Algoritmos implementados:

**unsigned char\* loadPixels(QString input, int &width, int &height);**

Esta función abre un archivo de texto que contiene una semilla en la primera línea y, a continuación, una lista de valores RGB resultantes del proceso de enmascaramiento. Primero cuenta cuántos tripletes de píxeles hay, luego reserva memoria dinámica y finalmente carga los valores en un arreglo de enteros.

**bool exportImage(unsigned char\* pixelData, int width,int height, QString archivoSalida);**

Esta función crea una imagen de tipo QImage utilizando los datos contenidos en el arreglo dinámico pixelData, que debe representar una imagen en formato RGB888 (3 bytes por píxel, sin padding). A continuación, copia los datos línea por línea a la imagen de salida y guarda el archivo resultante en formato BMP en la ruta especificada.

**unsigned int\* loadSeedMasking(const char\* nombreArchivo, int &seed, int &n\_pixels);**

Esta función abre un archivo de texto que contiene una semilla en la primera línea y, a continuación, una lista de valores RGB resultantes del proceso de enmascaramiento. Primero cuenta cuántos tripletes de píxeles hay, luego reserva memoria dinámica y finalmente carga los valores en un arreglo de enteros.

**void unionOperacion(unsigned char\* pixelData, unsigned char\* arrMascara, unsigned int\* ArrTxt, unsigned char\* arrXor, int totalBytes, int tamMascara, int semilla);**

Esta función aplica las diferentes transformaciones a la imagen distorsionada hasta encontrar la correcta.

**bool verficarOperacionXor(unsigned char \*arrImagen,unsigned char \*arrMascara,unsigned int \*arrTxt,unsigned char \*arrXor, int tamMascara, int semilla )**

Esta es encargada de aplicar transformación aun bit, sumar con la máscara y verificar y es igual al txt (se le aplica transformación a la región donde se encuentra la máscara)

**bool xorOperacion(unsigned char\* pixelData, unsigned char \*arrMascara, unsigned int \*ArrTxt, unsigned char\* otherData, int totalBytes, int tamMascara, int semilla, int numOrden)**

Esta función es encargada de realizar la transformación Xor de manera permanente dependiendo del resultado de verficarOperacionXor

**Nota:** Este procedimiento es igual con las otras transformaciones con sus respectivas funciones, es decir, existen dos funciones para cada transformación verificarOperacionRotacion, rotacionOperacion y verficarOperacionDesplazamiento, despOperacion que nos permitirán realizar la operación de manera permanente dependiendo del resultado de su propia verificación.

**void procesarArchivos(int numArchivos,unsigned char \*pixelData, unsigned char \*arrMascara, unsigned char \*arrXor, int totalBytes);**

Esta función genera cadena de caracteres(Mn.txt) , usa la función loadSeedMasking para acceder al contenido de cada txt y usa la función unionOperacion para aplicarle las transformaciones a la imagen modificada aplicando enmascaramiento y comparando con el Mntxt respectivamente.

Problemas de desarrollo que afrontó:

1. Se nos presentó varios inconvenientes al usar Git al momento de hacer commits, por lo tanto, esto no nos facilitó el uso del sistema de control de manera adecuada.
2. Se nos dificulto comprender de manera adecuada los requisitos del problema
3. Se nos complicó la realización de la función desplazamiento

Evolución de la solución y consideraciones para tener en cuenta en la implementación.

Determinamos estas tres funcionalidades principales para llevar a cabo la realización del problema:

1. Operación bit a bit (Transformaciones)
2. Enmascaramiento
3. Verificación

Decidimos elaborar en primer lugar realizar las funciones anteriores.

Operaciones bit a bit:

1. Operación Xor
2. Operación desplazamiento
3. Operación Rotación

Enmascaramiento:

1. Esta función se encargar de aplicar el enmascaramiento después de cada transformación

Verificación:

1. Encargada de verificar que los resultados del enmascaramiento coincidan con los archivos txt

Las funciones que nos proporcionaron los docentes nos facilito el manejo de bit de las imágenes por medio de arreglos así que pudimos verificar la correctitud de las funciones realizadas anteriormente para evitar errores más adelante.

Evolución:

Nuestra primera intención para realizar las transformaciones a todo el arreglo.

Consideración1 :

* Hacer transformación en toda la imagen(arreglo)
* Hacer enmascaramiento
* Verificación

¿Era necesario hacer transformación en toda la imagen?, nos dimos cuenta lo que indica si una transformación es correcta es el contenido de los txt (enmascaramiento==txt), el enmascaramiento se hace con una imagen más pequeña o en el mejor de los casos igual así que lo hicimos fue aplicar transformación a una región de la imagen hacer enmascaramiento y verificamos. Esto nos permitió evitar operación innecesaria y evitar copias del arreglo.