I used ChatGPT to add comments to the script and to debug the image encryption and decryption option.

static void EncryptDecryptImageOption(string imageFile, string seed, string tapStr, bool isEncryption)

    {

        // Attempt to parse the tap position from string to integer

        if (!int.TryParse(tapStr, out int tap))

        {

            Console.WriteLine("Invalid tap position. It must be an integer.");

            return;

        }

        // Check if the image file exists

        if (!File.Exists(imageFile))

        {

            Console.WriteLine("Image file not found.");

            return;

        }

        // Verify that the provided file is a valid image

        if (!IsImageFile(imageFile))

        {

            Console.WriteLine("The file provided is not a valid image.");

            return;

        }

        // Open and decode the image using SkiaSharp

        using var input = File.OpenRead(imageFile);

        using var bitmap = SKBitmap.Decode(input);

        // Check if the image was successfully decoded

        if (bitmap == null)

        {

            Console.WriteLine("Failed to decode the image. The file may not be a valid image.");

            return;

        }

        // Initialize the LFSR with the provided seed and tap position

        LFSR lfsr = new LFSR(seed, tap);

        // Iterate over each pixel in the image to encrypt/decrypt it

        for (int y = 0; y < bitmap.Height; y++)

        {

            for (int x = 0; x < bitmap.Width; x++)

            {

                SKColor color = bitmap.GetPixel(x, y);

                // XOR each color channel with a randomly generated byte from the LFSR

                byte red = (byte)(color.Red ^ GenerateRandomByte(lfsr));

                byte green = (byte)(color.Green ^ GenerateRandomByte(lfsr));

                byte blue = (byte)(color.Blue ^ GenerateRandomByte(lfsr));

                // Create a new color with the encrypted/decrypted channels and the original alpha

                SKColor newColor = new SKColor(red, green, blue, color.Alpha);

                bitmap.SetPixel(x, y, newColor);

            }

        }

        // Define the output file name based on the operation

        string outputFileName = isEncryption

            ? $"{Path.GetFileNameWithoutExtension(imageFile)}ENCRYPTED{Path.GetExtension(imageFile)}"

            : GenerateDecryptedFileName(imageFile);

        using var output = File.OpenWrite(outputFileName);

        // Encode and save the encrypted/decrypted image in PNG format

        bitmap.Encode(output, SKEncodedImageFormat.Png, 100);

        string operation = isEncryption ? "Encrypted" : "Decrypted";

        Console.WriteLine($"{operation} image saved as {outputFileName}");

    }

    // Generates the decrypted image file name based on the encrypted file name

    static string GenerateDecryptedFileName(string imageFile)

    {

        string baseFileName = Path.GetFileNameWithoutExtension(imageFile);

        if (baseFileName.EndsWith("ENCRYPTED", StringComparison.OrdinalIgnoreCase))

        {

            baseFileName = baseFileName.Substring(0, baseFileName.Length - "ENCRYPTED".Length);

        }

        return $"{baseFileName}DECRYPTED{Path.GetExtension(imageFile)}";

    }