción se apoya en operaciones a nivel de bits —como rotaciones, desplazamientos y XOR— que se aplicaron en un orden que no se conoce, además de procesos de enmascaramiento con una máscara de color y desplazamientos aleatorios. El verdadero desafío radica en descubrir el orden correcto de estas transformaciones y revertirlas de manera controlada.

La solución que se construye se basa en criterios de eficiencia y claridad, usando solo punteros, memoria dinámica y arreglos lineales para gestionar los datos. Se valida cada etapa comparando resultados intermedios con la información que nos dan los archivos .txt, lo que permite identificar con precisión las transformaciones originales

**Metodología.**

Para abordar la problemática de reconstrucción de la imágen, se emplea como primera herramienta, un esquema conceptual (Figura 1) y se puntualizan los pasos a seguir de manera general con mayor clarividencia.

1. **Cargar imágenes ID e IM:** Se cargan las imágenes bitmap (BMP) y se almacenan en un arreglo dinámico lineal para su correcta manipulación bit a bit. La máscara se utilizará en operaciones XOR.
2. **Aplicar transformaciones posibles sobre ID o copias:** Sobre la imágen ID se asignan transformaciones a nivel binaria las cuales incluyen: XOR y desplazamientos y rotaciones (tanto a la derecha como a la izquierda). 
   1. **Enmascaramiento y verificación:** Para esto, se carga el archivo Mn.txt, extraemos la semilla y la aplicamos a la fórmula

S(k) = ID(k + s) + M(k) donde 0 ≤ k < i × j × 3

Cada iteración de k se comprobará con el archivo Mn.txt para validar que dicha transformación sea la correcta. Si para todo k se cumple dicha comparación, se deduce que la transformación aplicada es la indicada.

**Descripción y análisis de las funciones a desarrollar.**

A continuación, se describe la documentación de cada función, especificando el tipo de variable y su justificación:

***Desplazamiento izquierda o derecha***

Argumentos:

unsigned short int bits → Cantidad de bits a desplazar (máximo 8)

unsigned char \*ID → Puntero al arreglo de datos de píxeles de ID

return → unsigned char \*transformacion (Puntero al arreglo de datos de pixeles transformados)

Desplaza cada byte del arreglo (ID) la cantidad de bits indicados. Dependiendo de la función será de la izquierda a la derecha.

***Rotación izquierda o derecha***

Argumentos:

unsigned short int bits → Cantidad de bits a desplazar (máximo 8)

unsigned char \*ID → Puntero al arreglo de datos de píxeles de ID

return → unsigned char \*transformacion (Puntero al arreglo de datos de pixeles transformados)

Rota cada byte del arreglo (ID) la cantidad de bits indicados. Dependiendo de la función será de la izquierda a la derecha.

***Enmascaramiento***

Argumentos:

unsigned char \*transformacion → Puntero al arreglo de pixeles transformados

unsigned char \*mascara → Puntero al arreglo de pixeles de la máscara

unsigned int semilla → Informacion de la semilla para saber donde aplicar el enmascaramiento

return → unsigned char\* enmascarado (Puntero al arreglo de pixeles transformados y enmascarados)

Enmascara el arreglo de pixeles transformados y retorna un arreglo con dicho enmascaramiento

***Verificación***

Argumentos:

unsigned char \*enmascarado → Puntero al arreglo de pixeles transformados y enmascarados

unsigned int \*maskingData → Puntero al arreglo con la información del archivo .txt

return → (true o false)

Verifica si el arreglo de pixeles transformados y enmascarados coincide con la información del archivo .txt