Quantum Image Encryption

Deval Deliwala & Savar Sinha

June 19, 2024

The goal of this project is to devise a quantum-based image encryption algorithm that robustly confuses and diffuses information stored in an image. The procedure is roughly divided into the following steps:

1 Encryption

1.1 Pixels to Statevectors

We take a square $n \times n$ image, for example:

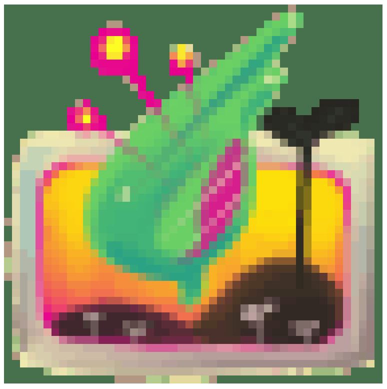

Figure 1: Starting Image

We transform every pixel on the image into a point on a unit sphere, ensuring they are *evenly distributed*. Something like the Fibonacci Sphere Algorithm, which is reversible to allow for decryption. This transforms the above image into the following:

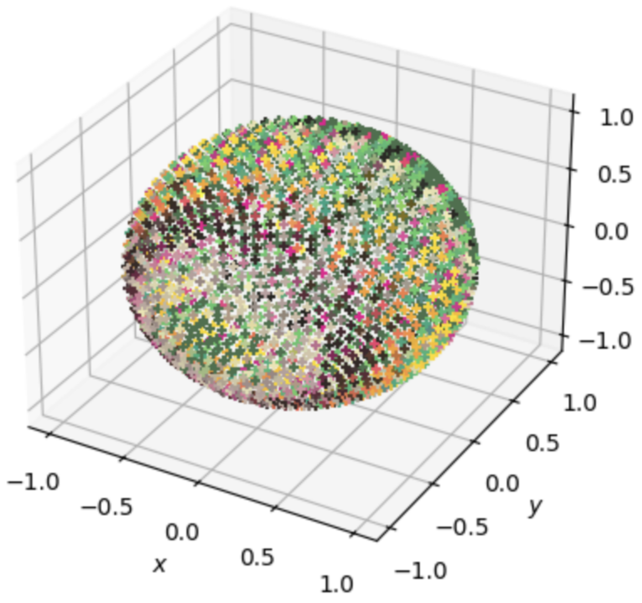

Figure 2: Pixels to Unit Sphere Points

If we initialize a vector $\vec{v}$ stemming from the origin to each "pixel-point", the norm $|\vec{v}| = 1$, and so represents a valid state vector on the Bloch Sphere. In this way we convert each pixel of an image into a valid statevector.

Now we somehow have to convert the sphere, using all of the pixel/state vector colors into a *gradient* consisting of the color space of the image. It would look something like this:

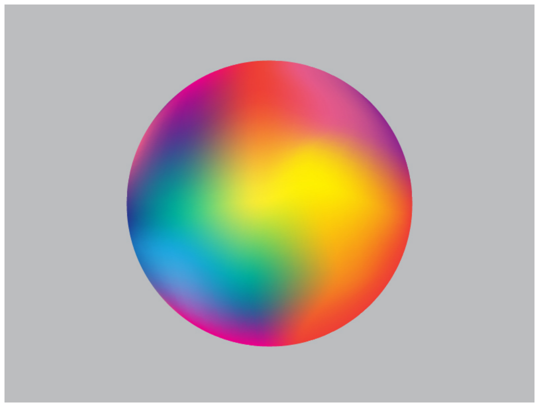

Figure 3: Bloch Sphere contains color-space of image

except it would consist of the colors in the image.

We could do this by diffusing the colors of each pixel radially until its half of the distance from the nearest pixel...If that makes sense. I'm not sure exactly, but it is possible. The only main point is that the exact position each pixel statevector points to is the color the pixel is in the original image. If we decide to block groups of vectors we'd have to figure out someway to do this as well.

The point is that the sphere now contains the rough RGB information of the digital image, and we also have a state vector for each pixel of an image (i.e. the state vector points to the exact position on the spherical gradient which corresponds to its color).

The spherical gradient will act as our **public key** (we'll have to figure out someway to communicate it, more on this later). It stores the RGB information of the digital image, but it is infeasible to try and recreate the image or create some array of arbitrary pixels that wouldn't yield rubbish.

1.2 Scrambling

Here is where we apply a sequence of randomized gates of large depth to all the pixel state vectors. Depending on the number of pixels on an image, we could apply a circuit on 5 qubits at a time, or 10, it doesn't really matter we will figure out which is optimal for cost and efficiency during analysis.

There's a current function that already can do this, provided a seed. It just uses numpy's pseudo-random number generator.
We could make our own it's not that complex but it would generate something like this:

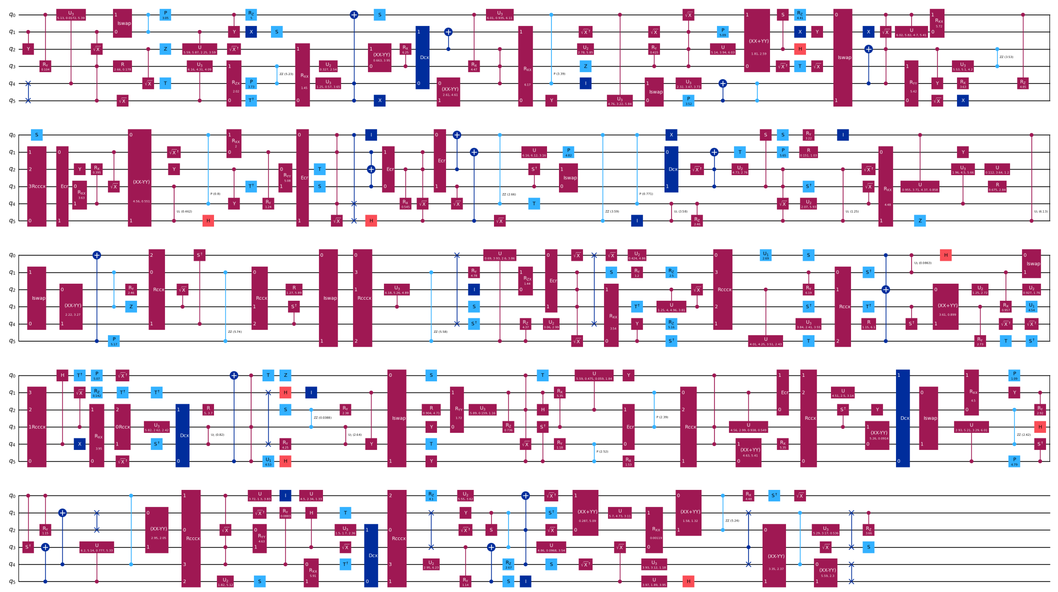

Figure 4: Arbitrary circuit on 5 qubits

We would have to ensure it consists only reversible gates, so we would likely have to make our own.

We would apply different circuits on (for example, 5) qubits at a time, following the ordering of the fibonacci sphere algorithm. We iterate through all pixel state vectors.

The seeds that are generated, along with the original and final positions of each qubit that we scrambled, will act as the **private key**. *This step will cause collisions.*

1.3 Variational Quantum Eigen-Spreading

Since scrambling will overlap pixels, we have to figure out someway to *reversibly* even out every pixel.

- We will call the orientation where all pixels are evenly spread the *ground state* of lowest energy.
- We define the energy of a pixel-statevector as the radial distance between it and the **nearest adjacent pixel-statevector**.
    - The total energy of the system is the sum of of the energy of the individual pixel-statevectors.

If we maximize the total energy of the system – that means every pixel is evenly distributed...I think. It makes sense in my head, but you get the idea.

Therefore we implement a quantum annealer like the Variational Quantum Eigensolver to start with an ansatz and continuously evolve the system until the energy is minimized and ground state is reached.

Now each pixel-statevector that originally pointed to its actual color before scrambling – points to an entirely new color on the bloch sphere, and we can undo the fibonacci sphere algorithm in the same way to yield a new image, with an entirely new orientation colors. The important thing is that the color information *is still there.*

1.4 Arnold's Cat Map & Peripheral-pixel blurring

After we have a new encrypted image, we can implement a chaotic mapping like Arnold's Cat Map (ACM) to further confuse the image. Afterwards we can implement some peripheral-pixel blurring or some other method of diffusing the image uniformly so it can past the RGB histogram statistical text.

Now, we should have an entirely new image that is much *much* different than the original image while still retaining all the information.

Encryption is complete.

2 Decryption

2.1 Undoing Arnold's Cat Map & Pixel Diffusion

This is fairly straightforward, both ACM and Pixel Diffusion are reversible.

2.2 Undoing VQE spreading

This involves undoing all the circuits that were used by VQE, knowing which qubit was placed in which circuit in which location, until all the qubits are back to their colliding places post-scrambling.

2.3 Undoing the scrambling

Using the private key, which stores all the original and final qubit positions before and after scrambling, in addition to all the seeds that were used for scrambling, we can just apply qc.inverse() to convert all the pixels back to their original locations, pointing to the correct color in the RGB gradient sphere.

2.4 Undo Fibonacci Sphere Algorithm

Straightforward, now that all pixels point to the correct color, we just take all the colors and undo the sphere algorithm to get the original image.

Decryption is complete.

3 Advantages & Disadvantages

3.1 Advantages

from the top of my head

- All the color information is stored in the public key, which reduces the amount of separate qubits we need by *half* compared to FIQR or NEQR. In addition, knowing the color intensity using a separate *gray* qubit isn't necessary either.
- By employing scrambling, VQE spreading, ACM, and a pixel diffuser, this algorithm is a solid level of encryption.
- The private key, which stores the intitial and final pixel state vector positions post-scrambling, in addition to the seeds from which we generated the circuits to *do* the scrambling, is very long.
- Allows Bob, (the receiver) to determine how much quality his decrypted image is (this is huge). By communicating with Alice to vary how much the RGB spherical gradient (the public key) contains the original images color information, (how much the colors were diffused), he can himself determine how much information he wants his decrypted image to have. This will take some working on, since even if the RGB sphere contains less information, the number

of pixels is the same, so we could develop some way to somehow accurately constrain the amount of pixels by seeing if their colors are similar delete on pixel statevec and just use the same one twice. (confusing, hope it made sense, see the image below).

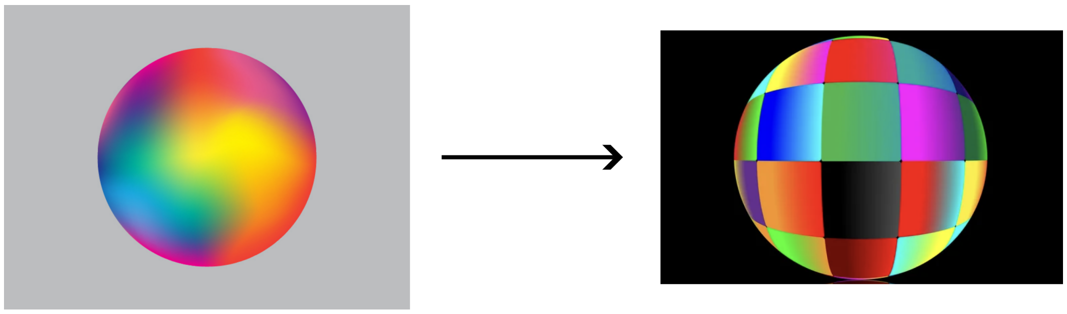

Figure 5: Reducing the quality of the RGB sphere

If we reduce the amount each pixel is diffused by discretizing the RGB sphere, the decrypted image will then have many pixels that point to the same rough color. If we get rid of these excess pixels, the image now has less image quality. In this manner Alice can choose to optimize this algorithm depending on how much information out of the original image she actually needs. (Or imagine if we figured out a way to do this to optimize it every time based on the image – but that may be too difficult). This is just an idea, might be hard to implement practically. But if it could work it would mean the sender could choose *how much* of the image he actually wants to transmit.

3.2 Disadvantages

- Still requires a qubit for every pixel, unless we figure out how to block pixels together while still retaining the information of the image
- Working with non-square images might pose some issues. The Fibonacci sphere algorithm does work for non square images so I think this shouldn't be an issue (which if true, would be another advantage).
- That's it. There's probably much more but I can not think of anything right now.

If there are any glaring issues please let me know.

Regards, Deval Manish Sigma Deliwala.