UNIVERSIDAD DE ANTIOQUIA
Facultad de Ingeniería

Informática II
Semestre 2025 – 1
Desafío 1
Abril 6, de 2025

Consideraciones Iniciales

Una imagen BMP (Bitmap) es un formato de archivo gráfico ampliamente utilizado para almacenar imágenes digitales, especialmente en plataformas Windows. Se caracteriza por su estructura simple y por almacenar los datos de imagen sin compresión, lo que permite un acceso directo y rápido a cada píxel. Cada píxel se representa mediante valores de color en forma de tripletas RGB (rojo, verde, azul), y dependiendo de la profundidad de color, puede usar 1, 4, 8, 16, 24 o 32 bits por píxel. En imágenes de 24 bits (Ver Figura 1), cada píxel ocupa exactamente 3 *bytes*, uno para cada canal de color.

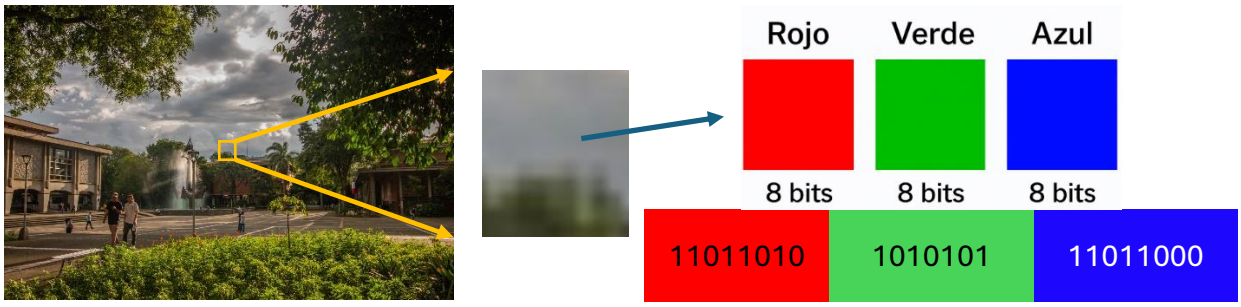

Figura 1. Representación binaria de un píxel específico de una imagen BMP de 24 bits

En el ámbito de la seguridad digital y el procesamiento de imágenes, es común emplear técnicas de transformación de datos para ocultar o proteger información visual. Estas técnicas permiten aplicar múltiples capas de cifrado y distorsión a una imagen, con el objetivo de volver irreconocible su contenido y dificultar su recuperación sin el conocimiento exacto del procedimiento utilizado.

En este escenario, una imagen BMP de 24 bits fue sometida a una serie de transformaciones a nivel de bits, que incluyen desplazamientos, rotaciones y operaciones XOR con una imagen de distorsión generada aleatoriamente. Estas transformaciones se aplicaron en un orden desconocido. Tras cada una de ellas, se empleó una técnica de enmascaramiento que consistió en sumar una porción de la imagen transformada con una máscara de color. La porción de la imagen utilizada para el enmascaramiento se determinó a partir de una posición aleatoria (semilla), desde la cual se aplicó la suma con la máscara.

Como resultado, se obtuvo una imagen final altamente transformada, junto con varios archivos de rastreo (uno por cada transformación realizada). Estos archivos contienen la información

resultante del enmascaramiento aplicado en cada etapa, y constituyen la única pista disponible para deducir el orden de las transformaciones y revertir sus efectos.

Este desafío simula un escenario de ingeniería inversa, en el que, con información limitada y sin conocer directamente el algoritmo completo ni el orden de los pasos aplicados, se debe reconstruir la imagen original utilizando lógica, deducción y habilidades de programación.

Objetivos

- Desarrollar la capacidad de solución de problemas en los estudiantes, enfrentándolos a problemáticas cercanas a situaciones reales.
- Evaluar si el estudiante ha adquirido las destrezas y conocimientos fundamentales de la programación en C++, tales como el uso de estructuras de control (secuenciales, iterativas y condicionales), manejo de tipos de datos, operaciones a nivel de bits, punteros, arreglos, memoria dinámica y funciones.

El objetivo principal de esta actividad es poner a prueba sus habilidades en el análisis de problemas y en el dominio del lenguaje C++. Si ha seguido un proceso disciplinado de aprendizaje a lo largo del semestre, esta es una excelente oportunidad para demostrarlo. Podrá proponer una solución efectiva y obtener un resultado satisfactorio. En caso contrario, esta actividad le permitirá identificar sus debilidades y tomar acciones correctivas con miras a prepararse mejor para enfrentar desafíos similares, como los planteados en la sección " Consideraciones Iniciales".

Se recomienda valorar adecuadamente la verdadera complejidad del problema planteado. No se rinda antes de intentarlo ni descarte posibles enfoques sin analizarlos. Descubrirá que, aunque al principio el reto pueda parecer difícil, ya ha desarrollado las herramientas necesarias para enfrentarlo con éxito. Si dedica el tiempo adecuado al análisis del problema, el proceso de codificación será mucho más fluido.

Esperamos que disfrute del desafío propuesto. Le sugerimos leer detenidamente todo el documento antes de comenzar, y asegurarse de entender completamente las instrucciones antes de abordar esta actividad evaluativa.

Fue revisado por los profesores Aníbal Guerra y Augusto Salazar.

Especificaciones

La empresa Informa2 necesita desarrollar un programa capaz de reconstruir una imagen original a partir de la siguiente información:

- Una imagen I_D de dimensiones m filas por n columnas, con tres canales (RGB), que representa el resultado final tras haber sido sometida a una serie de transformaciones a nivel de bits, aplicadas en un orden desconocido.

- Una imagen I_M, también de m filas por n columnas y tres canales (RGB), generada aleatoriamente, que pudo haber sido utilizada en una o varias ocasiones durante el proceso de transformación mediante operaciones XOR con versiones intermedias de I_D.
- Una máscara M, de dimensiones $i \leq m$ filas y $j \leq n$ columnas, con tres canales (RGB), utilizada para aplicar un enmascaramiento después de cada transformación a nivel de bits. Este enmascaramiento consiste en seleccionar aleatoriamente un píxel de desplazamiento s en la imagen transformada I_D, y a partir de este calcular las sumas:

  $$S(k) = ID(k + s) + M(k) \quad \text{para} \quad 0 \leq k < i \times j \times 3$$

- N archivos *.txt*, que contienen la información generada durante el enmascaramiento aplicado en cada etapa del proceso.

  Cada archivo incluye:

  - En la primera línea, un valor entero que representa el desplazamiento s utilizado para aplicar el enmascaramiento.
  - En las líneas siguientes, conjuntos de tres valores enteros que representan la suma de los canales RGB, píxel a píxel, entre una porción transformada de I_D y la máscara M. La Figura 2 uno muestra un ejemplo de las primeras siete filas de un archivo resultado que almacena el resultado del enmascaramiento.

```
100
303 303 423
423 423 275
275 275 295
295 295 408
153 153 145
145 145 245
```

Figura 2. Resultado del enmascaramiento usando una semilla de 100 (Primeras 7 filas)

Como insumo, se le proporciona:

1. Código fuente de un programa que abre una imagen BMP y permite acceder a los valores de los píxeles, de forma que pueda operar con dicha información usando arreglos dinámicos.
2. Código fuente para abrir los archivos que contienen el resultado del enmascaramiento, permitiéndole operar dichos valores con arreglos dinámicos.
3. Código fuente para exportar la información contenida en arreglos dinámicos como imágenes.
4. Dos conjuntos de archivos que representan dos posibles entradas y salidas del sistema.

A continuación, se muestran algunos detalles de conjunto de datos etiquetado como Caso 1. En este caso, se realizaron 3 transformaciones a una imagen en el siguiente orden:

- Paso 1: XOR entre la imagen original I_O y una I_M . Archivo de salida *P1.bmp*

- Paso 2: Rotación 3 bits a la derecha (I_{R3}) de la imagen resultante en el paso 1. Archivo de salida *P2.bmp*.
- Paso 3: XOR entre I_{R3} y una I_M . Archivo de salida *P3.bmp*.
- Luego de los pasos 1 y dos se aplica el enmascaramiento entre la imagen resultante la una máscara M. Esto produce dos archivos *M1.text* y *M2.txt*.

Luego del último paso, nunca se realiza enmascaramiento pues dicha información no sirve para el proceso de reconstrucción.

La Figura 3 muestra la salida en cada una de las transformaciones y la máscara M incluida en el Caso 1.

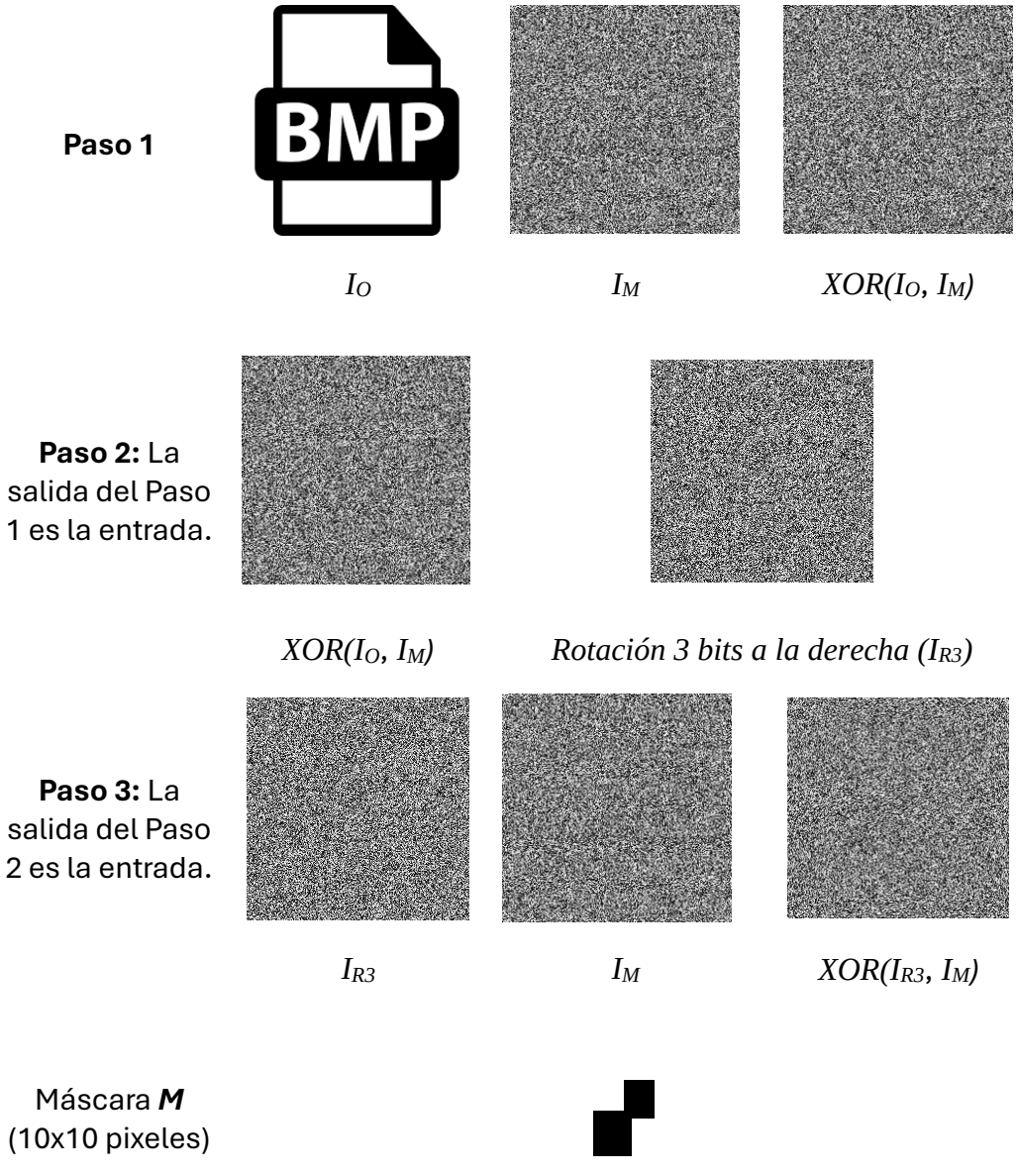

Figura 3. Resultado de 3 transformaciones sucesivas a nivel de bit

De acuerdo con lo anterior, entre otras cosas, usted deberá:

[15%] Escribir funciones para realizar operaciones a nivel de bit, tales como XOR, desplazamiento y rotación de bits. El máximo número de bits a rotar o desplazar es de 8.

[10%] Realizar experimentos con las diferentes operaciones a nivel de bit y analizar el efecto de las transformaciones sobre la integridad de los datos y su utilidad en un escenario de encriptación básica de información.

[10%] Implementar un algoritmo que permita verificar el resultado del enmascaramiento, comparando la imagen transformada y la máscara contra los archivos de resultados.

[65%] Implementar un algoritmo que permita identificar qué tipo de operaciones a nivel de bits fueron realizadas (y en qué orden), con el fin de reconstruir la imagen original.

Requisitos mínimos

A continuación, se describen los requisitos que se deben cumplir. El incumplimiento de cualquiera de ellos implica que su nota sea cero.

1. Genere un informe en donde se detalle el desarrollo del proyecto, explique entre otras cosas:
   a. Análisis del problema y consideraciones para la alternativa de solución propuesta.
   b. Esquema donde describa las tareas que usted definió en el desarrollo de los algoritmos.
   c. Algoritmos implementados.
   d. Problemas de desarrollo que afrontó.
   e. Evolución de la solución y consideraciones para tener en cuenta en la implementación.
2. La solución debe ser implementada en lenguaje C++ en el *framework* Qt.
3. La implementación debe incluir el uso de punteros, arreglos y memoria dinámica.
4. En la implementación no se pueden usar estructuras ni STL.
5. Se debe crear un repositorio público en el cual se van a poder cargar todos los archivos relacionados a la solución planteada por usted (informe, código fuente y otros anexos).
6. Una vez cumplida la fecha de entrega no se podrá hacer modificación alguna al repositorio.
7. Se deben hacer *commits* de forma regular (repartidos durante el periodo de tiempo propuesto para las entregas) de tal forma que se evidencie la evolución de la propuesta de solución y su implementación.
8. Se debe adjuntar un enlace de *youtube* a un video que debe incluir lo siguiente:
   a. Presentación de la solución planteada. Análisis realizado y explicación de la arquitectura del sistema (3 minutos máximo).
   b. Demostración de funcionamiento del sistema. Explicar cómo funciona: ejemplos demostrativos (3 minutos máximo).
   c. Explicación del código fuente. Tenga en cuenta que debe justificar la elección de las variables y estructuras de control usados. Por qué eligió uno u otro tipo de variable o estructura de control en cada caso particular y que ventaja ofrecen estos

en comparación de otras que también podrían haber sido usados (5 minutos máximo).
    d. La duración total del video no debe exceder 11 minutos ni ser inferior a 5 minutos.
    e. Asegúrese que el video tenga buen sonido y que se puede visualizar con resolución suficiente para apreciar bien los componentes presentados.
9. El plazo de entrega se divide en dos momentos:
    a. El día 12 de abril para adjuntar la evidencia del proceso de análisis y diseño de la solución.
    b. El día 25 de abril para adjuntar la evidencia del proceso de implementación.
10. En Ude@ se deben adjuntar **dos enlaces**: uno al repositorio y otro al video, nada más.
11. Para la evaluación del desafío se realizará una sustentación oral en un horario concertado con el profesor. La asistencia a la sustentación es obligatoria.