Informe proceso de análisis y diseño de la solución

Samuel Zuleta Zapata
Juan Fernando Henao Ledesma

Informática II

Universidad de Antioquia

2025

Tras haber analizado el problema planteado en el documento, nos damos cuenta que el primer paso a realizar siempre va a ser la aplicación de los diferentes procesos a nivel de bits que se mencionaron en clase (XOR, <<, >>, rotación a la derecha y rotación a la izquierda), como se debe verificar uno a uno hasta encontrar el procedimiento correcto hay que tener en cuenta que tanto el desplazamiento como la rotación pueden tener 8 opciones posibles, por tanto consideramos que lo mas eficiente seria primeramente revisar la opción del XOR, en el caso de que este proceso no sea el que se haya usado en la encriptación procedemos a verificar el resto uno a uno. Como idea inicial de solución para el desplazamiento y la rotación es implementar su verificación dentro de un ciclo, donde en cada iteración se revisará si es el proceso correcto, sino se procede a intentar con la próxima cantidad n de bits a rotar/desplazar; cabe aclarar que en el caso de la rotación, al no tener un operador definido por el lenguaje se tendría una función que realice esta operación.

Claramente luego de cada proceso se tiene que hacer el enmascaramiento para certificar si fue el procedimiento correcto comparando con los datos de los archivos.txt, la cuestion aqui es en el momento que se verifiquen cuáles fueron los procesos ¿donde almacenar los procesos en cuestión que fueron aplicados a la imagen para luego ser mostrados en pantalla?

Se plantean 2 posibles opciones de solución, siendo la primera un arreglo dinámico donde se almacenarán todos los procesos realizados y que se puede deducir el tamaño a reservar por medio de los nombres de los archivos.txt; la otra opción sería escribir en otro archivo.txt todos los procesos realizados. Por lo visto en el curso y su contenido hasta el momento consideramos que lo más apropiado es un arreglo dinámico, ya que podemos acceder con mayor facilidad a él durante la implementación y consideramos que va a ser más fácil de integrar a la lógica del programa.

Algo de lo que nos percatamos es que los procesos >> y << van a ocasionar pérdida de información, lo que va a hacer que no podamos reconstruir la información original con el 100% de precisión, todo esto no ocurre con las rotaciones ya que allí la información se reorganiza. Se sigue analizando qué repercusiones tiene la pérdida de cierta parte de la información para la desencriptación.

Se tiene claridad que el orden a seguir en nuestra estrategia va a ser:

1. Elegir y aplicar operación a nivel de bits.
2. Realizar la suma con la máscara.
3. Comparar con el archivo.txt para la verificación.

De acuerdo a los resultados del proceso se abren las dos siguientes posibilidades:

1. Si la verificación falla se repite todo el proceso.
2. Si la verificación es exitosa se guarda en el arreglo dinámico el proceso en cuestión y se avanza.

A raíz de esto nace otra pregunta, ¿cuando sabemos que hemos terminado? la respuesta a esto es simple y es mediante el número de archivos.txt, donde se nos dice implícitamente la cantidad de pasos realizados, siendo esta nuestra condición de parada.

IMPLEMENTACIÓN

Al iniciar el desarrollo del desafío y reanalizar el problema, optamos por una solución diferente a la planteada inicialmente, debido a un error en la interpretación. En un principio, utilizamos el resultado del enmascaramiento para construir la imagen encriptada, realizando una resta con la máscara para intentar recuperar los valores originales antes de aplicar la operación a nivel de bits. Sin embargo, tras implementar y probar, notamos que no funcionaba correctamente.

Posteriormente, entendimos que los valores de los archivos .txt solo se usaban para verificación, por lo que eliminamos las funciones relacionadas con la resta de la máscara y las verificaciones con el contenido del .txt. Luego de este ajuste, replanteamos el problema y volvimos al planteamiento inicial.

La principal modificación fue que no aplicamos la operación a toda la imagen de inmediato. Primero, verificamos únicamente la parte de la imagen que había sido sumada con la máscara. Si está coincidía con los datos del .txt, entonces sí aplicábamos la operación al resto de la imagen para proceder al siguiente paso.

En cuanto a la operación de rotación, analizamos que una rotación de 6 bits a la derecha equivale a una rotación de 2 bits a la izquierda, ya que la suma de ambas debe ser 8 bits. En la implementación, programamos el código para que solo verificará rotaciones hacia la derecha, pero mostrando también la correspondiente rotación hacia la izquierda.

Además, desarrollamos una función que cuenta automáticamente el número de archivos .txt, evitando que el usuario tenga que ingresar este dato manualmente. Esto

es posible debido a la notación conocida de los archivos (M0, M1, M2, etc.), permitiendo determinar de forma automática el número de pasos necesarios para terminar la ejecución del programa.