An Exploration of Secure Circular Multi-Party Quantum Computation

by

Margie Christ

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Science
in
Physics (Quantum Information)

Waterloo, Ontario, Canada, 2022

Author's Declaration

Statement of Contributions

Margie Christ is the primary author of all material in this thesis. Work in chapter 3 was done in collaboration with Hussein Abulkasim.

Abstract

Multi party quantum computation (MPQC) is any quantum cryptographic protocol where multiple untrusted users collaborate to perform calculations on their combined data without revealing their private information. MPQC is not guaranteed information-theoretically secure by the laws of quantum mechanics, and thus finding schemes for MPQC that ensure a high degree of security is an ongoing research task. In this thesis, we examine an approach for MPQC protocols that employs a circular structure. The circular structure minimizes the amount of information transmitted from user to user, increasing efficiency and security. This makes it a good structure for MPQC.

We address three main topics related to circular MPQC. First, we build a quantum circuit to practically implement an existing circular MPQC protocol. We demonstrate feasibility and reasonable efficiency in the circuit model. Second, we examine the security of circular MPQC. We consider the protocol's vulnerabilities to outside and inside attacks. After identifying weaknesses in the scheme, we suggest two improvements to increase security. The first involves inserting random data values into the protocol, and the second involves the help of a semi-trusted third party. Finally, given that quantum computers are not currently commercially available, we explore options for multiparty computation using classical resources and cloud-based quantum computing. We modify the circular MPQC protocol to function for fully classical clients using blind quantum computing. Our method proposes a semi-trusted intermediate server to use post-selection to simulate entanglement between two quantum servers.

Acknowledgements

Completing this degree almost completely online due to COVID has been an uphill battle. My heartfelt gratitude goes out to my supervisor, Dr Shohini Ghose, for giving encouragement and motivation when I needed it (and the occasional kick in the pants when I needed it too!). A huge thanks also to my co-supervisor, Dr Robert Mann, for being a steady and calming presence at times when I felt like I was losing my mind. Thanks also to Dr Atefeh Mashatan for helpful conversations and guidance, and to Dr Norbert Lutkenhaus for graciously agreeing to serve on my committee. I also wish to thank Dr Hussein Abulkasim for his help with the material in chapter 3. His expertise and experience were incredibly helpful. Thanks also to Marcus Edwards and Sayan Gangopadhyay for their advice regarding material in chapter 4.

On a personal level, I wish to thank my parents Marcus and Heather for constantly supporting me, especially my mother who has had to put up with constant phone calls and venting. I thank my dear friend Emily for her love and support as well, and I thank my wonderful Chicory for their love, taking me grocery shopping, and letting me constantly follow them to their job at Starbucks while writing this thesis. Finally, I thank my delightful kitty Bobby for getting me out of bed in the mornings and always being available for snuggles.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Cryptography is the process of securing private data against attacks. Quantum cryptography applies the principles of quantum mechanics to this process in order to further improve the security of the data. An early notable quantum cryptographic protocol is quantum key distribution (QKD), which can be compared with the classical Diffie-Hellmann key exchange [10]. The principles of quantum mechanics render QKD information-theoretically guaranteed secure [6, 21]; mathematically there is no way for a QKD protocol to be hacked when run on perfect devices over perfect channels. This represents an ideal degree of security, and a theoretical improvement over the classical version of key distribution. Unfortunately other quantum cryptographic protocols cannot be guaranteed information-theoretically secure. One specific example is that of multi-party quantum computation (MPQC), which includes any cryptographic protocol that allows multiple users to perform calculations on their combined data without revealing their individual private information. It is known that MPQC cannot be guaranteed secure by the principles of quantum mechanics, due to the fact that MPQC is a specific form of bit commitment, which is not information theoretically secure [18, 20]. An important field of study is therefore developing methods of MPQC that have the highest degree of security possible in order to protect the individual privacies of members of a group.

Because there is no one single way to guarantee security for a multiparty computation scheme, there are many different possible forms such a scheme can take. This thesis examines three questions related to usage of a circular structure for MPQC. In the next section we introduce circular structure in the context of key agreement.

1.1 Key Agreement and Circular Structure

Key agreement is a cryptographic structure in which multiple parties agree on a key in such a way that each one of them equally affects the generation of the key [17]. In other words, if Alice has a key k_1, Bob has a key k_2, and Charlie has a key k_3, the three of them collaborate to create a key $f(k_1, k_2, k_3)$ which they share [26]. Each party participates equally in the key generation process in order to ensure fairness. There are four necessary principles for a key agreement protocol to function successfully [27, 13]:

1. Security – the data must be secure against outside attacks.
2. Privacy – the personal data of each user must not be accessible to other users.
3. Fairness – all users should contribute equally to the generation of the shared key.
4. Correctness – the final shared key should be accurate and function for all users.

There are three main types of structures a key agreement protocol can have [16]. Firstly, there is the complete graph structure, where each user sends every other user a sequence containing the information of their personal secret key. Secondly, there is the tree structure, in which one single user sends data to every other user. Lastly, there is the circular structure, where each user sends a sequence in turn to the next user. The circular structure is of the most interest, as it is efficient (each user only has to send/receive a communication to/from one other user) [16]. Though the tree structure also involves the same number of transmissions, tree structure is vulnerable to a specific type of attack, the detection bits chosen attack. Circular structure is not vulnerable to this type of attack, and is also more fair as each user is sending and receiving and equal amount of information. Circular structure is thus preferable to tree structure.

In a circular QKA protocol, each user must first generate and encode their own sub-key. Each user then sends this sub-key to the next user, who will then perform some sort of function or process on the sub-key, creating a new evolved sequence that the user will then send to the next user. In this way the data passes around the "circle" of users, changing and evolving with each user, until it returns to the first user, closing the circle and resulting in the final evolved key [3, 16].

The circular structure is of interest as the amount of information transmitted from user to user is small when compared to the other structures, as each user communicates with just one other user. This keeps circularly structured protocols efficient and easier to protect against attacks. Circular protocols are therefore a natural fit for MPQC as the flow of information in these structures is well suited to group computation.

1.2 Privacy-Preserving Quantum Multi-Party Computation based on Circular Structure

Here we present an example of MPQC based on circular structure. The goal is to focus on the preservation of the users' privacy, while still ensuring accurate computational results. [9]. The protocol is probabilistic; only a subset of the full results is available at the end of each iteration of the protocol.

1.2.1 Circular MPQC Protocol Steps

In this scenario, there are n users who wish to compute some sort of statistical function f on their data. Each of the n users holds m statistical data items, indexed as $i = 0...m-1$.
Step 1 User 1 has the data set $\{x_1[i]\}$. Based on this data set they prepare the state

$$\sum_{i=1}^{m-1} |i\rangle|x_1[i]\rangle = |0\rangle|x_1[0]\rangle + |1\rangle|x_1[1]\rangle + ... + |m-1\rangle|x_1[m-1]\rangle \tag{1.1}$$

which they then send to User 2.
Step 2 User 2 has their personal data set $\{x_2[i]\}$, which they combine with what they received from User 1, performing the function f and resulting in the state

$$\begin{aligned}\sum_{i=1}^{m-1} |i\rangle|x_1[i]\rangle|f(x_1[i], x_2[i])\rangle = |0\rangle|x_1[0]\rangle|f(x_1[0], x_2[0])\rangle + ...+ \\ |m-1\rangle|x_1[m-1]\rangle|f(x_1[m-1], x_2[m-1])\rangle\end{aligned} \tag{1.2}$$

Because User 2 does not know $\{x_1[i]\}$, they cannot remove it from the state; this protects both User 1's private data and the results of the function f. User 2 send this state to User 3.
Steps 3...n-1 Each user repeats these steps in turn; receiving a state from the user before them, performing the function f, and sending the resulting state to the next user.
Step n User n has the data set $\{x_n[i]\}$. They perform f on their data and the state they received from User $n-1$, resulting in the state

$$\begin{aligned}\sum_{i=1}^{m-1} |i\rangle|x_1[i]\rangle|f(x_1[i], ..., x_n[i])\rangle = |0\rangle|x_1[0]\rangle|f(x_1[0], ..., x_n[0])\rangle + ...+ \\ |m-1\rangle|x_1[m-1]\rangle|f(x_1[m-1], ..., x_n[m-1])\rangle\end{aligned} \tag{1.3}$$

which they send to User 1.
Step n+1 User 1 receives the state from User n. User 1 can then delete the $|x_1[i]\rangle$ information because they know the values, and measure to get the result of the sum for one of the $i = 1...m$ indices. Figure 1.1 illustrates the form of the protocol and the flow of information.

1.2.2 Notes on Circular MPQC and Classical MPC

Some restrictions must be placed upon the function f, chiefly that the function must be a linear one. This may seem strict, but considering the possible uses of this protocol, it should not prove too burdensome a restriction. For example, in a healthcare situation perhaps a hospital wishes to know the average/greatest/least/median height and weight of newborns born in the hospital. Perhaps in a banking scenario the bank wishes to determine the average/greatest/least/median value stored in chequing and savings accounts. From these examples we can see that restricting f to be linear should not affect the desired functionality of the protocol. A further restriction on the protocol, introduced in chapter 3, will be reversibility; we wish for f to be a reversible function.

The probabilistic nature of this protocol means that there is no way to pick which of the indices is selected for measurement. If User 1 wishes to receive statistics for all m indices, the protocol must run multiple times. The protocol requires at minimum m iterations to provide User 1 with the full results.

In general, due to the circular structure, later users have no way of knowing what the individual personal data of previous users is, protecting all users' privacy. As previously mentioned, the circular structure requires a relatively small amount of information to be transmitted which also improves the security of the scheme.

While multi-party computation can be achieved classically, the quantum process has some advantages. These advantages come from the nature of quantum states themselves. The superposition of states that is sent from user to user prevents all the information being sent from being accessed at the same time. Measuring one data item destroys the information about the other data items, thus protecting the information from malicious parties. This cannot be achieved classically, and is thus a quantum tool for data protection. Quantum superposition is integral to the successfully preserving privacy and forms the heart of the protocol. In this case, quantum is not being used to make the calculation faster, as with the quantum speedups seen with Grover's search or Shor's algorithm. Here its used to keep the calculation private.

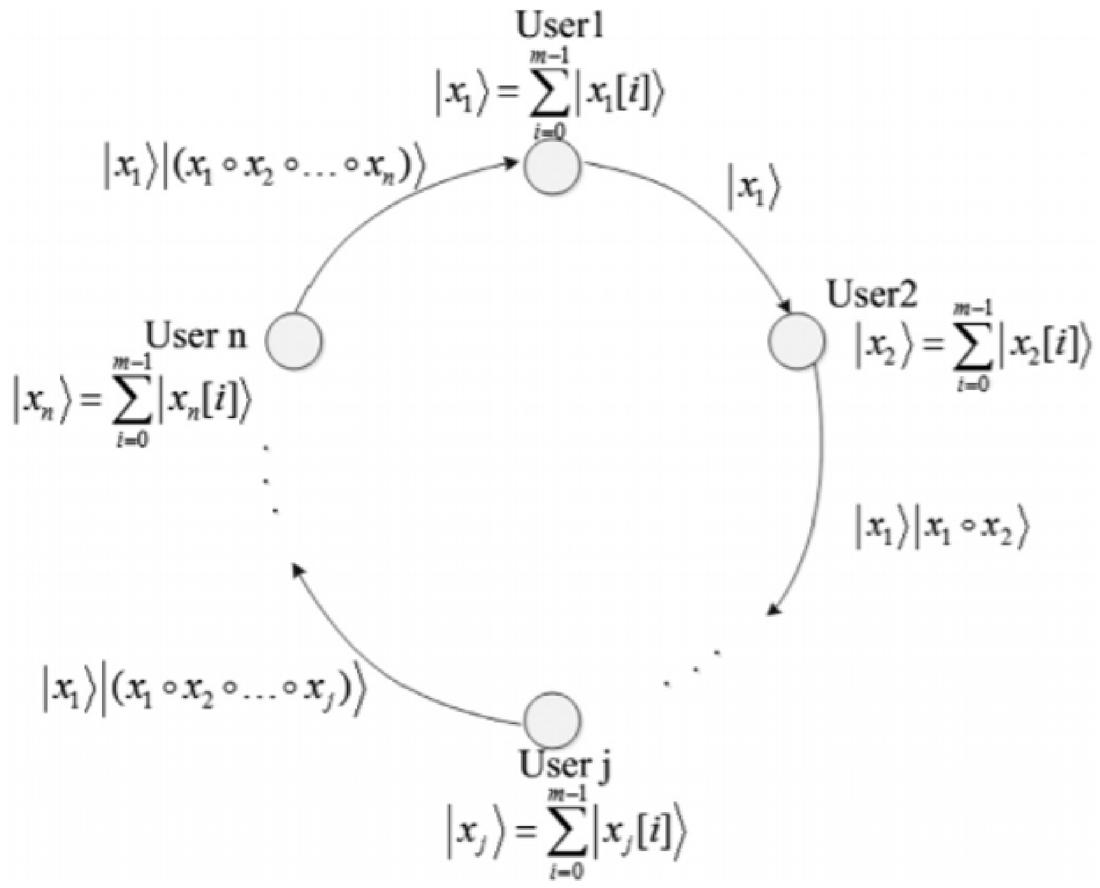

User1
$|x_1\rangle = \sum_{i=0}^{m-1} |x_1[i]\rangle$
$|x_1\rangle$
User2
$|x_2\rangle = \sum_{i=0}^{m-1} |x_2[i]\rangle$
$|x_1\rangle|x_1 \circ x_2\rangle$
User j
$|x_j\rangle = \sum_{i=0}^{m-1} |x_j[i]\rangle$
$|x_1\rangle|(x_1 \circ x_2 \circ \ldots \circ x_j)\rangle$
User n
$|x_n\rangle = \sum_{i=0}^{m-1} |x_n[i]\rangle$
$|x_1\rangle|(x_1 \circ x_2 \circ \ldots \circ x_n)\rangle$

Fig. 4. Circle structure of PQMC.

Figure 1.1: An example of circular structure, reproduced from [9]. The $|x_1\rangle$ between User1 and User2 is analogous to equation 1.1. The $|x_1\rangle|x_1 \circ x_2\rangle$ between User2 and User3 is analogous to equation 1.2. Finally, the $|x_1\rangle|(x_1 \circ x_2 \circ ... \circ x_n)\rangle$ is analogous to equation 1.3.

1.3 Overview of Thesis

This thesis addresses three questions related to circular MPQC.

In chapter 2, we build a quantum circuit to implement circular MPQC. We provide a review of the circuit model for quantum computing, and introduce the software used to build the circuit. We note that this protocol will require a different circuit depending on the desired statistical function f; our circuit executes binary addition. We first build the circuit for a test case of $n = 3$ users, $m = 2$ indices, and $p = 1$ qubit of data for each user. We then discuss qubit and gate scaling, and construct a larger circuit of $n = 4$, $m = 3$, and $p = 2$ to demonstrate the scaling. We assess the feasibility and reasonable efficiency of this scheme when executed as a circuit.

Chapter 3 addresses security questions related to MPQC. We first examine the security analysis presented in [9] and note the vulnerabilities already identified there. We then propose a simple change to the protocol that eliminates these vulnerabilities. We next examine the security analysis presented in [4] and note that the protocol is still vulnerable to both malicious action by an eavesdropper and collusive attacks by multiple dishonest users. We then propose an update to the improvements presented in [4]. Our version adds authentication, allowing a greater degree of security as users can now verify each other. This prevents imposters from joining the circle and impersonating valid users.

Finally, Chapter 4 examines how circular MPQC can be adjusted for fully classical clients and an untrusted quantum server. We introduce blind quantum computing (BQC) and provide a brief review of some important schemes for BQC. BQC for fully classical clients is often performed using entangled quantum servers; we note that server entanglement is impractical given near-term hardware constraints. We examine a method for BQC that uses the post-selection loophole to simulate entanglement, and identify inefficiencies within this method. We then propose an amendment to the simulated entanglement method that eliminates the inefficiency.

Chapter 2

The Quantum Circuit Model and Circular MPQC

While there are numerous models for quantum computing, the circuit model is a primary model among them because it is a "universal language for describing sophisticated quantum computations." [22] Classically, circuits are made up of bits that encode information, gates that perform operations on the bits, and measurements that extract information from the bits. Similarly, quantum circuits are made up of qubits that encode quantum information, quantum gates that perform operations upon the qubits, and measurements that extract information from the qubits [1, 22]. The circuit model is a useful tool for illustrating the performance of a specific protocol and demonstrating the protocol's feasibility and efficiency. It allows for quantification of resources necessary to execute a protocol (through number of gates used or circuit depth) [22]. Thus it is useful to translate from the theoretical language of a protocol to the more practical language of the circuit model, as it allows easier visualization of the processes.

Though quantum computers are not yet commercially available, cloud-based software options allow the construction of virtual quantum circuits. These services can either simulate the circuit or delegate it to an actual quantum processor for execution. In this chapter, we will use such a system to build a quantum circuit for circular MPQC in the specific case that the statistical function f is binary addition. First we will provide a review of qubits and quantum gates.

2.1 Qubits and the Bloch Sphere

As mentioned above, the qubit is analogous to the classical bit. The qubit has two possible states which can be represented as state vectors:

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix} \; |1\rangle = \begin{bmatrix}0\\1\end{bmatrix} \tag{2.1}$$

The qubit q can also be in a superposition of these two states, expressed as $|q\rangle = \alpha|0\rangle + \beta|1\rangle$, where $|\alpha|^2$ represents the probability that the qubit is in state $|0\rangle$ and $|\beta|^2$ represents the probability that the qubit is in state $|1\rangle$. α and β can be complex numbers, and $|\alpha|^2 + |\beta|^2$ must be equal to 1, as the total probability of measuring a $|0\rangle$ or a $|1\rangle$ must be 1 [22]. Upon measurement, the superposition collapses and the qubit is either measured as $|0\rangle$ or $|1\rangle$ [9, 1].

One common qubit representation is the Bloch sphere as it allows graphical visualization of the single qubit state. In order to use the Bloch sphere we express our qubit in spherical coordinates. We restrict α and β to the set of real numbers, and establish a term ϕ to represent the phase between α and β [1]. This gives us

$$|q\rangle = \alpha|0\rangle + e^{i\phi}\beta|1\rangle \tag{2.2}$$

Because $|\alpha|^2 + |\beta|^2 = 1$, we can write $\alpha = \cos\frac{\theta}{2}$ and $\beta = \sin\frac{\theta}{2}$, which allows us to write

$$|q\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle \tag{2.3}$$

We can then use spherical coordinates (r, θ, ϕ) to plot the qubit on a graph (as the magnitude of the qubit is 1, we set $r = 1$). Any single qubit state can thus be plotted on the surface of a sphere of $r = 1$ centered around the origin. This sphere is called the Bloch sphere [1]. A visual representation of the Bloch sphere is presented in Figure 2.1.

2.2 Quantum Gates

A single-qubit quantum gate can be thought of as performing a rotation of the qubit upon the Bloch sphere. Because these gates represent rotations, they are always reversible [1]. Single-qubit quantum gates can be represented as 2x2 matrices that act upon the two-dimensional qubit state vectors. All gates, single- or multi-qubit, must be unitary,

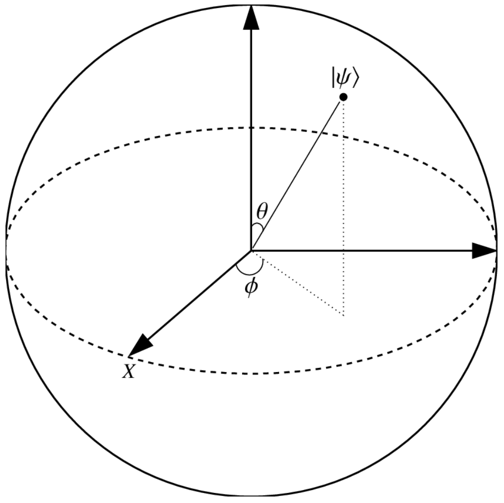


Figure 2.1: Representation of a qubit $|\psi\rangle$ on the Bloch sphere. Reproduced from [2]

indicating that their matrices must also be unitary. This means that a matrix U representing a quantum gate must satisfy the requirement $U^{\dagger}U = I$, where I is the identity matrix and $U^{\dagger}$ indicates the adjoint of U found by taking the transpose and then the complex conjugate of U. Gates must be unitary in order to preserve the normalization of the qubit or qubits upon which they act [22].

Any 2x2 unitary matrix can represent a valid single-qubit gate; we will present four commonly used examples here. These are the three Pauli gates (Pauli-X, Pauli-Y, and Pauli-Z), and the Hadamard gate. The Pauli gates are interesting because they each represent a rotation around one of the axes of the Bloch sphere, and the Hadamard gate is interesting because it creates an equal superposition of $|0\rangle$ and $|1\rangle$.

Below is the Pauli-X matrix:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

The resulting quantum gate that performs this matrix upon a qubit is often called the NOT gate, as it performs analogously to a classical NOT or bit-flip gate. If we have a qubit $|q\rangle = \alpha|0\rangle + \beta|1\rangle$, and perform the operation $X|q\rangle$, the result is $X|q\rangle = \beta|0\rangle + \alpha|1\rangle$. In terms of the Bloch sphere, this gate performs a rotation about the x-axis.

Below is the Pauli-Y matrix:

$$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

The effect of the Y gate is to perform a rotation of the qubit around the y-axis. If we have a qubit $|q\rangle = \alpha|0\rangle + \beta|1\rangle$, and perform the operation $Y|q\rangle$, the result is $Y|q\rangle = -\beta i|0\rangle + \alpha i|1\rangle$.

Below is the Pauli-Z matrix:

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

The effect of the Z gate is to perform a rotation of the qubit around the z-axis. If we have a qubit $|q\rangle = \alpha|0\rangle + \beta|1\rangle$, and perform the operation $Z|q\rangle$, the result is $Z|q\rangle = \alpha|0\rangle - \beta|1\rangle$.

Below is the Hadamard matrix:

$$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

The effect of the Hadamard gate upon a qubit is to produce an equal superposition of computational basis states (in this case $|0\rangle$ and $|1\rangle$). If we have a qubit $|q\rangle = \alpha|0\rangle + \beta|1\rangle$, and perform the operation $H|q\rangle$, the result is

$$H|q\rangle = \frac{1}{\sqrt{2}}((\alpha+\beta)|0\rangle + (\alpha-\beta)|1\rangle) \tag{2.4}$$

We will now introduce multi-qubit gates. An important two-qubit gate is the controlled-NOT or CNOT gate. As it acts upon two qubits instead of just one, it is represented by a 4x4 matrix as opposed to the smaller 2x2 matrices that represent single-qubit gates:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

The effect of this gate is to perform a NOT gate upon a target qubit if the state of the control qubit is $|1\rangle$, and to do nothing if the state of the control qubit is $|0\rangle$. This is analogous to the classical XOR gate. Table 2.1 is the truth table of the CNOT gate.

Finally, a useful three qubit gate is the Toffoli gate, which will be necessary for our circuit. This gate functions as the CNOT gate except it requires two control qubits. In this case, a NOT gate is performed on a target qubit if the state of BOTH control qubits is $|1\rangle$. In all other cases, nothing happens. This gate is represented by a 6x6 matrix, as it

Input		**Output**	
Control	Target	Control	Target
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Table 2.1: CNOT truth table

acts upon 3 qubits:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Table 2.2 is the truth table for the Toffoli gate:

Input		**Output**	
Control	Target	Control	Target
00	0	00	0
00	1	00	1
01	0	01	0
01	1	01	1
10	0	10	0
10	1	10	1
11	0	11	1
11	1	11	0

Table 2.2: Toffoli gate truth table

The above single- and multi-qubit gates will prove more than sufficient to construct a circuit to demonstrate circular MPQC. Our circuit will only use the Hadamard, NOT, CNOT, and Toffoli gates.

2.3 Circuit Construction

Our quantum circuit will be built using IBMQ. This is a cloud-based software that allows users to construct virtual quantum circuits. These circuits can then be run on a quantum simulator or (depending on the size of the circuit) delegated to a quantum processor. The circuit will demonstrate circular MPQC as proposed in [9], where the statistical function f is binary addition.

2.3.1 Preliminaries

In preparation to construct the circuit we will divide the protocol into three sections. Breaking down the protocol in this manner allows us to construct the circuit section by section, simplifying the process. The three sections are as follows:

1. Qubit preparation
2. Computation
3. Results extraction

The computation section is the core of the protocol. This circular MPQC scheme can, in general, be used to execute any statistical function between multiple users. However in order to build a circuit, we must pick a specific function to execute. Different statistical functions will require different circuits. The function used as an example in [9] is binary addition; this is the function we have chosen to perform as well. The computation portion will therefore compute the sum of the users' data values.

2.3.2 Methods of Addition

The 2015 review of existing quantum adders by Orts et al [23] provides an excellent summary of different methods for quantum addition. Basic addition is carried out by the half-adder, which sums two binary digits, and the full-adder, which sums three binary digits. A full-adder can be constructed out of three half-adders, though this is not the optimal method. Some examples of more efficient full-adder circuits are presented in [8, 25].

Ripple-carry and carry-look-ahead adders are the two main ways used to add two (or more) n-bit numbers together. In terms of computational cost, ripple-carry adders are the most efficient way to add n-bit numbers together, and can be constructed using a series of full adders [23].

Reviewing the different principles for quantum addition give an overview of how the computation in our circuit will function. The requirements of the protocol necessitate some ingenuity in regards to how these principles are applied. The protocol necessitates a rolling sum where each user applies the function individually. We are thus not implementing a single large adder that finds the total in one step. Instead each user will require their own small adder to add their own qubits of data to the cumulative total.

2.3.3 Base Case Circuit Construction

We first wish to construct a circuit to illustrate the base case of the protocol. The protocol as described in section 1.2.1 has three main variables that will affect the size of the circuit. These are the number of users n, number of indices m, and size of data p. $n = 3$ is the minimum n that allows proper execution of the circular structure. We will call our three users Alice, Bob, and Charlie. The minimum m required to illustrate the probabilistic nature of the protocol is two. The index i will thus range from $0 - (m - 1)$. We will call the index $i = 0$ "height" and the index $i = 1$ "weight". Finally the minimum data size p is one qubit. Therefore each user will have two total qubits of data, one corresponding to $i = 0$, or height, and one corresponding to $i = 1$, or weight. Figure 2.2 displays the circuit. Outlined in blue is qubit preparation. Outlined in orange is the second section, computation. Finally, outlined in green is the final section, results extraction.

2.3.4 Explanation of Qubits and Gates

Each qubit in the circuit has a specific role. Q[0] is the index or label qubit; when $q[0] = 0$ this indicates $i = 0$ or height and when $q[0] = 1$ this indicates $i = 1$ or weight. Q[11] and q[12] are the measurement qubits, used to extract the results associated with either $i = 0$ or $i = 1$. The roles of the other qubits are detailed in Table 2.3.

The gates outlined in blue initialize the qubits. A Hadamard gate is applied to the label qubit. This creates an equal superposition of $|0\rangle$ and $|1\rangle$, resulting in an equal chance of measuring $i = 0$ or $i = 1$. The X or NOT gates initialize the values of the data qubits. In this case, the values for Alice's height, Alice's weight, and Charlie's weight are equal to 1. The other data values are all 0. The series of CNOT and Toffoli gates outlined in orange form the adder and store the height results in q[7]-q[8] and the weight results in q[9]-q[10]. Finally, the gates outlined in green extract the results. In order to obtain a result corresponding to a random index data type, the label qubit is measured. If the result is $|0\rangle$, indicating height, the data from q[7] and q[8] is transferred to q[11] and q[12],

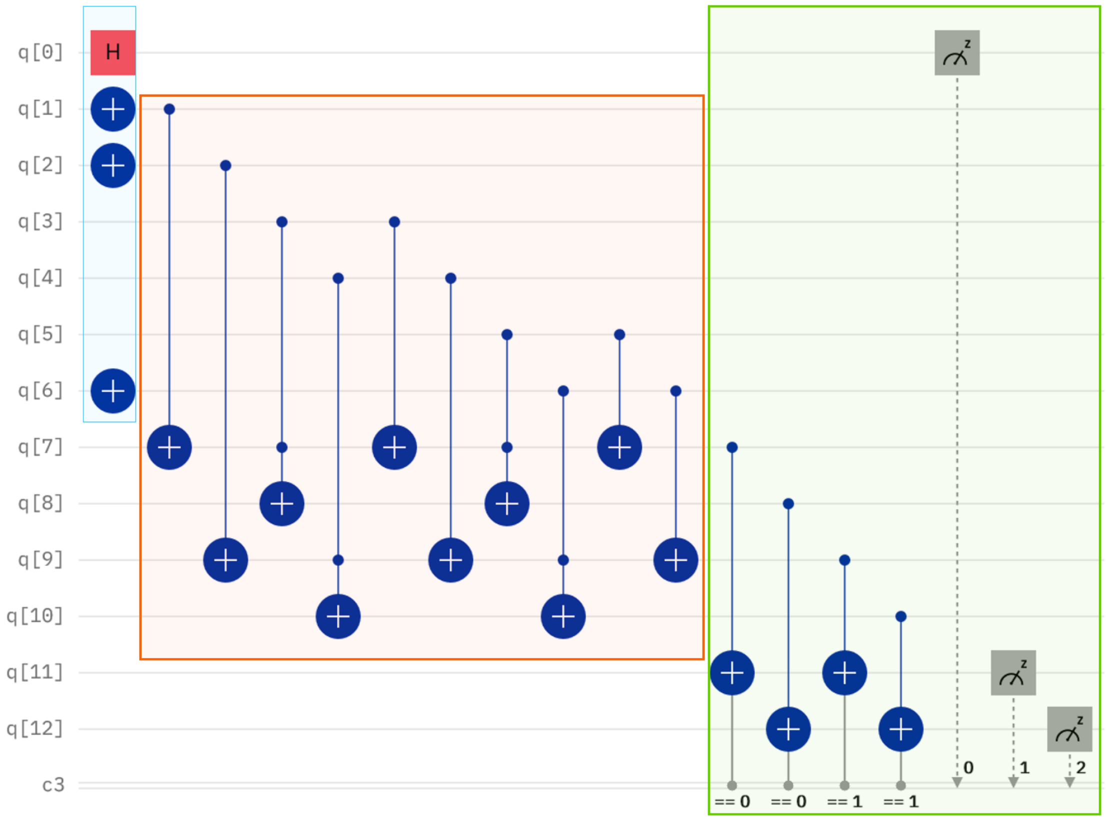


Figure 2.2: Base case circuit when $n = 3$, $m = 2$, $p = 1$ and the function f is binary addition

which are then measured. If the data qubit is $|1\rangle$, indicating weight, the data from q[9] and q[10] is transferred to q[11] and q[12] for measurement.
The expected performance will return index $i = 0$ and index $i = 1$ with equal probability. In the example circuit above, the result for $i = 0$ or height should be $|01\rangle$, and the result for $i = 1$ or weight should be $|10\rangle$.

Results for these data values can be found in Figure 2.3. The least significant bit (LSB) is the result of the label qubit's measurement, and the middle bit and most significant bit (MSB) is the result of the calculation. The left bar represents $i = 0$ or height, and the right bar represents $i = 1$ or weight. We can see that the left column has a result of 01 in its middle and MSB, and the right column has a result of 10. Thus the circuit has performed the addition calculation as expected. We also wish to ensure that the circuit is

	Qubit Roles			
	Data			Results
Index or Category	**Alice**	**Bob**	**Charlie**	
0	q[1]	q[3]	q[5]	q[7], q[8]
1	q[2]	q[4]	q[6]	q[9], q[10]

Table 2.3: Qubit Roles in Base Circuit

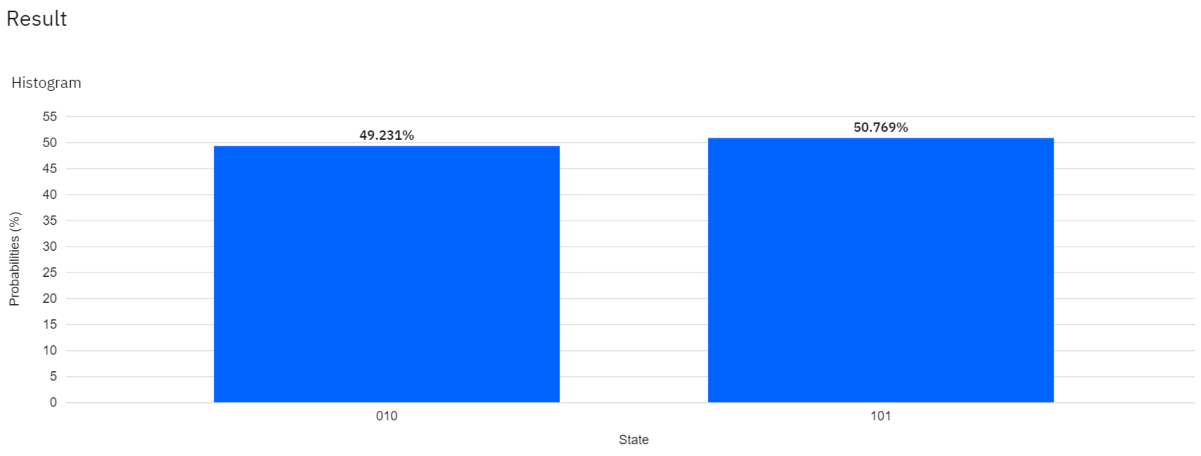


Figure 2.3: Results from example circuit

properly returning results for $i = 0$ and $i = 1$ with equal probability. It should return a measurement of $LSB = i = 0$ half the time and a measurement of $LSB = i = 1$ half the time. We observe that the probability of measuring $i = 0$ is 49.231% and the probability of measuring $i = 1$ is 50.769%, which corresponds to the desired result.

Further results for this circuit can be found in Appendix B.

2.3.5 Flow of Information

The qubits that must be transmitted from user to user as part of the computation are q[1]-q[2] and q[7]-q[10]. q[3] and q[4] remain in Bob's possession, q[5] and q[6] remain in

Charlie's possession. Alice retains the label q[0], and the measurement qubits q[11] and q[12].

Alice begins the protocol in possession of q[0] (label), q[1]-q[2] (her personal data), q[7]-q[8] (height result qubits), q[9]-q[10] (weight result qubits), and q[11]-q[12] (measurement qubits). She initializes her data as necessary using CNOT gates. She then performs the Hadamard gate on q[0] to initialize the label qubit. She then performs two CNOT gates, with her data qubits as the controls and q[7] and q[9] as the targets. This adds her data to the result qubits. Alice then sends q[7]-q[10] to Bob.

Upon receipt of q[7]-q[10], Bob performs a series of CNOT and Toffoli gates that add the information from his qubits (q[3] and q[4]) to q[7]-q[10]. He thereafter sends q[7]-q[10] to Charlie, who also performs a series of CNOT and Toffoli gates to add the information from his qubits (q[5] and q[6]) to q[7]-q[10]. Finally, Charlie sends q[7]-q[10] back to Alice.

Alice then measures the label qubit. If the measurement returns a 0, Alice uses CNOT gates to transfer the information from q[7]-q[8] to q[11]-q[12] for measurement. Similarly, if the measurement of the label qubit returns a 1, she uses CNOT gates to transfer the information from q[9]-q[10] to q[11]-q[12] for measurement. In the former case, she now has the results of the calculation for label 0, or height, and in the latter case she has the results of the calculation for label 1, or weight.

If working with a software that allows conditional measurements, the final step could be simplified. It would be more efficient to measure q[0] and based on the result subsequently measure the appropriate results qubits directly. However, the method as described above accommodates the case where direct conditional measurements are not possible.

2.4 Scaling

There are three main variables that influence the number of gates and qubits needed. These are number of users n, number of indices m, and size of data p qubits.

2.4.1 Qubit scaling

There are four different roles a qubit can fulfill in this circuit. The label qubits differentiate between the m indices. Data qubits encode each user's individual data values. Result qubits contain the rolling results of the statistical function f (in our circuit, binary addition). Measurement qubits receive the result from one index when the calculation is finished and are then measured. Each of these types scales differently with the three variables mentioned above.

Label qubits are used as a binary representation of the number of data categories. The number of users n and the data size p do not affect the number of label qubits necessary.

The amount of label qubits depends entirely on m, the number of indices, and is equal to $log_2(m) + 1$, rounded down to the nearest integer. This represents the number of qubits necessary to represent the total value of m in binary.

Note that depending on the number of indices, some values of the label qubit produce an invalid result. For example, for $m = 3$ one requires two label qubits, q_0 and q_1. The values $q_0q_1 = 00, 01, 10$ indicate valid results, and $q_0q_1 = 11$ indicates an invalid result because this value of m does not exist. This does not impact the efficiency of the circuit because there is no data associated with the invalid result of m, and thus no gates or computations associated with it either. Qubit scaling is positively affected because it takes fewer qubits to represent a number in binary than it would to represent the same number in decimal.

Data qubits are the binary representation of each user's data values, and thus the total number of data qubits required depends on all three variables. The circuit requires $n \cdot m \cdot p$ data qubits; the number of users times the number of indices times the size of the data.

Result qubits encode the rolling result of the calculation and are passed from user to user. In the base case circuit, we simply require enough result qubits to store the total sum of all data values for a particular index. However for any larger case we will also require ancillary qubits to act as "carries" to aid in the binary addition calculation. This will become apparent in section 2.5 when we introduce a scaled up version of the circuit.

Because the maximum value of a qubit is 1, we require enough results qubits to be able to represent $n \cdot m \cdot p$ in binary, or $m \cdot (log_2(n \cdot p) + 1)$ rounded down to the nearest integer.

We also require $(log_2(n \cdot p) - 1)$ ancillary qubits. For efficiency, the carry qubits are reset to 0 after each use and are then reused multiple times throughout the circuit.

Measurement qubits encode the final result of the sum of one data category. Thus we require $(log_2(n \cdot p) + 1)$ rounded down to the nearest integer measurement qubits.

Having considered all four types of qubits, we now combine the results to create an equation for the total number of qubits necessary for n users, m indices, and p data size:

$$(log_2(m) + 1) + n \cdot m \cdot p + m \cdot (log_2(n \cdot p) + 1) + (log_2(n \cdot p) - 1) + (log_2(n \cdot p) + 1) \quad (2.5)$$

with all logarithmic terms rounded down to the next integer. Note that m is the only term to appear twice outside of a logarithm; p and n only appear outside of a logarithm once each. This indicates that the number of indices has the greatest effect on the number of qubits necessary. This is a polynomial growth model (as opposed to undesirable exponential growth) indicating reasonable qubit efficiency.

2.4.2 Gate Scaling

The number of gates necessary depends upon the number of qubits. Let l be the number of label qubits, d the number of data qubits, r_1 the number of result qubits and r_2 the number of ancillary qubits, and s the number of measurement qubits. Let n remain the number of users, m remain the number of indices, and p remain the size of the data in qubits.

In the initialization portion of the circuit, each label qubit requires a Hadamard gate, and each data qubit potentially needs a NOT gate in order to represent the correct data values. Thus the initialization section requires $l + d$ gates at maximum.

The addition section requires a CNOT or Toffoli gate from each data qubit to each possible result qubit. We thus require at maximum

$$n \cdot p \cdot r_1 \tag{2.6}$$

gates for binary addition. Note that this is an upper bound, not the actual number of gates required. The true number of gates required will be less than this as users early in the circle require fewer gates than users later in the circle. This equation assumes users require gates to connect their data qubits for an index to all the results qubits for that index. However, users early in the circle do not need to be connected to all the results qubits. The maximum sum early in the circle does not require the whole amount of results qubits to represent the result of the sum in binary and thus fewer gates will be required for User 1 than for User 2, for example. As the base case circuit does not require ancillary qubits and thus does not require ancillary gates for the computation, we leave a discussion of ancillary gates to section 2.5.

For the results portion, each result qubit needs a CNOT gate to the corresponding measurement qubit, meaning we require $m \cdot s$ gates to extract the result (note that only s of those gates are ever used per pass). Finally, we require $l + s$ measurements.

The total number of gates apparently necessary at maximum is thus

$$(l + d) + (n \cdot p \cdot r_1) + (m \cdot s) + (l + s) \tag{2.7}$$

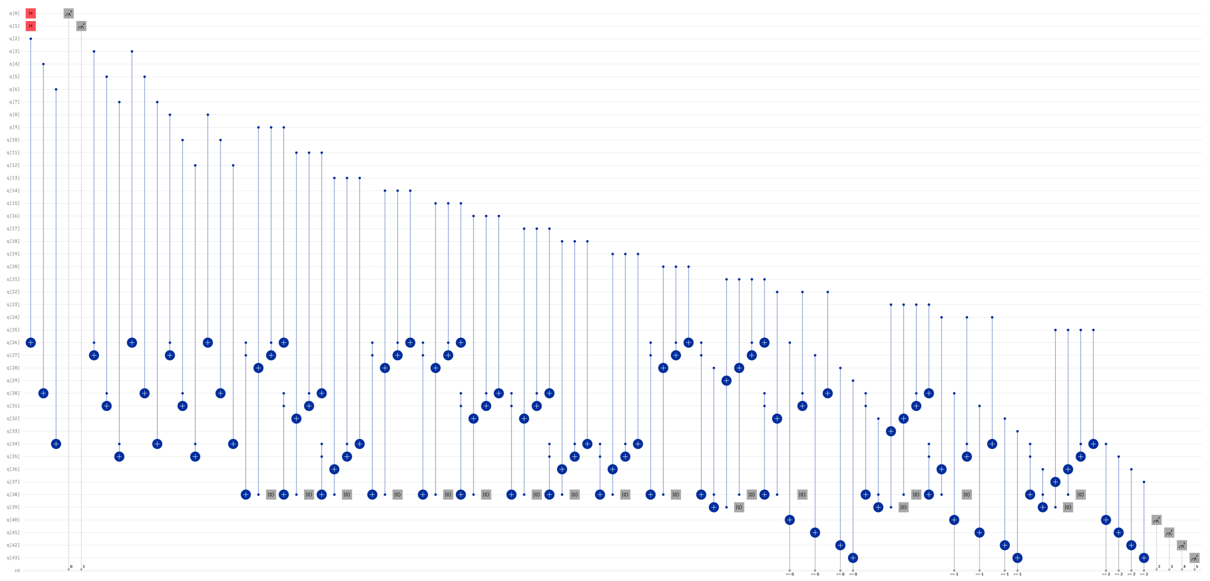


Figure 2.4: Scaled up circuit when $n = 4$, $m = 3$, $p = 2$, and the function f is binary addition

This remains a polynomial gate scaling model. In the next section, we present a scaled up circuit illustrating qubit and gate scaling. We will also discuss the scaling of the ancillary gates required for the computation.

2.5 Scaled Up Circuit

To illustrate these principles of gate and qubit scaling, we built a larger circuit, with each variable increased by one. The circuit can be seen in Figure 2.4. This version is constructed for $n = 4$ users (Alice, Bob, Charlie, and David), $m = 3$ indices (category 0, category 1, and category 2), and $p = 2$ qubits. Results for this circuit are presented in Appendix B.

We now require two label qubits, a total of twelve result qubits, four measurement qubits, and twenty-four total data qubits (six data qubits per user). We also now require two ancilla qubits to carry out the computation. Q[0] and q[1] are the label qubits, q[38]-q[39] are the ancilla qubits, and q[40]-q[43] are the measurement qubits. The roles of the other qubits are detailed in Table 2.4. In total we require 44 qubits, which can be confirmed by evaluating Equation 2.5 for $m = 3, q = 2$, and $n = 4$.

$$
\begin{aligned}
(log_2(m)+1)+n\cdot m\cdot p+m\cdot(log_2(n\cdot p)+1)+(log_2(n\cdot p)-1)+(log_2(n\cdot p)+1)\\
=(log_2(3)+1)+4\cdot 3\cdot 2+3\cdot(log_2(4\cdot 2)+1)+(log_2(4\cdot 2)-1)+(log_2(4\cdot 2)+1)\\
=2+24+12+2+4\\
=44
\end{aligned}
\tag{2.8}
$$

	Qubit Roles				
Category	Data				Results
	Alice	**Bob**	**Charlie**	**David**	
0	q[2], q[3]	q[8], q[9]	q[14], q[15]	q[20], q[21]	q[26]-q[29]
1	q[4], q[5]	q[10], q[11]	q[16], q[17]	q[22], q[23]	q[30]-q[33]
2	q[6], q[7]	q[12], q[13]	q[18], q[19]	q[24], q[25]	q[34]-q[37]

Table 2.4: Qubit Roles in Expanded Circuit

We must now address the gate scaling for the ancillary qubits. The number of gates required increases with each results qubit. Requiring a third result qubit necessitates one ancilla gate each time the third result qubit is used; the fourth result qubit requires two ancilla gates each time it is used, and so on. The gates necessary at maximum to properly carry digits are equal to

$$m\cdot n\cdot(\frac{r_1}{m}-2) \tag{2.9}$$

This is again an upper bound and not an exact amount, for the same reason as was discussed in section 2.4.2.

The newly corrected gate scaling equation is thus

$$(l+d)+(n\cdot p\cdot r_1)+(m\cdot s)+(l+s)+m\cdot n\cdot(\frac{r_1}{m}-2) \tag{2.10}$$

We can confirm this gate upper-bound equation by calculating the upper bound for the expanded circuit. We use $l=2, d=18, s=4$ and $r_1=12$, with $m=3, p=2$, and $n=4$.

$$
\begin{aligned}
(2+18)+(4\cdot 2\cdot 12)+(3\cdot 4)+(2+4)+3\cdot 4\cdot (\frac{12}{3}-2) \\
= 20+96+12+6+24 \\
= 158
\end{aligned}
\tag{2.11}
$$

Counting the gates of the expanded circuit (when all data qubits are set to 1, requiring the maximum number of initialization gates) reveals 119 gates. Thus the upper bound holds.

This remains a polynomial growth model, indicating that this method can be used to execute circular MPQC with relative efficiency when the function f is binary addition.

2.6 Conclusions

In this chapter, we have built a circuit to illustrate the execution of circular MPQC, when the computation in question is binary addition as in [9]. We have reviewed the circuit model and outlined the steps taken to construct the circuit. We have first presented a base case circuit for the minimum number of users, indices, and data values. We then discussed the qubit and gate scaling of the scheme as related to the base case. To further demonstrate the scaling, we constructed a circuit for increased numbers of users, indices, and data values. We then re-evaluated our scaling and determined that the overall scaling for both number of qubits and number of gates follows a polynomial growth model. We leave a more complete efficiency analysis and possible improvements to future work. Also left for future work is determining whether or not this method of addition is optimal.

As previously mentioned, this circular MPQC protocol could be used for other statistical functions. Some examples of possible alternative applications are finding the mean, finding the maximum value, or finding the minimum value. However, the protocol requires a different circuit for each different statistical function. Thus our circuit presented here is useful solely for calculating binary addition. Future work is necessary to extend the circuit representation to alternate statistical functions. Though the computation section represents the largest section of the circuit, the overall structure of the circuit (initialization, computation, results extraction) will remain the same no matter what function is being calculated. Thus we are able to replace the computation portion of the circuit with whatever series of gates necessary to compute any desired function, without necessitating rebuilding the whole circuit from the beginning.

Chapter 3

Security Model

The security analysis of a quantum cryptographic protocol is incredibly important. The security analysis demonstrates the strengths and weaknesses of a protocol. The process of creating the security analysis can also assist in identifying security loopholes that may be present in the protocol.

A complete security analysis considers as many different types of attacks as possible. Broadly these attacks can be split into two types: inside attacks and outside attacks. Inside attacks are those actions performed by one or more malicious users of the protocol. Outside attacks are actions performed by malicious outside parties (eavesdroppers).

The first step to proving the security of a protocol is to consider its behaviour when performed on ideal devices and channels. This is what we will examine in this chapter. We will start with the security analysis of circular PQMC provided in [9]. After identifying the loopholes specified in their analysis, we suggest a possible improvement to close those loopholes. We note that our improvement does not serve to eliminate the possibility of collusive attack as described in [4]. We then build upon the work presented in [4] to further improve the security of the protocol.

3.1 Original Security Analysis

The analysis provided in [9] only considers one type of attack from a single malicious user. The authors present two theorems:

Theorem 1. *If User 2 is malicious, he/she can only get max* $\log_i |x_1[i]|$ *bits of User 1's data at most.* [9]

Theorem 2. *If User i is malicious, he/she can only get max $\log_i |x_1[i]|$ bits of User 1's data and $\log |f|$ bits about the function $f(x_1[r], x_2[r], ..., x_{i-1}[r])$ at most, where $|f|$ means the domain size of the function f.* [9]

These theorems are proven using the Holevo bound, which gives an upper bound on the accessible information [22]. For the full proof the reader is directed to [9].

There are two main problems with this analysis. The first problem is that the analysis is too limited in scope. This analysis only considers one type of inside attack from a single malicious user. In order to complete the analysis we will consider collusive attacks between multiple malicious users. We will also consider the effect of outside attacks on the protocol. The second problem is identified in the analysis itself. The two theorems indicate that it is possible for a malicious user to steal $\log_i |x_1[i]|$ bits of User 1's data and $\log |f|$ bits of data regarding the result of the statistical function f. This is a major vulnerability. Despite the limits placed on how much data can be stolen, it is still undesirable for any amount of data to be leaked to a malicious party.

In the following section we address this second problem.

3.2 MPQC with Random Data

A very simple way to eliminate the vulnerability of User 1's data is for User 1 to send random data values to User 2 to start the protocol. User 1 can then remove the random data at the end of the protocol and replace it with their own genuine data to obtain the correct result. This small change also provides greater protection against eavesdroppers.

3.2.1 Example

We assume $n = 3$, Alice, Bob, and Charlie, and $m = 2$ indices. Users want to keep their personal data private, but they wish to compute a statistical function f on their data. In this example they wish to compute the binary sum of their data items.
Step 1 Alice has data $x_a[i]$ but she starts the protocol by generating random values $x_r[i]$

$$\sum_{i=1}^{m-1} |i\rangle|x_r[i]\rangle = |0\rangle|x_r[0]\rangle + |1\rangle|x_r[1]\rangle \tag{3.1}$$

She sends this state to Bob.

Step 2 Upon receipt of this state, Bob adds his own data $x_b[i]$, resulting in the state

$$\sum_{i=1}^{m-1} |i\rangle|x_r[i]\rangle|x_r[i] + x_b[i]\rangle = |0\rangle|x_r[0]\rangle|x_r[0] + x_b[0]\rangle + |1\rangle|x_r[1]\rangle|x_r[1] + x_b[1]\rangle \tag{3.2}$$

Bob sends this state to Charlie.

Step 3 Charlie adds in his data $x_c[i]$, resulting in the state

$$\begin{gathered}\sum_{i=1}^{m-1} |i\rangle|x_r[i]\rangle|x_r[i] + x_b[i] + x_c[i]\rangle = \\ |0\rangle|x_r[0]\rangle|x_r[0] + x_b[0] + x_c[0]\rangle + |1\rangle|x_r[1]\rangle|x_r[1] + x_b[1] + x_c[1]\rangle\end{gathered} \tag{3.3}$$

Charlie sends this state to Alice, completing the circle.

Step 4 Alice now has the final state. She measures to receive the total sum of data corresponding to either index 0 or index 1. The result of the measurement will be $x_r[i] + x_b[i] + x_c[i]$. However, she knows that this sum is incorrect because it contains the $x_r[i]$ value instead of her true $x_a[i]$ data. Alice can subtract $x_r[i]$ from the result, and replace it with $x_a[i]$, giving her the correct sum of $x_a[i] + x_b[i] + x_c[i]$.

3.2.2 General Case

Suppose there are n users (n=1, 2, 3,...,n), each with m indices ranging from $i = 0...m-1$. They wish to compute a statistical function f upon their data without compromising their individual privacy.

Step 1 User 1 has their own data values $x_1[i]$ but prepares random data values $x_r[i]$ instead, and then forms the state

$$\sum_{i=1}^{m-1} |i\rangle|x_r[i]\rangle = |0\rangle|x_r[0]\rangle + |1\rangle|x_r[1]\rangle + ... + |m-1\rangle|x_r[m-1]\rangle \tag{3.4}$$

User 1 sends this state to User 2

Step 2 User 2 adds their data $x_2[i]$ into the state and performs the function f, resulting in the state

$$\sum_{i=1}^{m-1} |i\rangle|x_r[i]\rangle|f(x_r[i], x_2[i])\rangle \tag{3.5}$$

User 2 sends this state to User 3.

Step 3 User 3 performs the same steps as User 2, sending the state on to User 4 when finished.

Steps 4-n In general, User j receives this state from User $j-1$:

$$\sum_{i=1}^{m-1} |i\rangle|x_r[i]\rangle|f(x_r[i], x_2[i], ..., x_{j-1}[i])\rangle \tag{3.6}$$

User j adds their data $x_j[i]$ and sends the state to User $j+1$:

$$\sum_{i=1}^{m-1} |i\rangle|x_r[i]\rangle|f(x_r[i], x_2[i], ..., x_{j-1}[i], x_j[i])\rangle \tag{3.7}$$

This process continues for each user until User n receives the state from User $n-1$:

$$\sum_{i=1}^{m-1} |i\rangle|x_r[i]\rangle|f(x_r[i], x_2[i], ..., x_{n-1}[i])\rangle \tag{3.8}$$

User n adds their data $x_n[i]$ and sends the resulting state back to User 1.

$$\sum_{i=1}^{m-1} |i\rangle|x_r[i]\rangle|f(x_r[i], x_2[i], ..., x_n[i])\rangle \tag{3.9}$$

Step n+1 Upon receipt of the state, User 1 measures a random index to get the associated result of the function f. However, User 1 knows that the result is not correct because it contains $x_r[i]$ instead of $x_1[i]$. As long as f is a reversible function, User 1 can remove $x_r[i]$ and replace it with $x_1[i]$, giving them the correct result.

3.2.3 Discussion

In the original protocol presented in section 1.2.1, User 1 would encode their true private data as part of the state. A malicious user could measure and obtain User 1's private information. In our updated protocol, if a malicious user measures they only obtain whatever random data User 1 has sent. This protects User 1's privacy.

Also in the original protocol, a malicious user could also measure and receive the result of the function f up to that point in the circle. In our updated protocol, measuring a random data index will give a malicious user, and an incorrect result for f because it contains the random data inserted by User 1.

It should be noted that when a malicious user measures a random index it destroys the state. This destroys the information about the data categories that were not measured. In order for the malicious user's actions to remain undetected they must recreate the state they received before sending it on to the next user. The malicious user can estimate what the maximum value of the function f to that point should be and then prepare a new state that is less than that value. However it is possible that they will recreate a state that is visibly too large. This will be a sign to User 1 at the end of the protocol that a user behaved maliciously. If this is the case, User 1 must declare the result invalid and restart the protocol.

We now establish a theorem in comparison to the theorems presented in [9]. We assume that a malicious user desires to access any private data of individual user(s) and/or information about the results of the function f. Malicious users have the same capabilities of honest users; they can send/receive qubits, perform single/double qubit operations, perform qubit measurements, and store qubits.

Theorem 1: *If User i is malicious, they can get no information about User 1's data, and whatever result they measure for f will be incorrect up to a factor determined by the size of data p.*
Proof: No information can be gained about User 1's data since it is never shared. For the second part, if a malicious user measures the result of f, this result is incorrect because it contains random data, not User 1's correct data values. The amount by which the result will differ from the correct result depends on the number of qubits p and the function f. The number of qubits p necessary to represent User 1's data (in other words, the maximum value of User 1's random data), will relate to the maximum amount by which the function f can be incorrect, depending on the nature of the function. For example, if the function f is binary addition, the maximum amount that the function will be incorrect is $\pm(2^p - 1)$.

It is evident that this change has closed the previously identified loophole that caused User 1's data to be vulnerable. Depending on the size of the random data User 1 inserts, malicious users that are later in the circle could have more ability to determine the correct value for f. As the circle progresses, the random data takes up a proportionally smaller role in the result of f, giving later users more information about the overall result. This problem can be mitigated by User 1 generating a larger random data value that is comparable in size to the final result.

We now return to the other problem we identified with the original security analysis; there are more kinds of attack that have not been considered. What if the malicious party is an outside third party; an eavesdropper? There are no protections in this protocol against eavesdropping, so an outside party can perform actions such as the intercept-and-resend attack, entangle-and-measure attack, or measure-resend attack. The improvement prevents the outside attacker from gaining any information about User 1's data or about the correct result of the function. However, an outside attacker can still attack by, for example, inserting incorrect information into the messages being shared between users. Even though the outside attacker is unable to gain valid information, they can still tamper with the results. The protocol has no eavesdropping checks, meaning there is no way for valid users to discover an eavesdropper. This causes the protocol to still be vulnerable to outside attack.

We have also not considered collusive attacks. An example of a collusive attack is given in the 2021 work by Abulkasim et al [4]. For example, assume that User 2 and User 4 are malicious, while User 3 is honest. User 2 prepares a "fake" quantum state and sends it to User 3 instead of sending the true state. User 2 also shares the fake quantum state with User 4. User 3 honestly performs the calculation as expected, and sends the evolved quantum state to User 3. The malicious User 4 now possesses the "fake" state and the state that User 3 performed an honest computation upon. User 4 can easily recover User 3's private information by comparing the states they received from User 2 and User 3 and extrapolating User 3's information from the difference.

This collusive attack is still possible in the new protocol; the addition of random data for User 1 does nothing to prevent such an attack. It should be noted that this collusive attack only works when the multiple malicious users are in the proper positions in the circle. It requires there be a single honest user between the malicious users. For example, if user 2 and 5 are malicious, and users 3 and 4 are honest, this attack only gives the malicious users at best information about the f of user 3 and 4's data; not the actual private data of either user. However if user 2 and 4 are malicious and user 3 is honest, the attack can successfully identify user 3's private data to the malicious parties.

3.3 MPQC with Third Party

The above discussion demonstrates that though the improvement we made closed the original loopholes, it still does not fully protect against outside attacks or collusive inside attacks. A further improvement to the protocol is suggested in the next section. This improvement builds upon the work done in [4], making use of a semi-trusted third party to eliminate the possibility of collusive attacks. The presence of the third party allows all data to be encrypted, thus rendering any information gained by malicious parties functionally useless. Furthermore, the suggested improvement below also authenticates all parties involved in the protocol, allowing an even greater level of security.

3.3.1 Two Party Case

Assume two users, Alice and Bob, with two private data sets $x[i]$ and $y[i]$, respectively, where $i = 0, 1, ..., m - 1$ and m refers to the data categories. The two users want to execute some secure computation (e.g., summation) on their private data $x[i]$ and $y[i]$ using a quantum function f. A semi-honest third party (STP) is used to help participants complete computation on the private data successfully. The STP pre-shares two encryption keys, $k_a[i]$ and $k_b[i]$, with Alice and Bob, respectively, using QKD [6, 14]. The STP also pre-shares k_{auth} with both Alice and Bob. k_{auth} will be used to determine the initial and measurement bases of the decoy qubits with sufficient length. The STP uses a quantum direct secure communication protocol [19] to pre-share k_{auth}, which will also be employed in the authentication process.

A robust quantum private secure computation protocol should satisfy the following requirements:

· Security: the private inputs of participants cannot be leaked out to eavesdroppers without being caught.

· Privacy: the private information of participants must be protected.

· Correctness: the proposed protocol must guarantee the correctness of the computational result.

Assumptions:

· The third party is semi-honest, which means that they can execute the steps of the protocol correctly but wants to acquire the private data of the participants. The STP is not allowed to conspire with any dishonest participants.

· The final result of the computation can be publicly announced.

· The quantum channels are optimal.

The detailed steps of the proposed protocol can be described as follows:

Step 1 STP generates a random data set $z[i]$, and from that data set generates the quantum state $|z[i]\rangle$ where

$$|z[i]\rangle = \sum_{i=0}^{m} |0\rangle|z[0]\rangle + |1\rangle|z[1]\rangle + ... + |m-1\rangle|z[m-1]\rangle \tag{3.10}$$

STP then generates m decoy states selected randomly from one of the four quantum states $|0\rangle$, $|1\rangle$, $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. STP inserts the decoys into $|z[i]\rangle$ at random positions, creating the set of states $|z_d[i]\rangle$. STP then sends $|z_d[i]\rangle$ to Alice.

Step 2 ***Checking eavesdropping and authentication*** After receiving the quantum states from STP, Alice sends a confirmation to STP and asks him to announce the positions of the decoy qubits. Alice measures the decoys according to the measurement bases indicated in the authentication key (k_{auth}) she received from STP. She publicly announces the measurement outcomes of decoy qubits to STP. STP and Alice compare the measurement outcomes with the initial bases to compute the error ratio. If the error ratio exceeds a predefined threshold value they end the protocol and try again. If the error ratio is small enough, STP and Alice believe that the quantum channel is secure against eavesdroppers. Since only legitimate users have k_{auth}, STP authenticates Alice after making sure that the error rate is less than a predefined value. Finally, Alice discards the decoys and continues with the protocol.

Step 3 Alice uses the random key TP sent her ($k_a[i]$) to encrypt her data by summing each component of $k_a[i]$ with her data $x[i]$, getting the data set $x_{k_a}[i]$. Once her data is encrypted, she generates the state

$$|x_{k_a}[i]\rangle = \sum_{i=0}^{m} |0\rangle|x_{k_a}[0]\rangle + |1\rangle|x_{k_a}[1]\rangle + ... + |m-1\rangle|x_{k_a}[m-1]\rangle \tag{3.11}$$

She then performs the function f on $|x_{k_a}[i]\rangle$ and the state $|z[i]\rangle$ she received from STP, getting the state

$$|f(x_{k_a}[i], z[i])\rangle = \sum_{i=0}^{m} |0\rangle|f(x_{k_a}[0], z[0])]\rangle + |1\rangle|f(x_{k_a}[1], z[1])\rangle + ... + |m-1\rangle|f(x_{k_a}[m-1], z[m-1])\rangle \tag{3.12}$$

Just as STP did, Alice generates m decoy states and inserts them into $|f(x_{k_a}[i], z[i])\rangle$ at random positions, resulting in the state $|f_d(x_{k_a}[i], z[i])\rangle$. She then sends this state to Bob.

Step 4 Bob and Alice repeats step 2 with the state Alice sent to Bob. After they make sure that the communication channel is secure, Bob authenticates Alice and discards the decoy qubits getting $|f(x_{k_a}[i], z[i])\rangle$

Step 5 Bob encrypts his data in the same way that Alice did, resulting in the state

$$|y_{k_b}[i]\rangle = \sum_{i=0}^{m} |0\rangle||y_{k_b}[0]\rangle + |1\rangle||y_{k_b}[1]\rangle + ... + |m-1\rangle||y_{k_b}[m-1]\rangle \tag{3.13}$$

He performs the function f on his data and the data he received from Alice, resulting in the set of states

$$f(x_{k_a}[i], y_{k_b}[i], z[i])\rangle = \sum_{i=0}^{m} |0\rangle|f(x_{k_a}[0], y_{k_b}[0], z(0)]\rangle + |1\rangle|f(x_{k_a}[1], y_{k_b}[1], z[1])\rangle + ...+ \\ |m-1\rangle|f(x_{k_a}[m-1], y_{k_b}[m-1], z[m-1])\rangle \tag{3.14}$$

He generates m decoy states and inserts them at random positions, resulting in the state $|f_d(x_{k_a}[i], y_{k_b}[i], z[i])\rangle$, which he then sends to the STP.

Step 6 STP repeats step 2 with the state they received from Bob. After they make sure that the communication channel is secure,STP authenticates Bob and discards the decoy qubits getting $|f(x_{k_a}[i], y_{k_b}[i], z[i])\rangle$

Step 7 STP now has the final set of states, $|f(x_{k_a}[i], y_{k_b}[i], z[i])\rangle$. STP measures to get the final result. These values are of course not the correct solution to the function f because Alice and Bob's data values were encrypted, and STP included random data values at the beginning.. TP must remove the encryption by subtracting $k_a[i]$, $k_b[i]$, and $z[i]$ from $f(x_{k_a}[i], y_{k_b}[i], z[i])$, resulting in

$$f(x[i], y[i]) = f(x[0], y[0]), f(x[1], y[1]), ..., f(x[m-1], y[m-1]) \tag{3.15}$$

which is the desired result. STP can then announce the result to Alice and Bob.

3.3.2 General Multi-Party Case

Assume n users with data sets $x_1[i], x_2[i], ..., x_n[i]$, where i ranges from 0 to $m-1$ and indicates the number of items in each data set. The users wish to execute some secure computation using a quantum function f. A semi-honest third party (STP) is used to help participants complete computation on the private data successfully. The STP pre-shares n encryption keys $k_{x_1}[i], k_{x_2}[i], ..., k_{x_n}[i]$ with each user respectively, using QKD [6, 14]. The STP also pre-shares k_{auth} with all n users. k_{auth} will be used to determine the initial and measurement bases of the decoy qubits with sufficient length. The STP uses a quantum direct secure communication protocol [19] to pre-share k_{auth}, which will also be employed in the authentication process.

Step 1 STP generates a random data set $z[i]$, and from that data set generates the quantum state $|z[i]\rangle$ where

$$z[i]\rangle = \sum_{i=0}^{m} |0\rangle|z[0]\rangle + |1\rangle|z[1]\rangle + ... + |m-1\rangle|z[m-1]\rangle \tag{3.16}$$

STP then generates m decoy states selected randomly from one of the four quantum states $|0\rangle$, $|1\rangle$, $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. STP inserts the decoys into $|z[i]\rangle$ at random positions, creating the set of states $|z_d[i]\rangle$. STP then sends $|z_d[i]\rangle$ to User 1.

Step 2 ***Checking eavesdropping and authentication*** After receiving the quantum states from STP, User 1 sends a confirmation to STP and asks them to announce the positions of the decoy qubits. User 1 measures the decoys according to the measurement bases indicated in the authentication key k_{auth} she received from STP. They publicly announce the measurement outcomes of the decoy qubits to STP. STP and User 1 compare the measurement outcomes with the initial bases to compute the error ratio. If the error ratio exceeds a predefined threshold value they end the protocol and try again. If the error ratio is small enough, STP and User 1 believe that the quantum channel is secure against eavesdroppers. Since only legitimate users have k_{auth}, STP is able to authenticate User 1 after making sure that the error rate is less than the threshold value. User 1 discards the decoys, and continues with the protocol.
Step 3 User 1 uses the random key TP sent them ($k_{x_1}[i]$) to encrypt their data by summing each component of $k_{x_1}[i]$ with their data $x_1[i]$, getting the data set $x_{k_1}[i]$. Once their data is encrypted, they generate the quantum state

$$|x_{k_1}[i]\rangle = \sum_{i=0}^{m} |0\rangle|x_{k_1}[0]\rangle + |1\rangle|x_{k_1}[1]\rangle + ... + |m-1\rangle|x_{k_1}[m-1]\rangle \tag{3.17}$$

They then perform the function f on $|x_{k_1}[i]\rangle$ and the state $|z[i]\rangle$ they received from STP, getting the state

$$|f(x_{k_1}[i], z[i])\rangle = \sum_{i=0}^{m} |0\rangle|f(x_{k_1}[0], z[0)]\rangle + |1\rangle|f(x_{k_1}[1], z[1])\rangle + ... + |m-1\rangle|f(x_{k_1}[m-1], z[m-1])\rangle \tag{3.18}$$

User 1 generates m decoy states and inserts them into $|f(x_{k_1}[i], z[i])\rangle$ at random positions, resulting in the state $|f_d(x_{k_1}[i], z[i])\rangle$. They then send this set of states to User 2.

Step 4 User 2 repeats step 2 with the state they received from User 1. After ensuring that the communication channel is secure, User 2 authenticates User 1 and discards the decoy qubits, getting $|f(x_{k_1}[i], z[i])\rangle$

Step 5 User 2 encrypts their data in the same way that User 1 did, resulting in the state $|x_{k_2}[i]\rangle$. They performs the function f on their data and the data they received from User 1, resulting in the state

$$f(x_{k_a}[i], x_{k_2}[i], z[i])\rangle = \sum_{i=0}^{m} |0\rangle|f(x_{k_1}[0], x_{k_2}[0], z[0)]\rangle + |1\rangle|f(x_{k_1}[1], x_{k_2}[1], z[1])\rangle + ... + |m-1\rangle|f(x_{k_1}[m-1], x_{k_2}[m-1], z[m-1])\rangle \tag{3.19}$$

They generate m decoy states and insert them at random positions, resulting in the state $|f_d(x_{k_1}[i], x_{k_2}[i], z[i])\rangle$, which they then send to User 3.
Steps 6...2n+1 Each user repeats steps 2 and 3, ensuring the communication channel is secure, authenticating the user before them, encrypting their data, performing function f on their own encrypted data and the state they just received, generating and inserting decoys, then sending the new set of states to the next user, with User n sending the final set to STP.

Step 2n+2 STP repeats step 2 with the set of states they received from User n. After ensuring that the communication channel is secure, STP authenticates User n and discards the decoy qubits, getting $|f(x_{k_1}[i], x_{k_2}[i], ..., x_{k_n}[i], z[i])\rangle$

Step 2n+3 STP now has the state

$$\begin{aligned}|f(x_{k_1}[i], x_{k_2}[i], ..., x_{k_n}[i], z[i])\rangle = \sum_{i=0}^{m} |0\rangle|f(x_{k_1}[0], x_{k_2}[0], ..., x_{k_n}[0], z[0])]\rangle + \\ |1\rangle|f(x_{k_1}[1], x_{k_2}[1], ..., x_{k_n}[1], z[1])\rangle + ... + \\ |m-1\rangle|f(x_{k_1}[m-1], x_{k_2}[m-1], ..., x_{k_n}[m-1], z[m-1])\rangle\end{aligned} \tag{3.20}$$

STP measures to receive the result of the calculation, $f(x_{k_1}[i], x_{k_2}[i], ..., x_{k_n}[i], z[i])$. These values are of course not the correct solution to the function f because the users' data values were encrypted, and STP included random data values at the beginning. STP must remove the encryption by subtracting all $k_n[i]$ and $z[i]$ with the set of states, resulting in

$$\begin{aligned}f(x_1[i], x_2[i], ..., x_n[i]) = f(x_1[0], x_2[0], ..., x_n[0]), f(x_1[1], x_2[1], ..., x_n[1]), ..., \\ f(x_1[m-1], x_2[m-1], ..., x_n[m-1])\end{aligned} \tag{3.21}$$

which is the desired result. STP can then announce the result to the users.

3.3.3 Illustrative Example

Consider three users, Alice, Bob, and Charlie, as well as a semi-trusted third party STP. Let $m = 2$ and let Alice, Bob, and Charlie have data $a[i]$, $b[i]$, and $c[i]$. The STP shares encryption keys $k_a[i], k_b[i]$, and $k_c[i]$ with each user respectively. The STP also shares k_{auth} with each user.

User	**Data**	$k_n[i]$	k_{auth}
Alice	$\|01\rangle, \|11\rangle$	$\|11\rangle, \|10\rangle$	$\|1001\rangle$
Bob	$\|00\rangle, \|10\rangle$	$\|01\rangle, \|11\rangle$	$\|1001\rangle$
Charlie	$\|11\rangle, \|01\rangle$	$\|00\rangle, \|01\rangle$	$\|1001\rangle$

Table 3.1: Data values and keys

The STP begins the protocol by inserting four decoy qubits randomly into their data set. They then send the state to Alice, announcing the positions of the decoys, allowing Alice to measure the decoys according to the measurement bases indicated in the authentication key (k_{auth}). She publicly announces the measurement outcomes of decoy qubits to STP and they compare the measurement outcomes with the initial bases to compute the error ratio. Since only legitimate users have k_{auth}, STP authenticates Alice after making sure that the error rate is less than the predefined threshold value. Finally, Alice discards the decoys and continues with the protocol.

Alice now encrypts her data by adding $k_a[i]$ to $a[i]$, resulting in $a_{k_a} = |100\rangle, |101\rangle$. She adds this to the $z[i]$ she received from STP, resulting in $|110\rangle, |110\rangle$. She generates four decoy qubits; for ease of illustration let them all be $|0\rangle$. Alice inserts the four decoy qubits randomly into $|110\rangle, |110\rangle$, resulting in the state $|10100\rangle, |00110\rangle$, and sends this result to Bob. Once Bob has received the protocol, Alice announces the positions of the decoy qubits, and Bob measures them according to the bases specified in k_{auth}. Once the channel is deemed secure and Bob has been authenticated by Alice, Bob discards the decoy qubits and continues.

Bob now encrypts his data by summing $k_b[i]$ and $b[i]$, resulting in $b_{k_b} = |01\rangle, |101\rangle$. Bob adds this to the $|f(a_{k_a}[i], z[i])\rangle$ that he received from Alice, resulting in $|111\rangle, |1001\rangle$. He inserts four decoy qubits randomly into this result, and sends the state to Charlie, announcing the positions of the decoys once Charlie has received the communication. Charlie then measures based on k_{auth}, and Bob authenticates Charlie. Once the channel is deemed secure, Charlie discards the decoys and continues.

Charlie encrypts his data in the same way, resulting in $c_{k_c} = |11\rangle, |10\rangle$. He adds this to $|f(a_{k_a}[i], b_{k_b}[i], z[i])\rangle$, resulting in $|1010\rangle, |1011\rangle$. He inserts four decoy qubits randomly into this result and sends the state to STP, who repeats the authentication and eavesdropping check. Once the channel is deemed secure and authentication has been performed, STP discards the decoys.

STP now measures to get the result of the calculation:

$$f(a_{k_a}[i], b_{k_b}[i], c_{k_c}[i], z[i]) = a_{k_a}[i] + b_{k_b}[i] + c_{k_c}[i] + z[i] = 1010, 1011 \tag{3.22}$$

However, STP knows that this is not the correct result, because the encryption keys and their own $z[i]$ data should not be included. STP then subtracts $z[i], k_a[i], k_b[i]$, and $k_c[i]$ from the result, to get the correct answer of

$$f(a[i], b[i], c[i]) = a[i] + b[i] + c[i] = 101, 110 \tag{3.23}$$

3.3.4 Security Analysis

We will first examine the protocol's vulnerability to external attacks. There are four main types of external attack that a malicious party (Eve) can perform. These are the

Trojan-horse attack, the measure-resend attack, the intercept-resend attack, and entangle-and-measure attack. Each type of attack will be discussed in the following section.

Trojan-horse Attack

The Trojan-horse attack exploits the vulnerability of a two-way communication channel by shining a bright light on the channel's detectors and analyzing the back-reflections. Because we only allow one-way communication in this protocol, the Trojan-horse attack cannot be used, and thus the protocol is secure against this type of attack.

Measure-Resend Attack

In a measure-resend attack, Eve intercepts the communication coming from Alice, measures the qubits in a random basis, and then sends the measurement results on to Bob. Doing so, however, changes the original states of the decoy qubits. When Bob performs the eavesdropping check in step 2, the presence of Eve will thus be identified, and the protocol can be scrapped and restarted. Eve's presence will be detected with a probability of $1-(\frac{3}{4})^d$, where d is the number of decoy qubits.

Intercept-Resend Attack

For an intercept-resend attack, Eve steals the message sent from Alice and sends a forged message on to Bob. Because of the authentication step however, it will become immediately apparent that the decoy qubits were not initialized in the correct measurement bases or inserted into the correct positions when Bob does the eavesdropping check. As in the measure-resend attack, when Eve's position is revealed, the legitimate users scrap the protocol and restart. Eve's position will be revealed with a probability of $1 - \frac{1}{2^d}$, where d is the number of decoy qubits.

Entangle and Measure Attack

In the entangle and measure attack, Eve attempts to entangle an ancillary qubit in her possession with the qubits that make up the legitimate communication between Alice and Bob. Eve then attempts to gather information about the communication by measuring her ancillary qubit, which will give her information about the legitimate qubit she entangled the ancilla with.

Eve can accomplish this by applying a unitary operation U to entangle her ancillary system $|a\rangle$ as follows:

$$U|0\rangle|a\rangle = \alpha|0\rangle|a_{00}\rangle + \beta|1\rangle|a_{01}\rangle \tag{3.24}$$

$$U|1\rangle|a\rangle = \gamma|0\rangle|a_{10}\rangle + \delta|1\rangle|a_{11}\rangle \tag{3.25}$$

$$U|+\rangle|a\rangle = \frac{1}{\sqrt{2}}(\alpha|0\rangle|a_{00}\rangle + \beta|1\rangle|a_{01}\rangle + \gamma|0\rangle|a_{01}\rangle + \delta|1\rangle|a_{11}\rangle)$$
$$= \frac{1}{2}[|+\rangle((\alpha|a_{00}\rangle + \beta|a_{01}\rangle + \gamma|a_{01}\rangle + \delta|a_{11}\rangle) + |-\rangle((\alpha|a_{00}\rangle - \beta|a_{01}\rangle + \gamma|a_{01}\rangle - \delta|a_{11}\rangle)] \tag{3.26}$$

$$U|-\rangle|a\rangle = \frac{1}{\sqrt{2}}(\alpha|0\rangle|a_{00}\rangle + \beta|1\rangle|a_{01}\rangle - \gamma|0\rangle|a_{01}\rangle - \delta|1\rangle|a_{11}\rangle$$
$$= \frac{1}{2}[|+\rangle((\alpha|a_{00}\rangle + \beta|a_{01}\rangle - \gamma|a_{01}\rangle - \delta|a_{11}\rangle) + |-\rangle((\alpha|a_{00}\rangle - \beta|a_{01}\rangle - \gamma|a_{01}\rangle + \delta|a_{11}\rangle)] \tag{3.27}$$

where $|\alpha|^2+|\beta|^2 = |\gamma|^2+|\delta|^2 = 1$ and $|a_{00}\rangle, |a_{01}\rangle, |a_{10}\rangle, |a_{11}\rangle$ are the ancilla states selected by Eve. The problem with this attack is that Eve obviously wishes to go undetected by the eavesdropping check. In order to do this, she sets $\beta = \gamma = 0$ if the target legitimate qubit is $|0\rangle$ or $|1\rangle$, and sets $(\alpha|a_{00}\rangle - \beta|a_{01}\rangle - \gamma|a_{10}\rangle + \delta|a_{11}\rangle) = (\alpha|a_{00}\rangle - \beta|a_{01}\rangle + \gamma|a_{10}\rangle - \delta|a_{11}\rangle) = 0$ when the target legitimate qubit is $|+\rangle$ or $|-\rangle$.

However, this strategy will be unsuccessful, because setting $\beta = \gamma = 0$ means that $|\alpha|^2 = |\delta|^2 = 1$, indicating that $\alpha|a_{00}\rangle = \delta|a_{11}\rangle$. This means that Eve will not be able to distinguish between $\alpha|a_{00}$ and $\delta|a_{11}\rangle$, and will subsequently not be able to determine any information about the target legitimate qubit. The protocol is thus safe against the entangle and measure attack.

Collusive Attack

Two dishonest participants may try to adopt the collusive attack strategy in [3]. For example, assume that User 1 and User 3 are malicious, while User 2 is honest. User 1 prepares fake quantum states $|f_f(x_{k_a}[i]])\rangle$ with m decoy states and sends it to User 2 instead of sending the true states $|f_d(x_{k_a}[i], z[i])\rangle$. User 1 also shares the fake quantum states with User 3. User 2 honestly encodes their private information, and sends the evolved quantum states to User 3. The malicious User 3 now possesses the original fake states and the evolved quantum states after User 2 encoded their private information. User 3 can easily recover User 2's private information by applying the subtracting function. However, this attack strategy will not succeed in disclosing the private information of User 2. Because each user encrypts their private information before encoding it, User 2's information is protected. Similarly, if $n-1$ dishonest users tried to steal the private information of the honest one, they would not succeed due to the encryption process. Thus, we can say that the proposed scheme is secure against collusive attacks.

3.4 Conclusion

Circular protocols are an excellent method for secure multiparty quantum computation, as their structure allows for efficient communication and enables the least amount of data to be sent from user to user, thus ensuring the least possible vulnerabilities to attack.

In this chapter we have examined the security of the circular MPQC protocol originally presented in section 1.2.1. We noted that the security analysis provided in [9] already identified vulnerabilities in the protocol. We then proposed a small change to the protocol that eliminated those vulnerabilities.

We also noted that the security analysis in [9] did not consider outside attacks or collusive attacks between multiple dishonest users. We have subsequently build on the work done in [4], by suggesting a circular MPQC protocol that functions with the help of a third party. The addition of the semi-trusted third party allows for data encryption to protect the individual users and also allows for an eavesdropping check and an authentication step. The eavesdropping check protects the protocol against outside attack. The authentication step allows the users to verify that there are no imposters participating in the protocol, adding an extra layer of security.

Future work can look to adding more features to the protocol, such as results verification. However, adding features comes at the cost of greater efficiency. Already in this chapter we have added decoy qubits for an eavesdropping check. This negatively affects the qubit scaling equation presented in chapter 2; requiring more qubits makes the circuit less efficient. Qubit scaling will depend on how many decoy qubits are desired; at minimum the number of results qubits should be doubled to have at least an equal number of decoy and non-decoy qubits being sent from user to user. The more decoys sent, the higher the security, but the worse the efficiency. A balance must thus be struck between maintaining efficiency while improving function and security.

Chapter 4

Blind QC

The work presented in previous chapters dealt with clients who had quantum capabilities. In the near term, assuming that clients have quantum capabilities is unrealistic, as quantum computers are not commercially available. Currently most clients have no quantum capability. However, classical clients (those with absolutely no quantum capabilities whatsoever) may still wish to make use of quantum computers through cloud based softwares. Our circuits in Chapter 2 were built using one such software. However, it cannot be assumed that these cloud based quantum servers are trustworthy. Classical clients no longer need to protect their information just from each other, but also from the quantum server itself. This necessitates Blind Quantum Computing (BQC). BQC schemes are designed to hide private data and the computation itself from a third party quantum server. In this chapter, we will examine how a circular MPQC protocol can be adapted to run blindly. First we will review some existing BQC schemes.

4.1 Background

4.1.1 Early Work

One of the earliest schemes for BQC was proposed by Broadbent et al. in 2009 [7]. This work deals primarily with a semi-classical user, which is a classical user augmented with the power of qubit preparation.

The scheme uses distributed measurement based quantum computation (MBQC). The user (Alice) prepares the qubits, the server performs entanglement and measurements, and Alice then computes the classical feedforward mechanism [7]. Normally, MBQC uses a family of *graph states* (the initial entangled states required for MBQC) called *cluster states*, which are universal for MBQC. However, the initialization of a cluster state involves some computational basis measurements, which would cause Alice to reveal information about the underlying graph state. This information leakage is undesirable, so the paper introduces a class of states called *brickwork* states. These states are universal for $X-Y$ plane measurements and therefore do not require initial computational basis measurements. Computations in this protocol are carried out then as a pattern of measurements on a brickwork state. Figure 4.1 illustrates the form of the brickwork state.

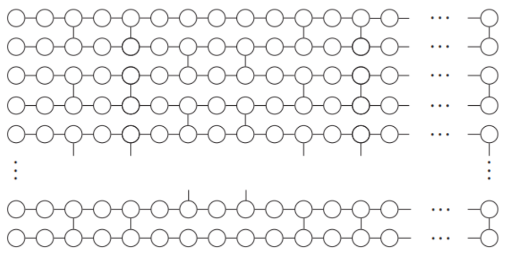


Figure 1: The *brickwork state,* $\mathcal{G}_{n\times m}$. Qubits $|\psi_{x,y}\rangle$ $(x = 1,\dots,n, y = 1,\dots,m)$ are arranged according to layer x and row y, corresponding to the vertices in the above graph, and are originally in the $|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ state. Controlled-Z gates are then performed between qubits which are joined by an edge.

Figure 4.1: Figure detailing the brickwork state. Reproduced from [7]

The structure of the protocol includes two phases; preparation, and computation. In the preparation phase, Alice prepares her qubits, and sends them to the server, who entangles them. In the computation phase, for each qubit in each layer of the brickwork state, Alice sends a classical message to the server to indicate which basis of the $X-Y$ plane should be used for measurement. The server then performs the measurement and communicates the result to Alice. The computation is finished either when all qubits are measured, or when the server returns the qubits to Alice, depending on Alice's own needs. The only information leaked is the dimensions of the brickwork state, which is not information that is useful to the server [7].

A number of modifications to the general protocol are also suggested. These include modifications to allow for quantum inputs and outputs, a modification to include authentication and fault tolerance, and a modification to function for a fully classical client. This modification uses two entangled quantum servers to perform BQC for the fully classical client. In this proposal, the BQC is performed as follows: a classical Alice delegates the qubit preparation step to server 1 and the computation step to server 2. This structure of a classical client delegating to two entangled servers is of interest, as future works build upon this foundation.

4.1.2 BQC with Entangled servers

A 2017 work from Huang et al [15] builds on the original concept from [7]. They propose a method for BQC with fully classical clients, using two entangled quantum servers. This protocol uses two quantum servers that share Einstein-Podolsky-Rosen (EPR) pairs but are separated in space so they cannot communicate. The client separates the algorithm they wish to execute into two parts, Computation A and Computation B, and delegates Computation A to the first server and Computation B to the second server. The two servers then execute their respective computations independently upon their EPR pairs, and return the measurement results to the client. Since the servers cannot communicate, they cannot combine the measurement results to get the total result of the algorithm and thus the computation is blind [24].

In order to test and ensure the honesty of the servers, the user also requires the servers to execute "dummy" protocols. There are three possible dummy protocols. The first is a simple a CHSH test. The second is a state tomography task that consists of asking server one to implement their portion of the legitimate protocol while server two performs a CHSH test. This certifies whether or not server one is performing the legitimate protocol correctly. The third is a process tomography task that consists of server two performing their portion of the legitimate protocol while server one performs the CHSH test. This certifies whether or not server two is running their legitimate protocol correctly. The user runs the legitimate protocol with a small probability η and the three dummy protocols with respective probabilities $\frac{1-\eta}{3}$ so that the servers cannot distinguish between the dummy protocols and the legitimate protocol. If the dummy protocols are executed correctly the client can determine whether or not the servers are also executing the legitimate protocol correctly [15].

This is an elegant solution for performing BQC with fully classical clients. The protocol is secure and efficient, as discussed in the supplemental material to [15]. However there are some complications, which are identified in the next section.

4.1.3 BQC with Simulated Entanglement

Practically, entangling quantum servers is not currently a realistic proposition. This type of entanglement will continually need to be "refreshed", as it will irrevocably collapse at the end of each computation. The quantum devices currently being built and developed may also not support this kind of server entanglement. Many quantum computers use highly isolated single-chip graphs of superconducting Transmon qubits. To entangle a pair of these qubits, there must be a physical channel embedded between each qubit. [11]. This would require highly specific device construction. There are thus practical considerations that make BQC using entangled servers less feasible.

A 2019 proposal [11] uses simulated entanglement to circumvent the problems encountered when using truly entangled servers. Quantum phenomena such as entanglement can be described by a local hidden-variable model, if the observers of said phenomena perform post-selection as part of their computations of correlation functions [5]. Thus the post-selection loophole can be used to simulate entanglement. An intermediate classical server is necessary in this method to delegate the algorithm to the two servers and to perform the steps of post-selection. The intermediate server splits up the algorithm between the two servers as in [15], and carries out the steps of the calculation. Once the protocol has been executed, the intermediate server instructs the servers to carry out the steps of the algorithm in reverse. In order to match the initial states of the quantum servers to an outcome observed during the forward-execution stage, state preparation using single qubit gates is performed first. Once the reversed execution is complete, the intermediate server applies a correlation function to match the results from the two servers. When the function is satisfied, entanglement is simulated between the two servers. The outcomes which satisfy the correlation function are kept, and others are discarded.

The presence of the intermediate server, however, creates a new problem. If the intermediate classical server is not trusted, we wish to hide the algorithm not only from the quantum servers but from the intermediate classical server as well. This is accomplished through an obfuscation algorithm that hides the gates of the algorithm in blocks of "noise" gates. However this method is extremely inefficient, with extra qubits and many gates necessary to execute even the simplest of algorithms, as demonstrated in [12]. This work seeks to provide a proof of concept for the simulated entanglement BQC method. Figure 4.2 shows the algorithm that is to be implemented. Figure 4.3 shows the obfuscated algorithm, run in the forwards direction. To further explain this figure, note that each of the two servers has computational and non-computational qubits, to aid in the obfuscation. The computational qubits of processor 1, mapped to the computational qubits of the algorithm in figure 4.2 are $q_0 = q_0$, $q_1 = q_3$, and $q_2 = q_2$. The computational qubits of processor 2 are $q_0 = q_4$, $q_1 = q_1$, and $q_2 = q_5$. Any noise gates applied to these qubits (such as the first Y gate applied to q_0 of processor 1) must be cancelled by applying the same gate again in order to preserve the correct results of the computation. Noise gates applied to the non-computational qubits are not required to be cancelled, but some should be in order to aid in the obfuscation. These two circuits demonstrate the large number of gates necessary to implement even a small algorithm. Note that in order to complete the computation these circuits would then have to be run in reverse, doubling the computational size.

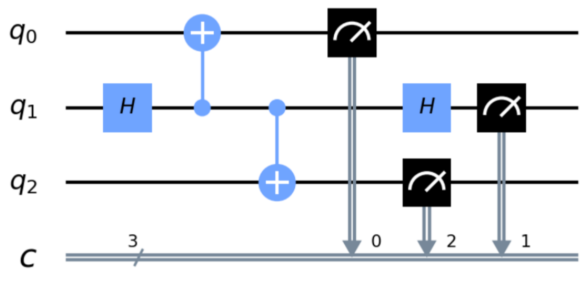
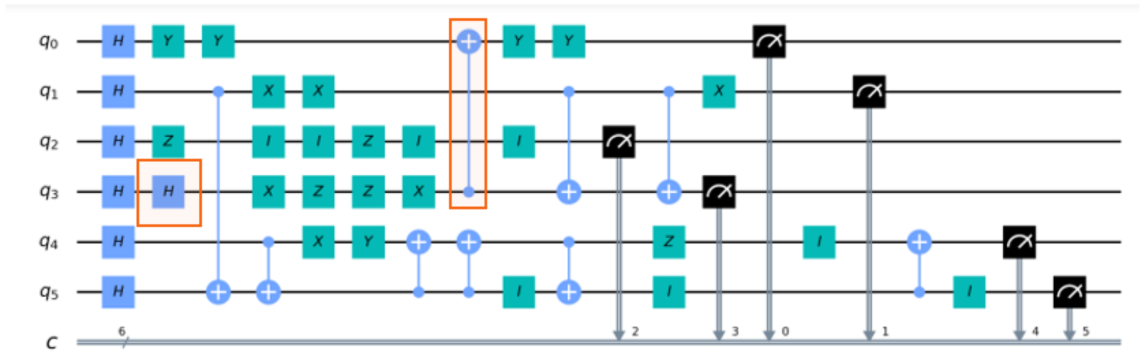


Figure 4.2: The desired algorithm, consisting of four algorithmic gates and three measurements. Reproduced from [12]

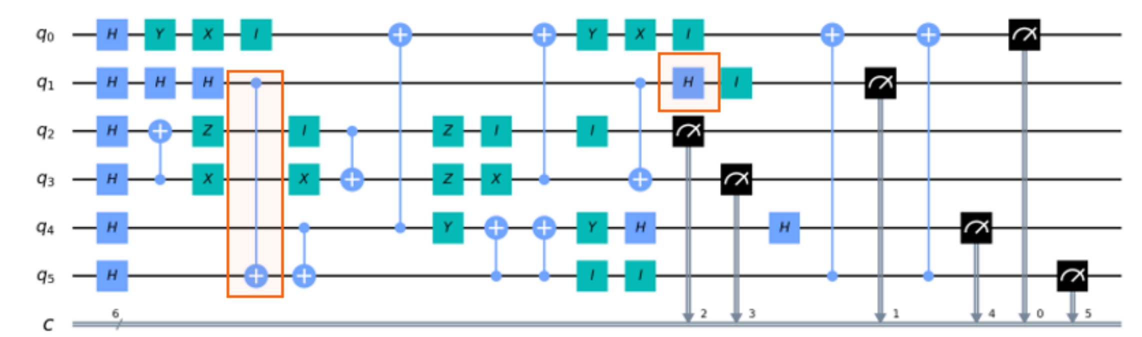
It is evident that neither of the two schemes discussed above are ideal. We now seek to combine these two methods in order to eliminate the inefficiencies in [11] while also eliminating true server entanglement. First we will discuss simulated entanglement in more detail.

4.1.4 Simulated Entanglement

We will use the above example from [12]. In this case, we have an algorithm with four gates and three qubits that we wish to execute blindly. The intermediate classical processor will split the algorithm between two quantum co-processors, named QC1 and QC2, each with six qubits. Both processors have three computational qubits and three decoy qubits. Qubit roles are assigned randomly. QC1's computational qubits are q[0], q[2], and q[3], while QC2's computational qubits are q[1], q[4], and q[5]. Next, the three qubits of the desired algorithm are mapped to the computational qubits of the two processors. This mapping is also created randomly, and is presented in table 4.1.



(a) QC1



(b) QC2

Figure 4.3: The circuits for obfuscated execution of the algorithm in Figure 4.2, distributed between two quantum servers. Note that the four algorithmic gates (highlighted in orange) are now hidden within more than sixty noise gates. Reproduced from [12]

Now that the computational qubits and their mappings have been established, the blind execution of the algorithm in Figure 4.2 can begin. As seen in Figure 4.3, all qubits are initialized to the maximally superimposed state. Next the algorithmic gates (outlined in orange in Figure 4.3) are buried within blocks of obfuscating gates. All qubits are measured, and the results of the measurements are returned to the intermediate classical server.

Next the obfuscated algorithms must be run in reverse, conditioned upon the results

Original	QC1	QC2
q[0]	q[0]	q[4]
q[1]	q[3]	q[1]
q[2]	q[2]	q[5]

Table 4.1: Mapping of original algorithmic qubits to the quantum co-processors' algorithmic qubits.

of the forward algorithm. For example, the results from QC1 were (0,1,0,0,0,1) and the results from QC2 were (0,0,0,0,1,1). The intermediate server first saves the results of the computational qubits ((0,0,0) and (1,0,1) from QC1 and QC2 respectively) then initializes the qubits for reverse execution according to the results of the forward execution. Once QC1 and QC2 have performed the process in reverse, the intermediate server will perform postselection.

In this case, the intermediate server will only select results where the QC1 computational qubits are all 0, because the initial state of the algorithmic qubits was 0. Other cases will be discarded and the protocol will repeat. Suppose the results are (0,1,0,0,0,1) and (0,0,1,1,0,0) for QC1 and QC2 respectively. Note that the results for the computational qubits (qubits 0, 3, and 2 on QC1 and qubits 4, 1, and 5 on QC2) are pairwise identical. This indicates that entanglement has successfully been simulated. The result of the calculation is thus (1,0,1), or the result of the forward execution of QC2.

4.2 Proposed Scheme

While it is true that entanglement of quantum servers is impractical, the simulated entanglement solution presented in [11] is inefficient, requiring an incredible amount of gates to execute simple algorithms. In order to retain the benefits of simulated entanglement while not encountering inefficiency, we propose making the intermediate server in [11] semi-trusted. We still wish to hide the private data of users from the intermediate server. However, the semi-trusted server is allowed knowledge of the algorithm or computation to be executed. Thus the intermediate server can delegate the steps of the algorithm to the two servers without obfuscation. This eliminates the inefficiency in [11].

The users individually generate keys to encrypt their own data. They then send their encrypted data to the intermediate classical server. The intermediate classical server then splits the computation in half and delegates it to the two quantum servers. The intermediate server processes the computation using the post-selection loophole to simulate entanglement as in [11]. When the intermediate extracts the result of the computation, it sends the result to User 1. User 1 removes their encryption key from the result and sends it to User 2. User 2 then removes their key and sends the result to User 3. This process continues around the circle until User n. User n removes the final encryption key and sends the result back to the intermediate server, who then announces the correct result to all users.

4.3 Protocol Steps

Assume n users with data sets $x_1[i], x_2[i], ..., x_n[i]$, where i ranges from 0 to $m-1$ and indicates the number of items in each data set. The users wish to execute some secure computation using a quantum function f. However, the users are fully classical and have no quantum capabilities. They will make use of an intermediate semi-trusted classical server, and two cloud-based quantum servers, to carry out said computation.

The two cloud-based quantum servers cannot communicate and do not share entanglement. The role of the intermediate server is to delegate the computation to the two quantum servers, and to use post-selection on the results to simulate entanglement.

Each classical user generates an encryption key ($k_{x_1}[i], k_{x_2}[i], ..., k_{x_n}[i]$ respectively), and encrypts their data by summing their key and their data. Each user then submits their encrypted data ($x_{k_1}[i], x_{k_2}[i], ..., x_{k_n}[i]$) to the intermediate server.

We use the basic form of the circuit from chapter 2 to describe the execution

Step 1 The qubits of each quantum server start in the maximally superimposed state. The intermediate server then instructs the quantum servers to initialize the data qubits to the correct values $x_{k_1}[i], x_{k_2}[i], ..., x_{k_n}[i]$. Results qubits and measurement qubits are initialized to $|0\rangle$.

Step 2 The intermediate server then splits up the steps of the computation and randomly delegates half of the computational and results extraction gates to the first quantum server and half to the second.

Step 3 After the quantum servers have performed their portions of the computation and reported the measurement results back to the intermediate server, the intermediate server again prepares the quantum servers in the maximally superimposed state, and then intializes them according to the observed results.

Step 4 The same algorithm is now executed in reverse, with the quantum servers again sending their measurement results to the intermediate server.

Step 5 If the measurement outcomes from the two different quantum servers satisfy a previously established correlation function, then those results simulate the effect that the quantum co-processors are working with shared entangled memories. These results are recorded and all other results are discarded. This is the post selection step which simulates the entanglement.

Step 6 Now the intermediate server holds the result of the computation that was performed with the users' encrypted data, meaning this result is not the correct sum. In order to get the correct result, the intermediate server sends the result to User 1, who subtracts their encryption key $k_{x_1}[i]$ to remove it from the result.

Step 7 User 1 now sends the result to User 2, who subtracts their encryption key $k_{x_2}[i]$ to remove it from the result. This continues around all of the users, with User n removing their encryption key and returning the result to the intermediate server.

Step 8 The intermediate server now is in possession of the correct result of the calculation, and can announce said result to all users.

4.4 Discussion

This method eliminates the need for truly entangled servers, and eliminates the need for an obfuscation algorithm and a large number of extra gates.

This method is incomplete, however, as it does not include any honesty check or "dummy" algorithms as in [15]. In order to insert dummy algorithms, we add more qubits, differentiating between "algorithmic" and "non-algorithmic" qubits. The non-algorithmic qubits can be used to perform the three tests as suggested in [15] (CHSH, state tomography, and process tomography) to test the honesty of the servers, while the algorithmic qubits are used to carry out the actual algorithm. The results from the non-algorithmic qubits will then be used to determine whether or not the servers have performed the true algorithm correctly.

4.4.1 Inclusion of dummy protocols and non-algorithmic qubits

The preparatory steps for the protocol remain the same as in section 4.3. The adjustments to include the non-algorithmic processes begin with the intermediate server.

Step 1 The qubits of each quantum server start in the maximally superimposed state. The intermediate server requires that there be n algorithmic and n non-algorithmic qubits;

the algorithmic qubits will be used for the true protocol and the non-algorithmic qubits will be used to check the honesty of the servers.

Step 2 The intermediate server splits up the steps of the algorithm and randomly delegates half of the gates to the algorithmic qubits of the first quantum server and half to the algorithmic qubits of the second server. Simultaneously, the intermediate server also randomly delegates the three dummy protocols (CHSH, state tomography, process tomography) to the non-algorithmic qubits of the appropriate servers.

Step 3 The quantum servers perform their portions of the algorithm and the required dummy protocols, and return the measurement results back to the intermediate server. The intermediate server then prepares the servers in the maximally superimposed state again, and initializes both the algorithmic and non-algorithmic qubits according to the observed results.

Step 4 The same algorithm is now executed in reverse, with the algorithmic qubits performing the true algorithm backwards and the non-algorithmic qubits performing the reverse of whatever dummy protocols they were originally delegated. The measurement results are again sent to the intermediate server.

Step 5 If the measurement results from the two servers satisfy the expected correlation function, entanglement has been successfully simulated. If the results from the non-algorithmic qubits are correct, the quantum servers have behaved honestly and the results from the algorithmic qubits are assumed to be correct as well.

From here on the steps are the same as in the general case described in section 4.3. The intermediate server can discard the results from the non-algorithmic qubits, and then proceeds to send the results to the users who remove their encryption one by one to achieve the correct result of the calculation.

4.5 Conclusion

Blind quantum computing is a vitally important facet of quantum cryptography. Because quantum computers are not universally available for commercial use, most clients do not have quantum capabilities. Those wishing to make use of the power of quantum computing must delegate their tasks to cloud-based quantum servers. These third party servers are not trusted, and the users wish to protect their data and their calculations from them. BQC allows fully classical users to protect their information when using untrusted quantum servers.

Many schemes for BQC for fully classical clients involve two entangled quantum servers. However, producing and maintaining entanglement of this kind is difficult and impractical. Using post-selection to simulate entanglement can eliminate this problem. However this method requires an intermediate server to perform post-selection. If the intermediate server is also not trusted with the data and steps of the protocol the obfuscation process requires an immense number of gates. This method thus is inefficient.

We have suggested a compromise. We trust the intermediate server with the steps of the algorithm but not the private data of the users. The users encrypt their own data, and the intermediate server delegates the algorithm to the two quantum servers. We thus have the benefits of truly entangled servers (ease of execution) combined with the benefits of simulated entanglement (no impractical true server entanglement). We also allow the intermediate server to delegate "dummy" algorithms to the two quantum servers. The dummy algorithms serve to test the quantum servers' honesty. As the quantum servers cannot differentiate between the dummy algorithms and the true calculation, if the dummy algorithms are performed correctly this indicates that the servers are behaving honestly and the true computation is also being performed correctly. Future work is necessary to determine whether this method is practical in terms of execution. The concept must be further explored to determine its efficacy. It is not proven whether this method will be able to fully eradicate the inefficiencies of the obfuscation method while still maintaining privacy. It is similarly unclear whether or not any quantum advantage is gained through using this method; would it not be simpler to allow the semi-trusted intermediate classical server to perform the computation? These questions are left to future exploration.

Chapter 5

Summary

This thesis has examined multi party quantum computation (MPQC), which is any protocol where multiple users perform calculations on their combined data without revealing individual private information. As MPQC is not guaranteed information theoretically secure [18, 20], it is necessary to find schemes that allow as high a level of security as possible in order to protect users' privacy. We have specifically explored using circular structure, which is a natural fit for MPQC. Each user must communicate with only one other user, making the circular method efficient. Because the number of communications is minimal, the amount of vulnerable transmitted information is also minimized. In this thesis, we have considered three questions related to circular MPQC.

In chapter 2, we built a quantum circuit to execute circular MPQC. The quantum circuit model is important to quantum computation for a number of reasons. It allows quantification of the computational cost of a protocol (through number of qubits and/or gates) [22]. Translating from a theoretical protocol to the language of quantum circuits also allows the protocol to then be tested on a quantum computer or simulator to ensure it functions correctly. We used IBM's virtual circuit builder for construction and simulation. The circular protocol can be used to perform any statistical calculation. However, a specific circuit is required to correspond with specific statistical functions; every statistical function requires its own quantum circuit. We chose binary addition for our statistical function, aligning with the example in [9]. We first built the circuit for the base case of $n = 3$ users, $m = 2$ indices, and $p = 1$ qubit of data each. We then examined scaling of both qubits and gates in our circuit, and built a circuit for $n = 4$, $m = 3$, and $p = 2$ to demonstrate our scaling arguments. We determined that the function can be executed with reasonable efficiency as the qubits and gates scale with a polynomial growth model (as opposed to undesirable exponential growth).

The circular nature of the protocol means that the computation must be carried out sequentially. We do not sum all of the data points at once; each user adds their data to the rolling sum individually. This means that existing methods for statistical computation may have to be adjusted slightly to fit this protocol. We examined existing methods for addition, and used them as a foundation for our method of sequential addition. Existing methods of addition function on the assumption that we start our calculation with knowledge of all the inputs. This is unsuitable to our protocol as that would compromise the privacy of the users. Thus we have constructed an adder that compiles a rolling sum of multiple inputs, in order to allow each user to add their data themselves sequentially. Further work is necessary in order to determine whether or not our method of addition is optimal. Future work will also examine building circuits to execute other statistical functions such as finding the minimum or maximum value. Though the computation will change, the principles we used to build our circuits can be used to simplify circuit construction for other statistical functions.

In chapter 3, we examined the vulnerabilities of the circular MPQC protocol presented in [9]. The security assessment presented therein only considered the threat posed by inside attacks from a single malicious user. That security analysis also already identifies weaknesses in the protocol. We presented an update to the protocol that closes the original loopholes by having User 1 insert random data into the calculation which they can then remove at the end of the protocol to ensure they still find the correct result. We then considered the updated protocol's vulnerability to external attack (eavesdroppers) and collusive attacks by multiple malicious users. The protocol did not contain an eavesdropping check and was thus vulnerable to outside attack. It was also still vulnerable to collusive attack. We then present a method to eliminate the possibility for collusive attack by use of a third party, building on the work in [4]. The third party generates keys for the users to encrypt their data. This encryption step allows the users to prevent collusive attack. Our method allows authentication to ensure that all users are legitimate. Furthermore, this method also includes an eavesdropping check in the authentication step which protects against the action of malicious outsiders. Overall we have significantly increased the security of the circular MPQC protocol, making it safe against multiple types of attack. Future work will include a verification step that allows the users to determine whether or not the calculated result is true and accurate. Future work should also investigate how to carry out statistical functions other than binary addition using this method. The method of encryption we used would not be suitable for a task such as finding the maximum or minimum data value. Our work was also carried out under the assumption of perfect device behaviour and perfect channels. This is obviously not a practical consideration, because no actual device for quantum computation behaves perfectly. Future work should also thus consider actual device behaviour to determine what new vulnerabilities device performance will introduce to the protocol, and also how to mitigate those vulnerabilities.

In chapter 4, we acknowledge that commercial availability of quantum computers has not yet been achieved. Thus most clients do not have quantum capabilities. However, classical users may still want to make use of the power of quantum computing. To do so, they must delegate their computations to cloud-based quantum servers. These third party servers may not be trusted. Therefore the users will wish to hide their private data and the details of the algorithm they wish to carry out. This can be accomplished using blind quantum computing (BQC). We briefly introduce BQC and discuss some of the possible methods. The most popular method for BQC with fully classical clients involves delegation of the computation to multiple entangled quantum servers. However this method is impractical because entanglement of servers is difficult to achieve and maintain [11]. A work from 2019 [11] suggests using post-selection to simulate entanglement. Eliminating the need for fragile server entanglement is a more practical method. This scheme requires the introduction of an intermediate server to delegate the algorithm and perform the post-selection. Because the intermediate server is not trusted, obfuscation is required to hide the data and steps of the algorithm from the intermediate server. Obfuscation requires a huge number of "noise" gates, which makes this method inefficient, as demonstrated in [12].

We wish to eliminate the need for true server entanglement, while also improving efficiency. We therefore wish to execute BQC using post-selection to simulate entanglement while no longer requiring the inefficient noise gates. We subsequently suggest a combination of methods. If the intermediate server is semi-trusted, we can allow it to know the steps of the algorithm. This eliminates the need to hide the algorithmic gates in blocks of noise gates, reducing the inefficiency. We do not trust the intermediate server with the users' private data however. Users can encrypt their own data to protect it from the intermediate server. Users then send their encrypted data to the intermediate server, who delegates the algorithm to the two quantum servers and uses post-selection to simulate entanglement. The encryption can be removed after the intermediate server has sent the results back to the users to generate the correct result of the calculation. We also suggest using "dummy" algorithms to verify that the servers are behaving honestly, as in [15]. The intermediate server can establish algorithmic qubits, which carry out the true protocol, and non-algorithmic qubits, which carry out the "dummy" protocols. If the dummy protocols are executed correctly it can be assumed that the true algorithm is also being executed correctly. In future, circuits should be constructed to execute this method as a proof of concept, similar to what was done in [12]. This method also could allow for MPQC with quantum clients on encrypted data without the use of a third party. Future work should explore this idea further to determine whether it is feasible and secure. The removal of encryption as described here may also prove problematic. As mentioned above, this method of encryption is based on binary addition as the desired calculation, and is not suitable for other statistical calculations such as finding the minimum or maximum value. Future work must then also find new ways to remove the encryption in the final step of the protocol to ensure receipt of a correct result.

Quantum cryptography is overall a fascinating subfield of quantum computing and quantum information. The extra security guaranteed for some protocols by the principles of quantum mechanics is an exciting step forward for information security. Because multi party quantum computation cannot be guaranteed information theoretically secure, finding ways to protect multiple users' privacy is an engrossing puzzle that is sure to occupy researchers for some time to come, and using the circular structure is sure to remain an important part of this puzzle.

References

[1] Amira Abbas, Stina Andersson, Abraham Asfaw, Antonio Corcoles, Luciano Bello, Yael Ben-Haim, Mehdi Bozzo-Rey, Sergey Bravyi, Nicholas Bronn, Lauren Capelluto, Almudena Carrera Vazquez, Jack Ceroni, Richard Chen, Albert Frisch, Jay Gambetta, Shelly Garion, Leron Gil, Salvador De La Puente Gonzalez, Francis Harkins, Takashi Imamichi, Pavan Jayasinha, Hwajung Kang, Amir h. Karamlou, Robert Loredo, David McKay, Alberto Maldonado, Antonio Macaluso, Antonio Mezzacapo, Zlatko Minev, Ramis Movassagh, Giacomo Nannicini, Paul Nation, Anna Phan, Marco Pistoia, Arthur Rattew, Joachim Schaefer, Javad Shabani, John Smolin, John Stenger, Kristan Temme, Madeleine Tod, Ellinor Wanzambi, Stephen Wood, and James Wootton. Learn quantum computation using qiskit, 2020.

[2] Samson Abramsky. Contextual semantics: From quantum mechanics to logic, databases, constraints, and complexity. *Contextuality from Quantum Physics to Psychology*, June 2014.

[3] Hussein Abulkasim, Atefeh Mashatan, and Shohini Ghose. Secure multiparty quantum key agreement against collusive attacks. *Scientific Reports*, 11, May 2021.

[4] Hussein Abulkasim, Atefeh Mashatan, and Shohini Ghose. Security improvements for privacy-preserving quantum multiparty computation based on circular structure. *Quantum Information Processing*, 21, Jan 2022.

[5] Sven Aerts, Paul Kwiat, Jan-Åke Larsson, and Marek Z˙ukowski. Two-photon franson-type experiments and local realism. *Phys. Rev. Lett.*, 83:2872–2875, Oct 1999.

[6] Charles H. Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. *Theoretical Computer Science*, 560:7–11, Dec 2014.

[7] Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. Universal blind quantum computation. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, Oct 2009.

[8] Steven A. Cuccaro, Thomas G. Draper, Samuel A. Kutin, and David Petrie Moulton. A new quantum ripple-carry addition circuit, 2004.

[9] Zhiliang Deng, Ying Zhang, Xiaorui Zhang, and Lingling Li. Privacy-preserving quantum multi-party computation based on circular structure. *J. Inf. Secur. Appl.*, 47(C):120–124, Aug 2019.

[10] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[11] Marcus Edwards. General entanglement-free blind delegated quantum co-processing with purely classical clients. Unpublished, 2019.

[12] Sayan Gangopadhyay. Blind qc notes. Unpublished, 2020.

[13] Cao Hao and Wenping Ma. Multiparty quantum key agreement based on quantum search algorithm. *Scientific Reports*, 7:45046, Mar 2017.

[14] Jianyong Hu, Bo Yu, Ming-Yong Jing, Liantuan Xiao, Suotang Jia, Guo-Qing Qin, and Gui Long. Experimental quantum secure direct communication with single photons. *Light: Science & Applications*, 5:e16144, Apr 2016.

[15] He-Liang Huang, Qi Zhao, Xiongfeng Ma, Chang Liu, Zu-En Su, Xi-Lin Wang, Li Li, Nai-Le Liu, Barry C. Sanders, Chao-Yang Lu, and Jian-Wei Pan. Experimental blind quantum computing for a classical client. *Phys. Rev. Lett.*, 119:050503, Aug 2017.

[16] B. Liu, D. Xiao, and HY. Jia. Collusive attacks to "circle-type" multi-party quantum key agreement protocols. *Quantum Information Processing*, 15:2113–2124, Feb 2016.

[17] Bin Liu, Fei Gao, Wei Huang, and Qiao-Yan Wen. Multiparty quantum key agreement with single particles. *Quantum Information Processing*, 12(4):1797–1805, Apr 2013.

[18] Hoi-Kwong Lo and H. F. Chau. Is quantum bit commitment really possible? *Phys. Rev. Lett.*, 78:3410–3413, Apr 1997.

[19] G. L. Long and X. S. Liu. Theoretically efficient high-capacity quantum-key-distribution scheme. *Phys. Rev. A*, 65:032302, Feb 2002.

[20] Dominic Mayers. Unconditionally secure quantum bit commitment is impossible. *Phys. Rev. Lett.*, 78:3414–3417, Apr 1997.

[21] Dominic Mayers. Unconditional security in quantum cryptography. *J. ACM*, 48(3):351–406, May 2001.

[22] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, 10th anniversary edition edition, 2010.

[23] Francisco José Orts Gómez, Gloria Ortega Lopez, Elias Combarro, and E M Garzon. A review on reversible quantum adders. *Journal of Network and Computer Applications*, 170:102810, Aug 2020.

[24] Ben W. Reichardt, Falk Unger, and Umesh Vazirani. Classical command of quantum systems via rigidity of CHSH games. *Nature*, 496:456–460, Apr 2013.

[25] Feng Wang, Mingxing Luo, Huiran Li, Zhiguo Qu, and Xiaojun Wang. Improved quantum ripple-carry addition circuit. *Science China Information Sciences*, 59, Feb 2016.

[26] Yu-Guang Yang, Bo-Ran Li, Shuang-Yong Kang, Xiu-Bo Chen, Yi-Hua Zhou, and Wei-Min Shi. New quantum key agreement protocols based on cluster states. *Quantum Information Processing*, 18(3):1–17, Mar 2019.

[27] Kun-Fei Yu, Chun-Wei Yang, Tzonelih Hwang, Chuan-Ming Li, and Jun Gu. Design of quantum key agreement protocols with strong fairness property, 2015.

APPENDICES

Appendix A

Qiskit Circuit Code

All code is for when all data qubits are 0. The users' data should be initialized to the correct values using NOT gates at the start of each circuit depending on the data values.

A.1 Base Circuit Code

```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[13];
creg c[3];


cx q[1], q[7];
h q[0];
ccx q[3], q[7], q[8];
cx q[2], q[9];
cx q[3], q[7];
ccx q[4], q[9], q[10];
cx q[4], q[9];
ccx q[5], q[7], q[8];
cx q[5], q[7];
ccx q[6], q[9], q[10];
cx q[6], q[9];
```

```
if (c == 0) cx q[7], q[11];
if (c == 0) cx q[8], q[12];
if (c == 1) cx q[9], q[11];
if (c == 1) cx q[10], q[12];
measure q[0] -> c[0];
measure q[11] -> c[1];
measure q[12] -> c[2];
```

A.2 Scaled Up Circuit Code

```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[44];
creg c[6];

h q[0];
h q[1];

cx q[2],q[26];
ccx q[3],q[26],q[27];
cx q[3],q[26];
cx q[4],q[30];
ccx q[5],q[30],q[31];
cx q[5],q[30];
cx q[6],q[34];
ccx q[7],q[34],q[35];
cx q[7],q[34];
ccx q[8],q[26],q[27];
cx q[8],q[26];
ccx q[26],q[27],q[38];
ccx q[9],q[38],q[28];
ccx q[9],q[26],q[27];
reset q[38];
cx q[9],q[26];
ccx q[10],q[30],q[31];
```

```
cx q[10],q[30];
ccx q[30],q[31],q[38];
ccx q[11],q[38],q[32];
ccx q[11],q[30],q[31];
reset q[38];
cx q[11],q[30];
ccx q[12],q[34],q[35];
cx q[12],q[34];
ccx q[34],q[35],q[38];
ccx q[13],q[38],q[36];
ccx q[13],q[34],q[35];
reset q[38];
cx q[13],q[34];
ccx q[26],q[27],q[38];
ccx q[14],q[38],q[28];
ccx q[14],q[26],q[27];
reset q[38];
cx q[14],q[26];
ccx q[26],q[27],q[38];
ccx q[15],q[38],q[28];
ccx q[15],q[26],q[27];
reset q[38];
cx q[15],q[26];
ccx q[30],q[31],q[38];
ccx q[16],q[38],q[32];
ccx q[16],q[30],q[31];
reset q[38];
cx q[16],q[30];
ccx q[30],q[31],q[38];
ccx q[17],q[38],q[32];
ccx q[17],q[30],q[31];
reset q[38];
cx q[17],q[30];
ccx q[34],q[35],q[38];
ccx q[18],q[38],q[36];
ccx q[18],q[34],q[35];
reset q[38];
cx q[18],q[34];
```

```
ccx q[34],q[35],q[38];
ccx q[19],q[38],q[36];
ccx q[19],q[34],q[35];
reset q[38];
cx q[19],q[34];
ccx q[26],q[27],q[38];
ccx q[20],q[38],q[28];
ccx q[20],q[26],q[27];
reset q[38];
cx q[20],q[26];
ccx q[26],q[27],q[38];
ccx q[28],q[38],q[39];
ccx q[21],q[39],q[29];
reset q[39];
ccx q[21],q[38],q[28];
ccx q[21],q[26],q[27];
reset q[38];
cx q[21],q[26];


ccx q[30],q[31],q[38];
ccx q[22],q[38],q[32];
reset q[38];
ccx q[22],q[30],q[31];
cx q[22],q[30];

ccx q[30],q[31],q[38];
ccx q[32],q[38],q[39];
ccx q[23],q[39],q[33];

ccx q[23],q[38],q[32];
ccx q[23],q[30],q[31];
reset q[38];
reset q[39];
cx q[23],q[30];

ccx q[34],q[35],q[38];
ccx q[24],q[38],q[36];
ccx q[24],q[34],q[35];
reset q[38];
cx q[24],q[34];

ccx q[34],q[35],q[38];
ccx q[36],q[38],q[39];
ccx q[25],q[39],q[37];
ccx q[25],q[38],q[36];
ccx q[25],q[34],q[35];
cx q[25],q[34];
reset q[38];
reset q[39];
measure q[0] -> c[0];
measure q[1] -> c[1];
if (c==0) cx q[26],q[40];
if (c==0) cx q[27],q[41];
if (c==0) cx q[28],q[42];
if (c==0) cx q[29],q[43];
if (c==1) cx q[30],q[40];
if (c==1) cx q[31],q[41];
if (c==1) cx q[32],q[42];
if (c==1) cx q[33],q[43];
if (c==2) cx q[34],q[40];
if (c==2) cx q[35],q[41];
if (c==2) cx q[36],q[42];
if (c==2) cx q[37],q[43];
measure q[40] -> c[2];
measure q[41] -> c[3];
measure q[42] -> c[4];
measure q[43] -> c[5];
```

Appendix B

Circuit Results

B.1 Base Case

Because there are three users each with one qubit per index, the sum result values will range from 0 to 3 (or 00 to 11 in binary). The index should have a roughly even split between 0 and 1. The least significant bit (LSB) is the label, the other two are the sum results.

B.2 Scaled Up Case

Because there are four users each with two qubits per index, the sum result values will range from 0 to 8 (or 0000 to 1000 in binary). Since there are only three valid index values, the result associated with an index of q[0]q[1]=11 will always return 0000 and be discarded as it is invalid. As in the previous example, the two LSBs are the label qubits; with 00, 01, and 10 representing valid labels. In every case the circuit was run on IBMQ's simulator with 8192 shots. As expected, the 8192 shots are roughly evenly distributed between the four existing label values, although there is a consistent small bias towards the q[0]q[1]=11 result because that index has no data values associated with it so the system prefers to choose that path as it is the most efficient.

Figures A.1 through A.6 provide a selection of results for the base circuit, and figures A.7 through A.12 provide a selection of results for the scaled up circuit. These demonstrate the correct addition results as well as the correct correlation with the label qubits.

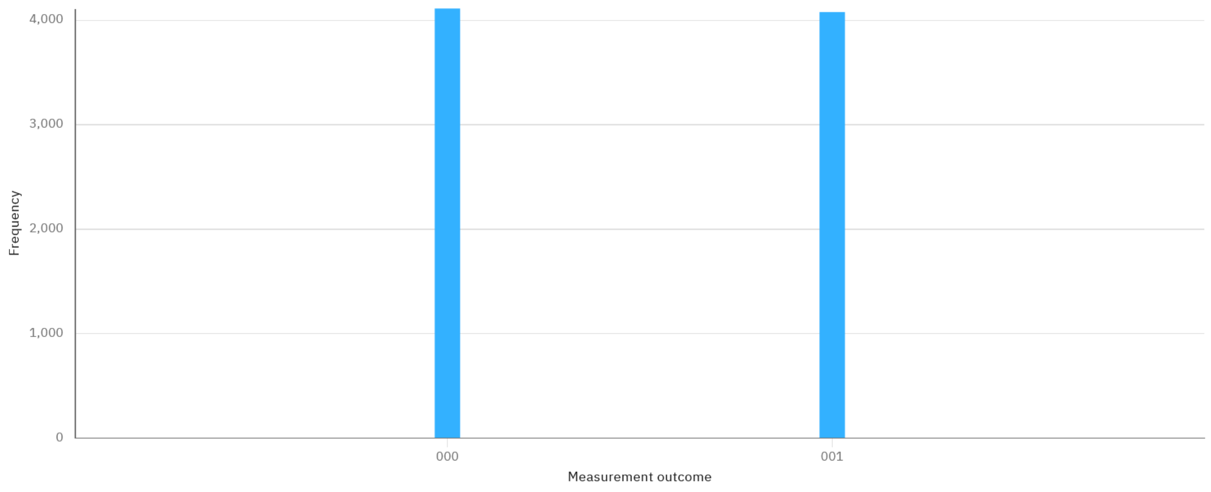


Figure B.1: Results when all input data is $|0\rangle$

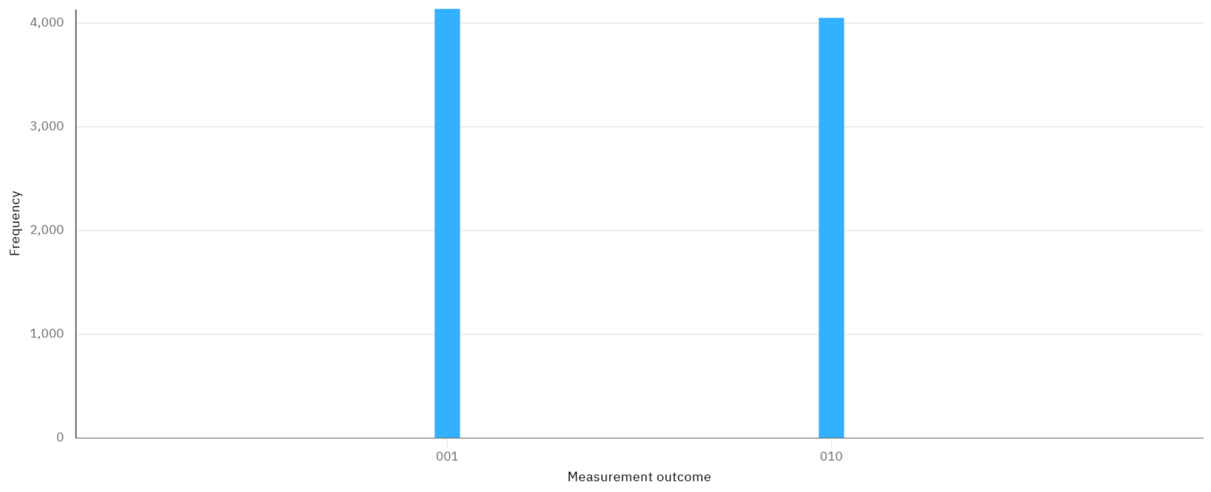


Figure B.2: Results when q[3] is $|1\rangle$

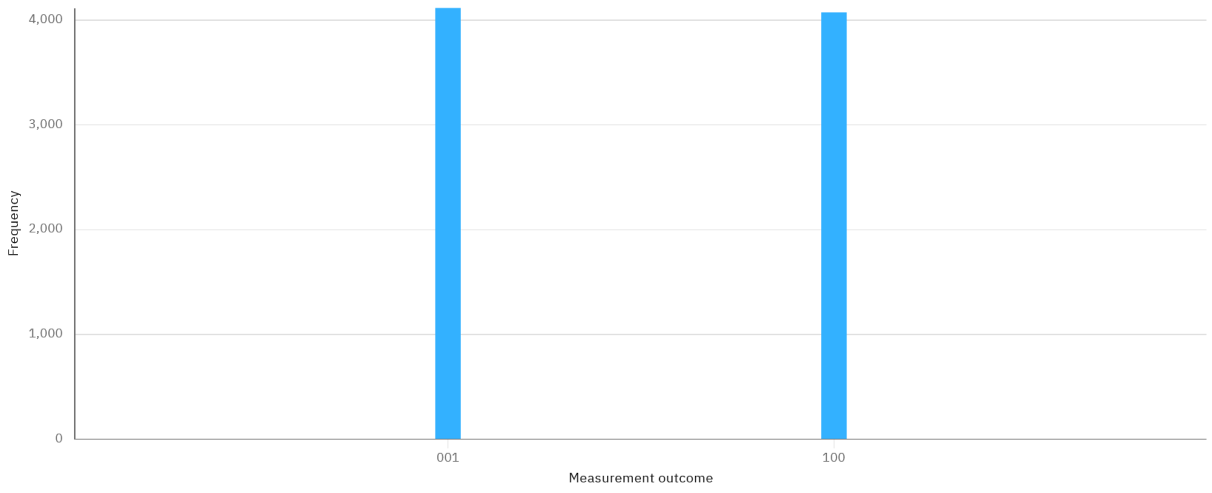


Figure B.3: Results when q[1] and q[3] are $|1\rangle$

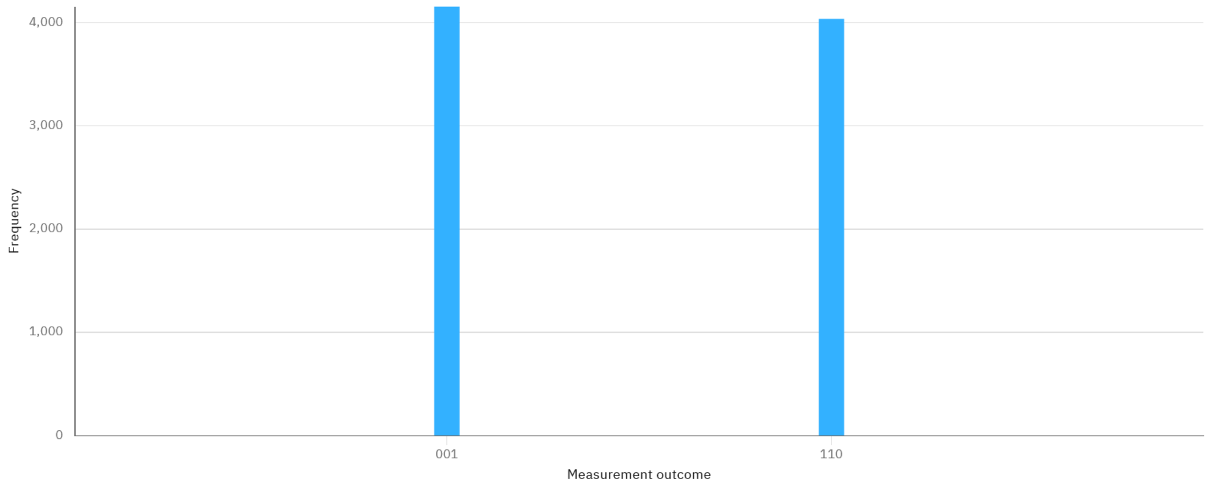


Figure B.4: Results when q[1], q[3], and q[5] are $|1\rangle$

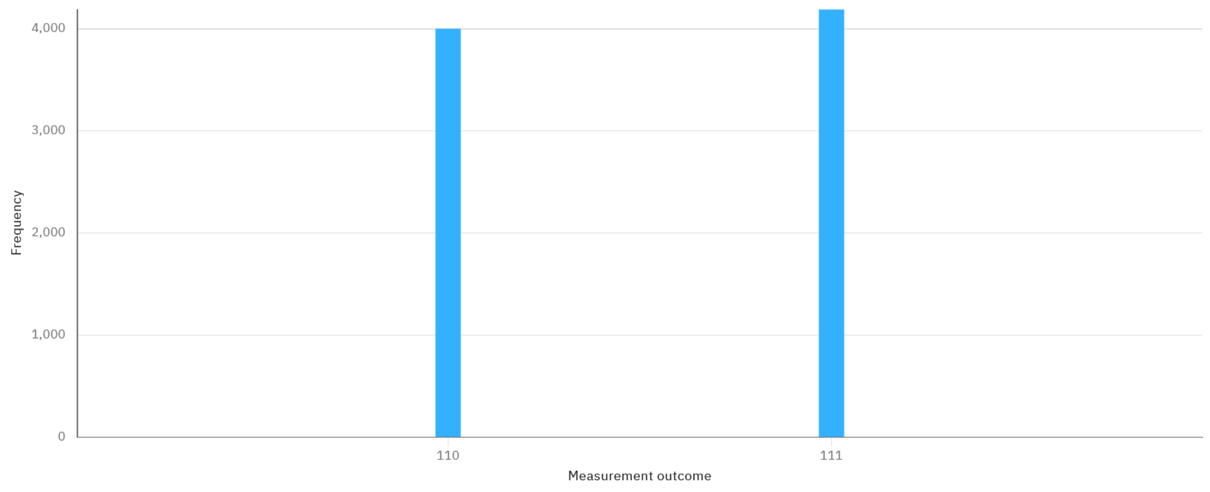


Figure B.5: Results when all data qubits are $|1\rangle$

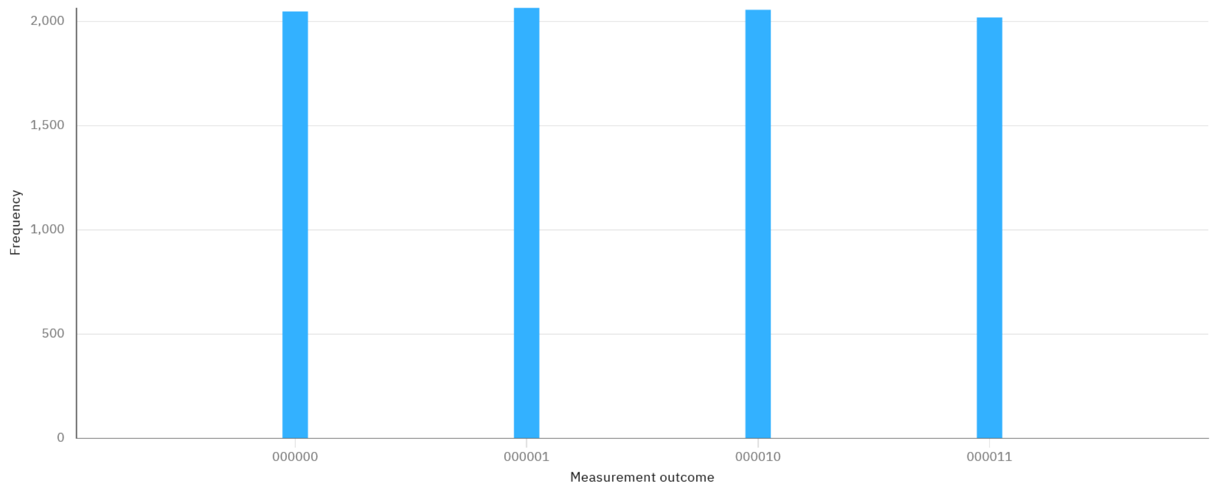


Figure B.6: Results when all input data is $|0\rangle$

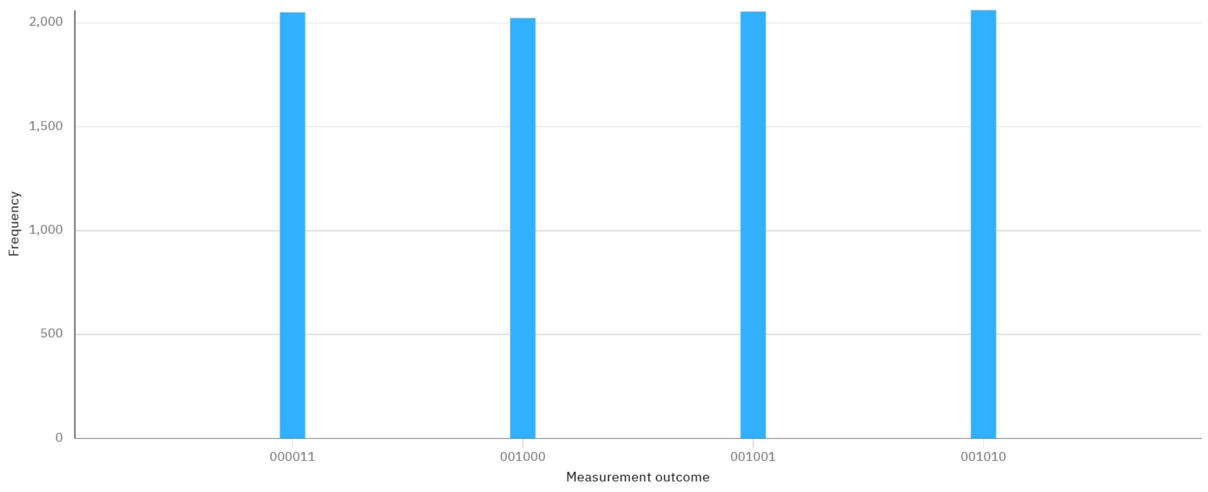


Figure B.7: Results when data qubits 2, 4, 10, 14, 18 and 24 are $|1\rangle$

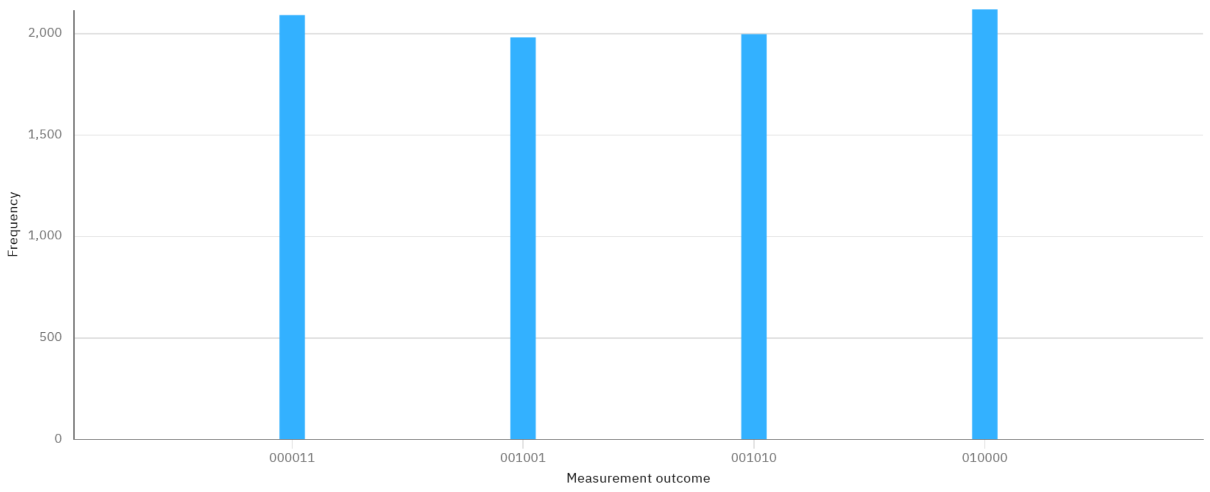


Figure B.8: Results when data qubits 2, 4, 8, 10, 14, 18, 20 and 24 are $|1\rangle$

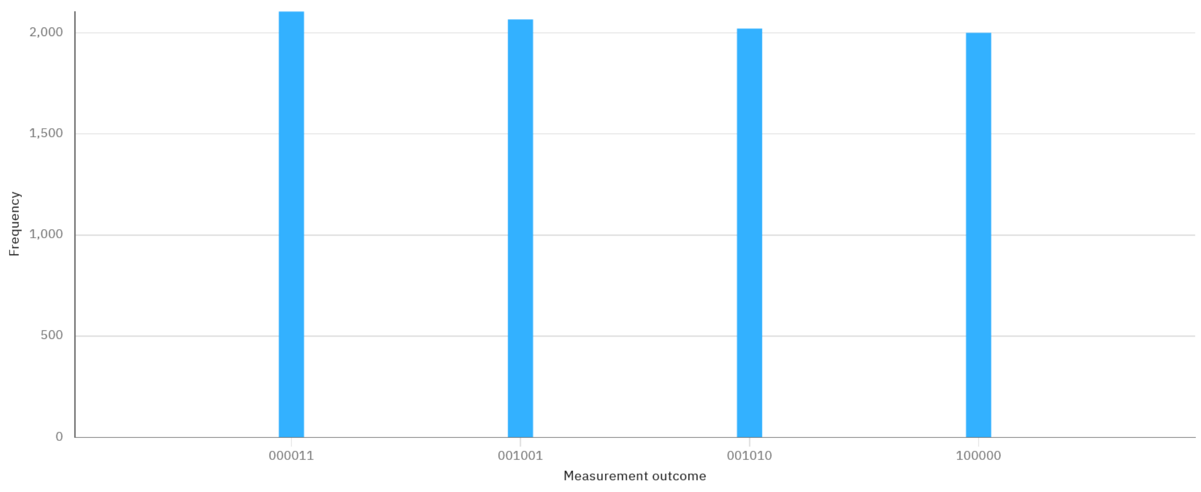


Figure B.9: Results when data qubits 2, 3, 4, 8, 9, 10, 14, 15, 18, 20, 21 and 24 are $|1\rangle$

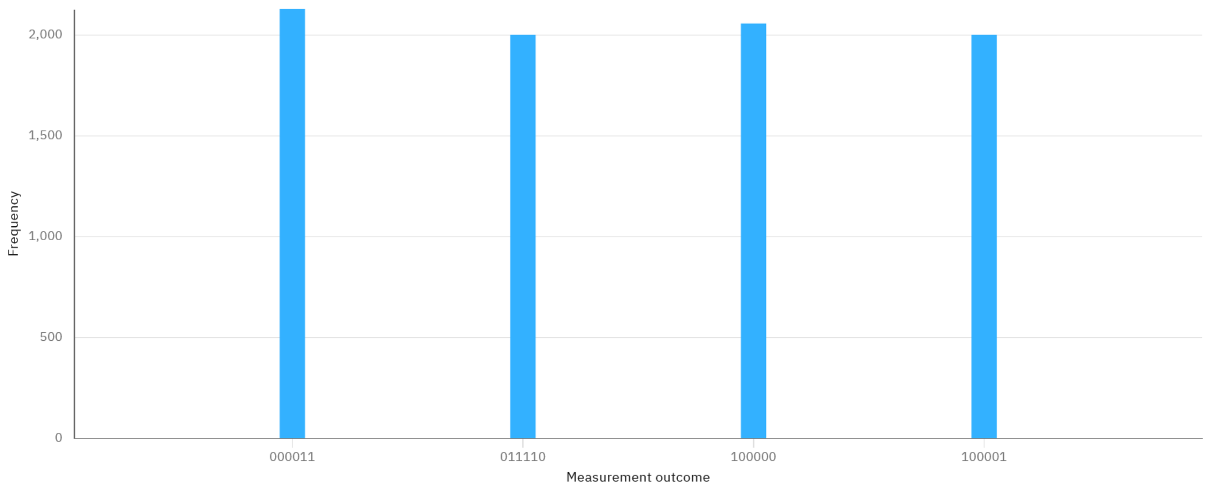


Figure B.10: Results when all data qubits except 25 are $|1\rangle$

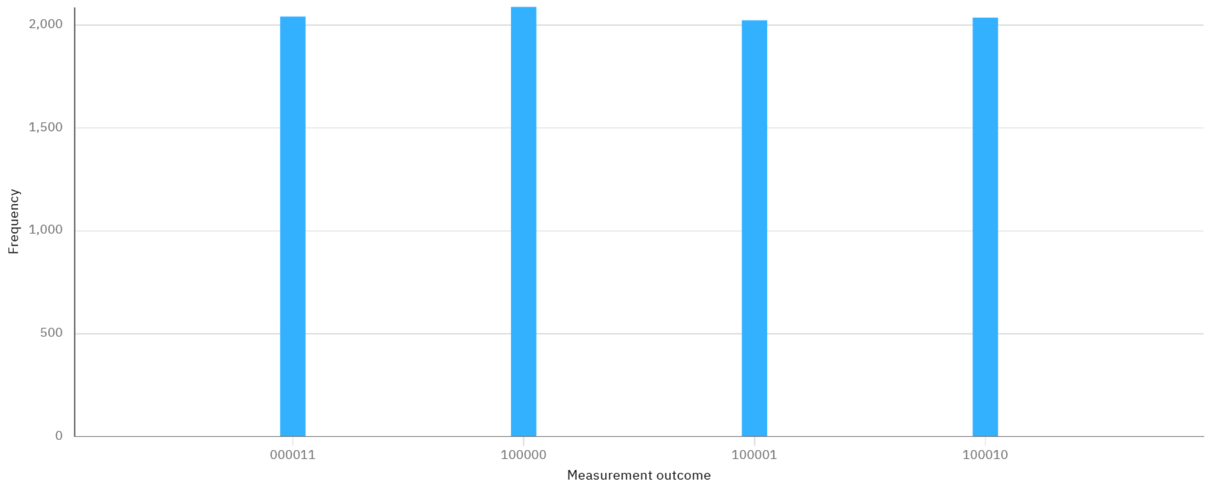


Figure B.11: Results when all data qubits are $|1\rangle$