FEC-CODE Explanation

Benjamin Rappoport, Howard Levinson

July 2025

1 Overview

FEC-Codes (Fourier Encrypted Color Codes) are a new type of computer readable matrix that we developed in the Oberlin Summer Research Institute. These new codes are encrypted through random sampling (shuffling of the QR code) and multiple Fourier transformations.

2 Process

Below is a general description of how to construct an FEC-Code, there are different methods that could be used for some steps (such as the shuffling or color pairing). To see our specific implementation of these processes, see the GitHub repository.

1. As an input you need some binary matrix, it could be a full QR code, data matrix, or direct data (though having error correction is very helpful as the encryption does lose a small amount of data).

2. Take a forwards Fourier transformation of the matrix, then shuffle this matrix (the shuffle is the primary encryption process of the FEC-Code) it can be done many ways, but the value pairs need to be kept, meaning if the value stored in (3,3) is put in (1,1) then the value stored in (-3,-3) is put in (-1,-1).

3. Now take a second Fourier transformation of the previous matrix, this will output a matrix of real values. To make it displayable round each value to one of 8 values. The way we did this step was finding the maximum and minimum values and then creating 6 values evenly between these numbers, and making each number whichever of the 8 values it was nearest to.

4. To turn this matrix into FEC-Code, we make the values in lowest- to highest-order black, blue, green, cyan, red, magenta, yellow, and white. Finally, create the border, the left and bottom sides should be all black, for scanning purposes. The top from position 1 to (length -

1) should be the binary number in floating point form of the minimum value. The right from position 1 to (length - 1) should be the binary number in floating point form of the maximum value. Both of these values may have to be rounded depending on the size of the code, but for matrices of size 29x29, this has a negligible effect on errors. Here is an example of an FEC-Code:

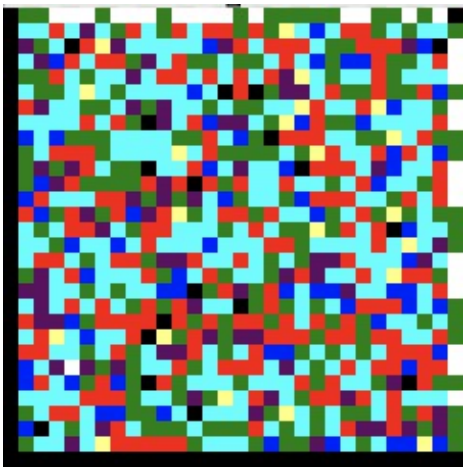

Figure 1: FEC-Code Example

Now to get a QR code (or any binary matrix from a FEC-Code) you will need an FEC-Code and a key (the specific type of key depend on how your shuffler is designed).

1. Using the binary numbers stored in the top and right side of the code find the maximum and minimum values, then calculate the 6 numbers between those two points.
2. Now construct a matrix with all of the previously calculated values.
3. Take this matrix and run an inverse Fourier Transformation on it.
4. Now using the key, which in our implementation is a dictionary of correct values, but could be any way to reverse the shuffler, to un-shuffle the third transformation.
5. Finally take another inverse Fourier Transformation, then round all values to either 0 or 1. This matrix is now a readable QR Code.

Below is an example of the full process of encryption. The first image is of the original QR code, the second of the shuffled transformation, the third of the second transformation (which is all real numbers), the fourth of the inverse of the real matrix, and the fifth of the unshuffled inverse, that returns the almost correct original image.

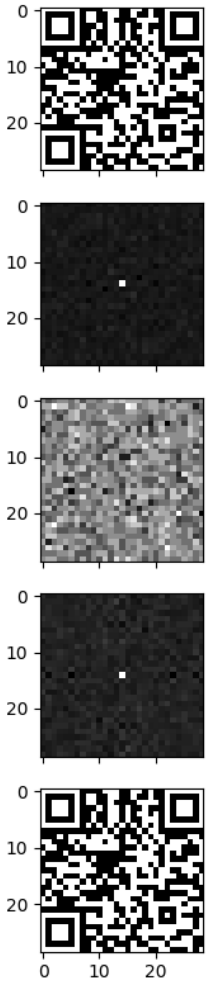

Figure 2: Full FEC-Code Process

3 Conclusion

This document is meant to be a general framework of how an FEC-Code could be created. This is because our research shows that the design is possible, but is not attempting to be a production ready model. To see the specific ways we implemented different elements look at the code file this PDF is attached too. This framework has shown new capabilities of Fourier transformation and could be translated into a new form of readable encrypted data.