Exploring Quantum Encryption

Bacic, Justin and Bowen, Aranalyn

April 5, 2025

1 Introduction

1.1 Context

When transferring data in networks, a high level of security needs to be maintained as communication networks are used to relay very sensitive information. For classical computing and communications, encryption is one such technique that is used to secure information. The goal of encryption is to scramble data so that only those who are authorized can understand the original information. There are two types of encryption, symmetrical encryption and asymmetric encryption. Symmetrical encryption, which is the type that we will be looking at in this report, is where the sender and receiver share information and use that shared information to scramble and unscramble the data.

1.2 Problem Statement

There are many encryption schemes for classical communications but with the emerging field of quantum computing we want to carry this idea over to quantum communications to make secure quantum networks. The goal of this project is to implement and examine the encryption scheme proposed in a paper by Barbeau that combines the Quantum Permutation Pads and Clifford Operators [1]. More specifically we want to implement a simulation where the sender and receiver are using the scheme to securely transfer data to see the system in practice. We also want to evaluate in practice how secure the encryption scheme is using the metric of collisions which will be explained in detail later. We hope to see which parameters in the encryption scheme allows us to maximize the security.

1.3 Result

We have been able to successfully implement a sender and receiver simulation, as well as a testing environment where we were able to collect data so that we can evaluate the security of this encryption scheme. For the implementation we used MatLab where we could work with Quantum Circuits and Quantum states to simulate actual quantum data that we could send from a sender class to a receiver class. For the testing we ran many iterations of our sender and receiver simulation to gather collision data, and we ran this testing setup for varying sizes of qubit registers to see the effects on security.

1.4 Outline

In the rest of this report we will be going over the following sections. Section 2 will go over the background information needed to understand the technical details of the project. Section 3 will go over our simulation of the encryption scheme. Section 4 will outline our evaluation of the encryption scheme implementation and section 5 will summarize and conclude our work.

2 Background Information

As mentioned earlier, the encryption scheme that we are implementing uses Permutations and Clifford Operators in order to encrypt and decrypt data, in this section we will go over what permutations and clifford operators are and how they are used for the purposes of encrypting quantum data.

The X, Y, Z and I gates also known as the Pauli operator, and are integral in the derivation of clifford operators. For an n-qubit register the set of Pauli Operators is defined as the tensor product of n Pauli Operators. For this encryption scheme we exclude the tensor of n identity gates when considering the set of Pauli Operators on n qubits. A Clifford operator is a unitary matrix C that maps from the set Pauli operators to itself through conjugation. In other words, the conjugate of a Pauli operator P by C is equal to a Pauli Operator. U(1) represents the set of unitary scalar matrices, and the set of clifford operators for an n-qubit register are all $2^n \times 2^n$ unitary matrices that satisfy the previous property modulo U(1) [3]. What this means is that if the result of two Clifford gates when divided by the set U(1) is the same then they are considered equivalent. This results in the size of the set of clifford operators being finite. The exact number of Clifford Operators for n qubits, using the modified set of Pauli Operators that excludes the identity, is given by $2^{n^2+2n}\prod_{i=1}^{n}(4i-1)$ we will refer to this set of clifford operators as $C(n)$ [1].

Permutations on a set are an ordering of the elements of the set such that every element appears exactly once and for a set of n elements there are n! unique permutations of those elements [2]. Each of these permutations can be represented by a matrix P and can be multiplied by a vector v such that the resulting vector vP switches the rows of v according to the permutation order. Additionally, every permutation matrix P is invertible and the inverse P^{-1} reverses the permutation P, in other words if we have the vector vP and multiply this by P^{-1} the resulting vector is v. For quantum states of an n-qubit register, they can be represented by a vector of the probability amplitudes of each state and this vector has 2^n entries. This means that if we wanted to apply permutations to this quantum state vector there are $2^n!$ possible permutations, we can call this set of permutations for n qubits $P(n)$.

Now that we have established those concepts we can see how they are applied in this cryptographic scheme as outlined by Barbeau M. [1]. Each quantum message M that we are sending can be broken up into k blocks of n qubits such that $M = m_0 \cdots m_{k-1}$ and each m_i has a size of n qubits. Each of the blocks has a signature field with length $l > 0$ that is added to it. The sender and receiver both share the following information: the block size, the number of blocks in a message, the length of the signature field, a set C' of d_1 randomly selected clifford operators for $n + l$ qubits, and a set P' of d_2 randomly selected permutations of $n + l$ qubits. They also have random number generators that are seeded the same which gives them an arbitrarily long sequence of random numbers which they share.

The sender uses the random number generator to select k clifford operators from the set C' which will be used to sign each of the k blocks of the message. After the signing is applied the sender then proceeds to select randomly k permutations from the set P' and applies these permutations to each of the message blocks. For each message block m_i of length $n + l$ this results in a block of ciphertext c_i with the same length where c_i is equal to m_i after the ith selected Clifford operator and Permutation have been applied. The sender also sends the number of operators that have been generated thus far in the communication session.

On the receiver end, the receiver checks the received number of operators generated thus far to ensure that it is new, this provides replay protection. Since the number of blocks per message is shared the receiver uses their random number generation to also select k Clifford Operators which will correspond to the operators used to encrypt the original message. Similarly the receiver will also select k permutations. Then for each Clifford operator the receiver calculates the conjugate transpose and for each permutation the receiver calculates the inverse. At this point the receiver has all the tools they need to decrypt the ciphertext, they first apply the inverse permutation, then they apply the conjugate transpose Clifford operator. For each block the receiver checks the last l qubits to ensure that the signature is correct and if so the message is accepted otherwise the message is rejected. The message can be obtained by putting together the first n qubits of each of the blocks.

With that we have a detailed outline from which we can move to the implementation and testing of this encryption scheme.

3 Results

We were able to successfully implement a modified version of the encryption scheme in Matlab using the Matlab Support Package for Quantum Computing. The structure of our implementation can be seen in Figure 1.

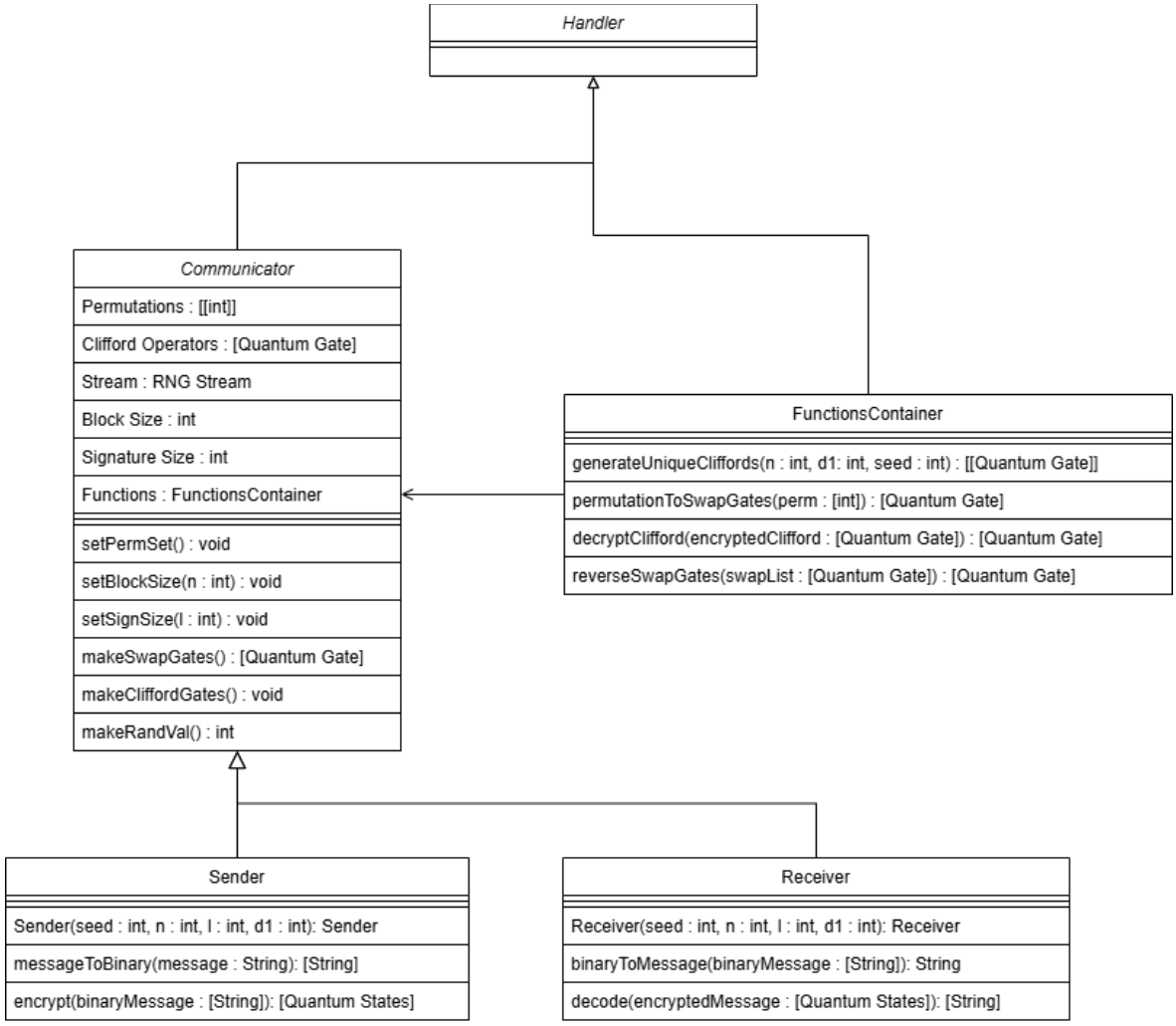

Figure 1: UML diagram of the Matlab implementation

In this UML diagram we can see four main classes, the sender class which sends messages, the receiver class that receives the messages that the sender sends, the communicator class which is inherited by the sender and receiver, and finally the functions container class that holds many utility functions. The two main classes that were acting in our implementation, the sender class and the receiver class, stored the following data: the set of randomly chosen permutations, the set of chosen clifford operators, the block size, the size of the signature, and a random number generation stream. In the original scheme the sender and receiver have two arbitrarily long sequences of random numbers that they share but for this implementation it is

simpler to have each of the parties generate these separately. During initialization the sender and receiver are given a seed so that their random number generation is the same, as well as the block size and the signature size. Since we just need one number for the seed we can establish this through QKD which adds to the security of the scheme.

In the sender class for the messageToBinary function the sender takes in a plaintext string, converts it to binary and breaks it up into chunks that have size equal to the block size. The output of this function is passed into the encrypt function which first adds signature bits to the end of each of the binary strings, which for our implementation is all 1's. Then it iterates through each of the blocks, generates a quantum state with a combination of ket 1 and ket 0 so that it is equivalent to the binary string. Then it takes a randomly chosen Clifford Operator and permutation and creates a quantum circuit combining the two. Then it runs the quantum plaintext through the circuit and stores the resulting quantum ciphertext in a list. It is worth noting at this point that this method of storage is not feasible with the current quantum technology that we have available and can only be achieved in the simulation environment. After all of the blocks have been encrypted the function returns the list of quantum ciphertext and the sender's work is done.

On the receiver's side, the quantum ciphertext outputted by the sender is given as the input to the decode function. For each block of quantum ciphertext the receiver selects the same clifford operator and permutation as the sender did when encoding it. To decode the block the receiver applies the inverse of the permutation first then the conjugate transpose of the clifford operator. This will give a quantum state that has all 0 probability amplitudes except for one state which will have a probability amplitude of 1. The receiver at this point can perform quantum measurement and record the measured classical bits. The receiver will then check that the signature is correct and if so they accept the rest of the binary string and store it in an array. After all of the blocks have been decoded the function returns the list of binary strings for each of the blocks. This list is then passed as the input to the binaryToMessage function which concatenates all of the binary strings and converts the binary string back into a string of characters and finally returns the string. If the encryption and decryption works as intended the plaintext that is given as the output will be identical to the input at the sender's side.

We did encounter one issue with the package that we used in Matlab that limited what we could do in terms of implementing the scheme. In the original scheme the permutations are applied to the probability amplitudes of the entire quantum state, which means for a n qubit register there would be $2^{n+l}!$ possible permutations. With the package that we used we were not able to permute the amplitudes and could instead only permute the qubits in the register, which meant that for the same n qubit register we would only have $(n + l)!$ permutations which is a significantly lower amount. This changes how secure our implementation is as it changes the rate of collisions which we will talk about in the next section.

4 Evaluation

We focused on assessing the performance and security of our implemented encryption scheme by measuring collision occurrences. A collision in encryption is defined as when two or more plaintexts produce the same ciphertext once encrypted. To do so, we conducted simulation tests where the encryption algorithm is tested across a range of block sizes (n) and the number of messages transmitted (i).

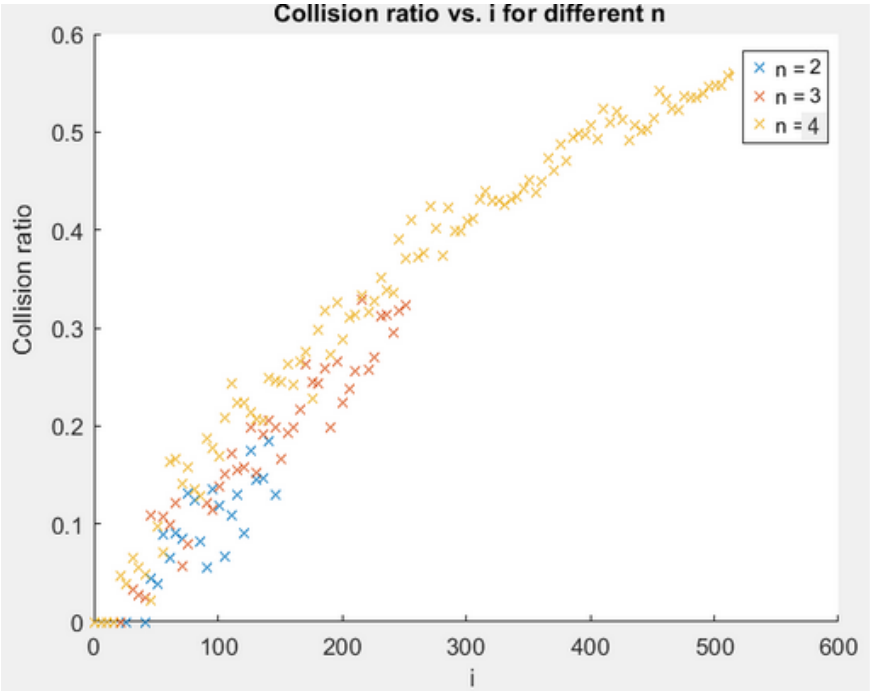

Figure 2: Results of collision simulation

Our simulation revealed that the collision ratio increases as a function of both the block size and the cumulative number of messages sent. Specifically, when a greater number of messages are processed, the probability of collision events naturally escalates. With each successive message, the likelihood of a permutation and Clifford operator combination producing an already used ciphertext grows, indicating the usefulness of the encryption algorithm diminishing under heavier data loads.

The collision modeling observed in our tests underscores the notable limitation of our current implementation. The effective permutation space reduction to (n+l)! compared to that in the original scheme directly affects the collision behavior, making it less robust.

It should be noted that in genuine applications, the full range of permutations and Clifford operators would be available, making the risk of collision far lower than what is shown in Figure 2.

5 Conclusion

5.1 Summary

With our project we were able to successfully implement in Matlab a simplified version of the Quantum Permutation Pad encryption scheme proposed by Barbeau M [1]. Our implementation shows how a sender and receiver would use this encryption scheme to send real messages to each other over a quantum channel. Additionally, with our implementation we were able to run testing to evaluate the performance of our version of the encryption scheme in practice.

5.2 Relevance

In terms of the content covered in the course, this encryption scheme ties in very closely with what we have seen. As the course is focused on quantum communications, this implementation provides a way that we can make quantum communications more secure using encryption. This project also requires many of the fundamental concepts of quantum computing such as quantum circuits, quantum gates, and quantum states.

5.3 Future Work

To take our implementation to the next level, we would need to find a way to correctly implement the permutation operations so that we can provide the same security as the original encryption scheme we based our work on. Additionally, incorporating the establishment of the random number generator seed through Quantum Key Distribution would be able to showcase more holistically how secure quantum communications can be done. For the entire encryption scheme to take it further we could consider adding error detection and correction for single bit errors. Adding this to the scheme would make it more feasible to implement since real world systems at the moment are not stable and can introduce many errors when sending quantum data.

Contributions of Team Members

Bacic, Justin:

- Report: Sections 1, 2, 3, 5
- Project presentation slideshow
- Matlab Demo file
- Implementation: Sender, Receiver, FunctionsContainer

Bowen, Aranalyn:

- Report: Section 4
- Evaluation section of the project presentation slideshow
- Setting up the testing and performing the testing of the encryption scheme performance
- Implementation: Sender, Receiver, FunctionsContainer

References

[1] Barbeau M. Quantum data communication protection with the quantum permutation pad block cipher in counter mode and Clifford operators [version 1; peer review: 1 approved]. ***F1000Research*** **2023, 12:1123 (https://doi.org/10.12688/f1000research.140027.1) (we'll cite these properly later)**

[2] "Permutation." Wikipedia, https://en.wikipedia.org/wiki/Permutation. Accessed April 2025.

[3] "Clifford (latest version)." ***IBM Quantum Documentation*****, IBM, https://docs.quantum.ibm.com/api/qiskit/qiskit.quantum_info. Accessed April 2025.**