```
    1: #include "FibLFSR.hpp"
    2:
    3: #include <SFML/System.hpp>
    4: #include <SFML/Window.hpp>
    5: #include <SFML/Graphics.hpp>
    6:
    7: // Transforms image using FibLFSR
    8: void transform(sf::Image&, FibLFSR*);
    9: // Display an encrypted copy of the picture, using the LFSR to do the enc
ryption
   10:
   11:
   12: int main(int argc, char* argv[]) {
   13:
   14:         // Command line arguments
   15:         const string inputFileName = argv[1];
   16:         const string outputFileName = argv[2];
   17:         const string binaryString = argv[3];
   18:
   19:         // Create a base image and a to-be-encrypted image
   20:         sf::Image imageBase, imageEncrypt;
   21:         if (!imageBase.loadFromFile(inputFileName)) return -1;
   22:         if (!imageEncrypt.loadFromFile(inputFileName)) return -1;
   23:
   24:         // Create sprite for base image
   25:         sf::Vector2u size = imageBase.getSize();
   26:         sf::RenderWindow windowBase(sf::VideoMode(size.x, size.y), "Base
Image");
   27:         sf::Texture textureBase;
   28:         textureBase.loadFromImage(imageBase);
   29:         sf::Sprite spriteBase;
   30:         spriteBase.setTexture(textureBase);
   31:
   32:         // Perform the transformation using the given binary string
   33:         FibLFSR L1(binaryString);
   34:         transform(imageEncrypt, &L1);
   35:
   36:         // Create sprite for encrypted image
   37:         sf::RenderWindow windowEncrypt(sf::VideoMode(size.x, size.y), "En
crypted Image");
   38:         sf::Texture textureEncrypt;
   39:         textureEncrypt.loadFromImage(imageEncrypt);
   40:         sf::Sprite spriteEncrypt;
   41:         spriteEncrypt.setTexture(textureEncrypt);
   42:
   43:         // While both windows are open
   44:         while (windowBase.isOpen() && windowEncrypt.isOpen()){
   45:
   46:                 sf::Event event;
   47:                 // If either recieve a call to close, close both
   48:                 while (windowBase.pollEvent(event) || windowEncrypt.pollE
vent(event)){
   49:                         if (event.type == sf::Event::Closed){
   50:                                 windowBase.close(); windowEncrypt.close()
;
   51:                         }
   52:                 }
   53:                 // Display base image
   54:                 windowBase.clear(sf::Color::White);
   55:                 windowBase.draw(spriteBase);
   56:                 windowBase.display();
   57:                 // Display encrypted image
   58:                 windowEncrypt.clear(sf::Color::White);
   59:                 windowEncrypt.draw(spriteEncrypt);
   60:                 windowEncrypt.display();
```

```
    61:         }
    62:         // Save to file, else return failure/error
    63:         if (!imageEncrypt.saveToFile(outputFileName)) return -1;
    64:         return 0;
    65: }
    66:
    67: void transform(sf::Image& img, FibLFSR* lfsr){
    68:
    69:         sf::Color p;
    70:         sf::Vector2u size = img.getSize();
    71:         // For every x and y in img, xor each rgb val with a newly genera
ted 8 bit num
    72:         for (int x = 0; x < (int)size.x; x++) {
    73:                 for (int y = 0; y < (int)size.y; y++) {
    74:                         p = img.getPixel(x, y);
    75:                         p.r ^= lfsr->generate(8);
    76:                         p.g ^= lfsr->generate(8);
    77:                         p.b ^= lfsr->generate(8);
    78:                         img.setPixel(x, y, p);
    79:                 }
    80:         }
    81: }
```