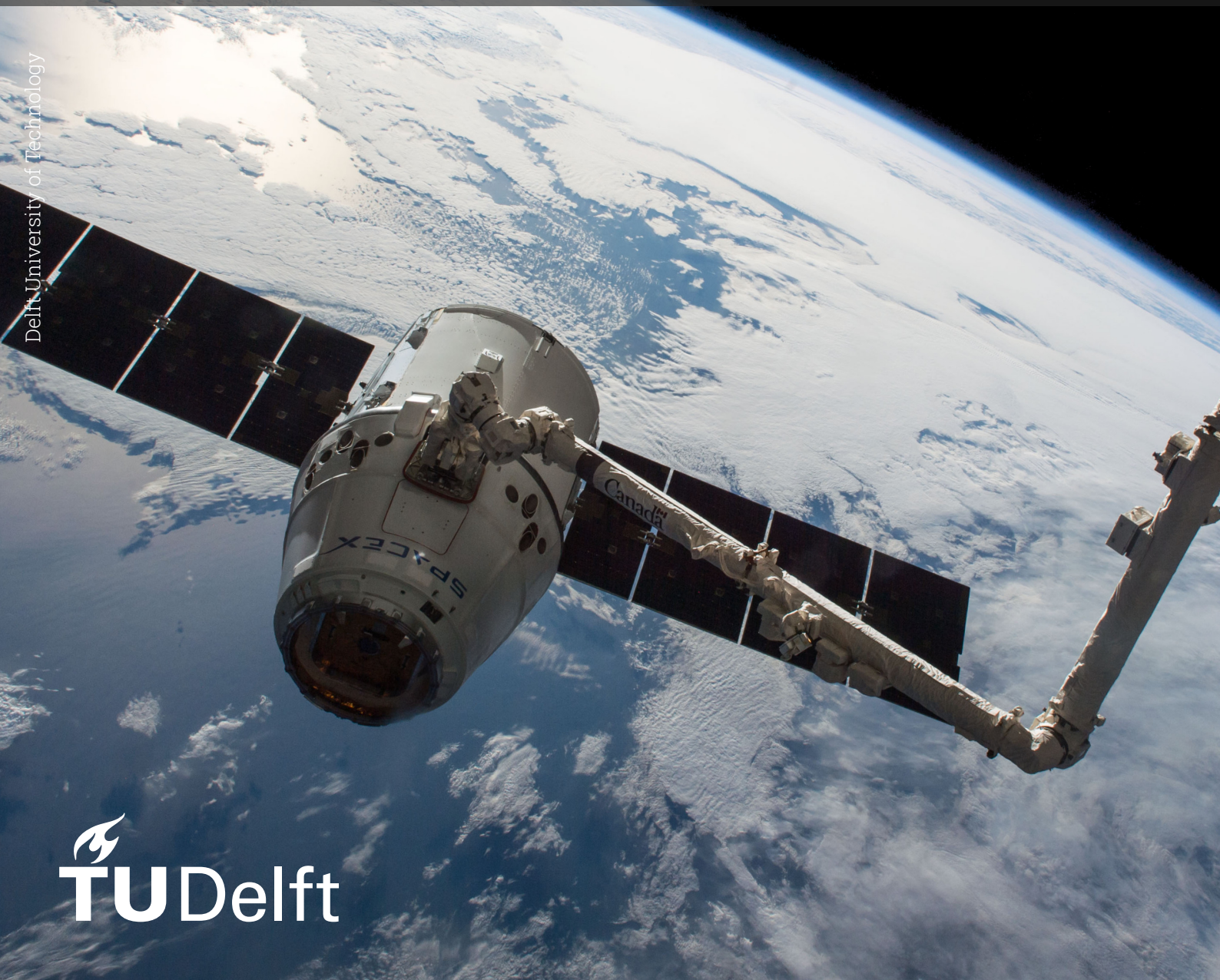
Finding Requirements for Blind Quantum Computing

Towards a Quantum Internet

Applied Physics Bachelor Thesis

Tim Albers



Delft University of Technology

TU Delft

Finding Requirements for Blind Quantum Computing

Towards a Quantum Internet

by

Tim Albers

Student Name	Student Number
Tim Albers	5406056

Responsible thesis Supervisor:	Prof. Dr. S.D.C. Wehner
Daily Supervisor:	J. van Dam
Second Examiner:	Prof. Dr. A.F. Otte
Project Duration:	April 2024 - July 2024
Faculty:	Faculty of Applied Physics, TU Delft

Cover:	Canadarm 2 Robotic Arm Grapples SpaceX Dragon by NASA under CC BY-NC 2.0 (Modified)
Style:	TU Delft Report Style, with modifications by Daan Zwaneveld

TU Delft | QuTech

Abstract

This thesis explores the fundamental requirements necessary for implementing Blind Quantum Computing (BQC) over a quantum internet, focusing on the non-verifiable case. BQC allows clients to perform quantum computations on a remote server without revealing their input, computation, or outputs, thereby ensuring privacy. The main objectives were to identify minimal hardware requirements to achieve a 70% success probability of BQC at a metropolitan distance of 50 km and to examine the effect of varying fibre lengths on the protocol's success. Using the NetSquid simulation environment, this study numerically modeled a scenario where the client and server are separated by a distance of 50 km and the client's quantum capabilities are limited to only making measurements while the server, realized as a trapped-ion quantum server has powerful quantum processing capabilities. The research employed a genetic algorithm to find optimal hardware settings that balance performance against cost, aiming to degrade the hardware parameters from their current state-of-the-art while still achieving the required 70% success threshold. The results demonstrate that while certain parameters such as coherence time and server efficiency have a relatively high tolerance for error, others, like the single-qubit gate fidelity, are more critical and permit less deviation from their state-of-the-art values. This work contributes to the quantum computing field by mapping the critical parameters that influence the practical deployment of BQC, thus paving the way for future advancements in quantum communications.

Contents

1

Introduction

Quantum computing is promised to revolutionize various fields of computer- and information science by leveraging the principles of quantum mechanics to process information in fundamentally new ways. Unlike classical computers, which use bits as the basic unit of information, quantum computers use quantum bits, or qubits, which can exist in multiple states simultaneously (superposition) and can be entangled with each other. These capabilities enable quantum computers to tackle specific problems much faster than classical computers, which makes them suitable for applications such as finding discrete logarithms and factoring integers [1], as well as material and drug discovery [2]. Furthermore, quantum computing introduces new exciting techniques, such as quantum encryption [3].

However, the potential advantages of quantum computers are currently not accessible. Looking ahead, as these advantages become available, it is likely that their usage will primarily involve delegated computation. This is because, despite significant progress in their development, the construction and operation of quantum computers will remain expensive. One solution is to provide remote quantum servers in the cloud, which would reduce the need for users to own these costly machines. However, this approach raises privacy concerns, especially when handling sensitive data. Ideally, users want to utilize the computational power of quantum servers without revealing their input data, computations, or results to the server.

Blind Quantum Computing (BQC) offers a solution to this problem of privacy by enabling clients to execute quantum algorithms on remote servers without revealing their input, computation, or outcomes. This method ensures that only an upper bound on the computation's size is known to the server [4]. Furthermore, for broader accessibility, it is preferable for the client to be as cost-effective as possible. This work focused on a version where the client's quantum capabilities are restricted to only making measurements [5, 6]. This approach was chosen because it has already been successfully implemented in laboratory settings [7].

This report builds upon the work presented in [8], which focuses on 'robust Verifiable Blind Quantum Computing' (rVBQC) protocols, introduced in [9]. In contrast, this study focuses on the non-verifiable case of BQC. While the underlying theory and methods share similarities, this study diverges by omitting verification steps, thereby streamlining the protocol and reducing complexity. This shift allows for the focus on optimizing the non-verifiable BQC process, potentially broadening the practical applications and accessibility of quantum computing.

Furthermore, while demonstrations of BQC and rVBQC have already been conducted in the laboratory [7, 10, 11] practical real-world applications require clients to be separated by some distance from servers, necessitating high-quality quantum computing and communication hardware. This study aims to determine minimal hardware requirements for performing BQC with a minimal success probability of 70% over a metropolitan distance of 50 km using a trapped-ion-based server, such as [12], and a measurement-only client.

To achieve this, the BQC protocol is simulated numerically using NetSquid, a discrete event simulator for quantum networks [13]. This involves modeling the trapped-ion quantum server with the trapped-ions NetSquid library [14] and utilizing a model for the measurement-only client as described in [8], with the simulations being guided by a set of specific hardware parameters.

Initially, the objective of this study was aligned with that of [8], with the primary adaptation focusing on the non-verifiable BQC protocol. Specifically, the goal was to determine the set of hardware parameters that would minimize the necessary improvements over current state-of-the-art parameters to enable the implementation of the BQC protocol with a success probability of at least 70% over a distance of 50 km. However, after conducting initial tests with these state-of-the-art parameters, it became evident that the 70% threshold was already being met according to the simulation results.

Consequently, the focus of this work shifted to two new objectives:

1. Determine the effect of fibre length on the success probability of the BQC protocol. Specifically, identify the maximum fibre length that still allows for a minimum success probability of 70% while maintaining all other hardware parameters at their baseline values, based on current state-of-the-art long-distance trapped-ion experiments. This objective is significant because it addresses a key challenge in quantum communication: maintaining high fidelity over long distances. Quantum states, particularly entangled qubits, are prone to decoherence and loss in optical fibres. Understanding the impact of fibre length on BQC success probability helps identify practical limits and optimal configurations for quantum networks.
2. Identify the set of hardware parameters that can be maximally degraded from the current state-of-the-art values while still achieving a minimum success probability of 70% for the BQC protocol. This will determine the error margins for each hardware parameter when building the BQC setup in practice. This objective is crucial for understanding the robustness and tolerances of the BQC protocol under less-than-ideal conditions. By identifying the extent to which each parameter can be degraded without reducing the protocol's success rate below the threshold, it becomes possible to prioritize the most critical hardware improvements and allocate resources more efficiently. This set of minimal hardware parameters is found by using a genetic algorithm, which combines the requirement for achieving a 70% success probability with the cost of hardware parameters into a single-objective minimization problem, similar to the techniques used in [8, 15, 16]. This approach allows for the systematic exploration of the parameter space and identifies the optimal configuration that balances performance and cost, ensuring the most efficient use of resources while maintaining the desired success rate for the BQC protocol.

This work is organized as follows. Chapter 2 covers the theoretical foundations. Sections 2.1 and 2.2 explain the underlying concepts of (measurement-based) quantum computing and the quantum internet. Section 2.3.1 provides a detailed description of the non-verifiable BQC protocol used, while section 2.4 offers background information on genetic algorithms. Chapter 3 outlines the methods used. Section 3.1 details the simulation setup, including the parameters used, their significance, and the modeling of the server and client within the simulation. Section 3.2 explains the optimization of simulation parameters using a genetic algorithm and how this approach generates the results. Chapter 4 presents and analyzes the findings of this study, discussing the implications of the results and the methodologies employed. Additionally, it offers recommendations for future research.

2

Theory and Background

2.1. Quantum Computing

Quantum computing represents a major change in computation, leveraging the principles of quantum mechanics to process information in fundamentally new ways. Traditional classical computers operate using bits as the basic unit of information, which can be either 0 or 1. In contrast, quantum computers use quantum bits, or *qubits* [17], which can exist simultaneously in multiple states (superposition) and can be entangled with each other. These capabilities enable quantum computers to solve certain problems exponentially faster than classical computers [18], potentially revolutionizing fields that are currently intractable for classical methods. Potential applications include finding discrete logarithms and factoring integers [1], material and drug discovery [2], and various other commercial uses [19].

2.1.1. Qubits and Operators

Quantum computation utilizes *qubits*, the quantum counterparts of classical bits. While a bit has two possible states (0 and 1), a qubit exists in a two-dimensional complex Hilbert space [A.1], which is mathematically equivalent to the two-dimensional complex vector space $\mathbb{C}^2$.

Typically, a preferred basis is chosen, which physically corresponds to two distinguishable states, often referred to as the ”zero” and ”one” state. The basis vectors are, using bra-ket notation [A.1.1], denoted as $|0\rangle$ and $|1\rangle$ respectively, allowing any vector ψ to be expressed as a linear combination of the two: $\alpha|0\rangle + \beta|1\rangle$, where $\alpha, \beta \in \mathbb{C}$. Moreover, the norm of a vector is defined as:

$$||\psi|| := \langle\psi|\psi\rangle^{\frac{1}{2}} = (\alpha^*\alpha + \beta^*\beta)^{\frac{1}{2}} \tag{2.1}$$

Below are some states that are often used in quantum computing:

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}, |1\rangle = \begin{bmatrix}0\\1\end{bmatrix}, |+\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}, |-\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix} \tag{2.2}$$

Note that a single qubit in the state $\alpha|0\rangle + \beta|1\rangle$ can be visualized as a point (θ, ϕ) on the unit sphere, where $\alpha = cos(\theta/2)$ and $\beta = e^{i\phi}sin(\theta/2)$. This is called the Bloch sphere representation, and the vector $(cos(\phi)sin(\theta), sin(\phi)sin(\theta), cos(\theta))$ is called the Bloch vector [20].

The Bell states are a set of four specific maximally entangled states of two qubits, defined as follows:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{2.3}$$

$$|\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \tag{2.4}$$

$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \tag{2.5}$$

$$|\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \tag{2.6}$$

Their entanglement means that one qubit, in a superposition of 0 and 1, can be measured in the standard basis to yield either 0 or 1 with equal probability. When the second qubit is measured, its outcome will either match the first qubit's outcome (for the Φ Bell states) or be the opposite (for the Ψ Bell states). Thus, the individual outcomes appear random but are perfectly correlated. These correlated results, regardless of the indeterminate nature of measuring a single qubit, demonstrate the entangled nature of the qubits.

Unitary operators are crucial in quantum computing because they preserve the inner product and the norm, allowing quantum states to evolve in a stable and reversible manner. Among the unitary operators, the *Pauli operators* are particularly significant. They are fundamental in describing quantum state transformations and rotations, and they play a key role in error correction and quantum circuit design. The Pauli operators are represented by the following matrices:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.7}$$

These matrices are of great importance in quantum mechanics because of the fact that they can describe rotation. Therefore these matrices are usually called the "Pauli spin matrices".

2.1.2. Measurement Based Quantum Computing

Measurement-based quantum computation (MBQC) is a method for quantum computing that relies exclusively on quantum measurements to drive the computational process. This approach contrasts with the traditional circuit model of quantum computation, which primarily involves the application of unitary operations to qubits. In 2001, Raussendorf and Briegel [21, 22] introduced the concept of the one-way quantum computer, where computations are performed using only single-qubit measurements and a specific multi-party entangled state known as the cluster state. The term "one-way" emphasizes that the computation is driven by these irreversible measurements.The process begins with preparing a collection of qubits in a standard entangled state. Measurements are then performed on individual qubits, with the outcomes potentially influencing subsequent measurements. Based on these outcomes, local unitary operators, referred to as corrections, are applied to certain qubits to eliminate the indeterminacy introduced by the measurements.

Danos, Kashefi and Panangaden developed the measurement calculus for MBQC [23]. In this section, the notation and essential steps for implementing Measurement-Based Quantum Computing (MBQC) will be reviewed, as the Blind Quantum Computing protocol discussed in this work is built upon this computational method.

Commands

First, a notation for computations based on 1-qubit measurements is established. The fundamental commands available for use are:

command	notation
1-qubit auxilary preparation	N_i
2-qubit entanglement operators	E_{ij}
1-qubit measurements	M_i^α
1-qubit Pauli operators corrections	X_i and Z_i

where i and j represent the qubits on which each of these operations apply and $\alpha \in [0, 2\pi]$. Sequences of these commands are called *patterns*, more on this later. It is important to note that corrections and measurements are allowed to depend on previous measurement outcomes.

By using the N_i command, the qubit i is prepared in the $|+\rangle_i$ state. The entanglement operation is by definition $E_{ij} := \wedge Z_{ij}$, where $\wedge Z_{ij}$ denotes the controlled-Z operator [A.1.2]. The commands for

the correction operators are the Pauli operators X_i and Z_i. And finally, the measurement command is defined by orthogonal projections on $|+_\alpha\rangle := \frac{1}{\sqrt{2}}(|0\rangle + e^{i\alpha}|1\rangle)$ and $|-_\alpha\rangle := \frac{1}{\sqrt{2}}(|0\rangle - e^{i\alpha}|1\rangle)$ followed by a trace-out operator. Note that projective measurements are probabilistic and irreversible. Here, $\alpha \in [0, 2\pi]$ is the *angle* of the measurement. Furthermore, the outcome of a measurement on qubit i will be denoted as s_i. A *signal* s is a summation of measurement outcomes.

Dependent x- and z-corrections and measurements will be represented as follows, respectively: X_i^s, Z_i^s and ${}^t[M_i^\alpha]^s$, where s ,$t \in \mathbb{Z}_2$ and $\alpha \in [0, 2\pi]$. In the case of dependent measurements, the measurement angle will depend on s, t and α as

$$ {}^t[M_i^\alpha]^s := M_i^{(-1)^s\alpha + t\pi} \tag{2.8}$$

An important part of the work by Danos, Kashefi and Panangaden [23] is the propagation of dependent corrections through measurement. This allows one to push corrections through measurements acting on the same qubit:

$$E_{ij}X_i^s = X_i^s Z_j^s E_{ij} \tag{2.9}$$

$${}^t[M_i^\alpha]^s X_i^r = {}^t[M_i^\alpha]^{s+r} \tag{2.10}$$

$${}^t[M_i^\alpha]^s Z_i^r = {}^{t+r}[M_i^\alpha]^s \tag{2.11}$$

This demonstrates that corrections can be commuted, allowing the measurement angles to be adjusted so that corrections do not need to be applied between measurements.

Patterns

By defenition, a *pattern* [23] consists of three finite sets (V, I, O), where V is the pattern *computation space* and I and O are respecticely the pattern *inputs* and *outputs*. Furthermore, a pattern also consists of a finite sequence of commands $(A_n, A_{n-1}, ..., A_1)$ that apply to a qubit in V form right to left, that is, A_1 first and A_n last. As an example, [23] shows that the Hadamard-gate can be constructed as such:

$$H = (\{1, 2\}, \{1\}, \{2\}, X_2^{s_1} M_1^0 E_{12} N_2) \tag{2.12}$$

In this equation, the computation space is $\{1, 2\}$, the imputs and outputs are respectively $\{1\}$ and $\{2\}$, and the command sequence $X_2^{s_1} M_1^0 E_{12} N_2$ describes in order: N_2 prepares the first qubit in some state ψ, and the second qubit in the state $|+\rangle$. E_{12} entangles these states using the controlled-z operator to obtain $\wedge Z_{12}(\psi_1 \otimes |+\rangle_2)$. Then, M_1^0 measures the first qubit in the $|+\rangle$, $|-\rangle$ basis to obtain s_1. If s_1 equals 1, then an additional X correction is done on the second qubit.

A crucial aspect of the work by Danos, Kashefi, and Panangaden [23] is the concept of standardization. The Standardization Theorem, a pivotal result in the MBQC framework, offers a systematic approach to transform any MBQC pattern into a standardized form using a set of rewrite rules defined in their paper. This transformation simplifies the analysis and implementation of quantum computations by structuring the process into a specific order of operations:

1. Preparation phase: All qubits are initialized in their respective states.
2. Entanglement phase: All entanglement operations (typically represented by controlled-Z operations) are performed.
3. Measurement phase: All measurements on the qubits are executed.
4. Correction phase: Classical corrections based on the measurement outcomes are applied.

The Standardization Theorem breaks down quantum computation patterns into clear phases, making it easier to understand and optimize quantum algorithms. The theorem also enables parallel execution of independent measurements by ensuring that all entanglements happen at the start and corrections at the end. This is particularly relevant in this work as it provides a structured framework for implementing and analyzing the non-verifiable Blind Quantum Computing protocol, ensuring efficient and reliable performance. Additionally, by commuting the corrections, measurement angles are adjusted, eliminating the need for intermediate corrections between measurements, thereby further enhancing the protocol's efficiency.

2.2. Quantum Internet

A quantum network consists of connected quantum resources that facilitate the transmission and manipulation of quantum information. By scaling up this concept, one achieves a quantum internet; a global network of interconnected quantum devices. The quantum internet is envisioned as a global quantum network composed of three critical hardware components. First, quantum channels like telecom fibres are required for qubit transmission. Second, due to the inherent lossiness of quantum channels, quantum repeaters (and potentially satellites) are necessary to extend transmission distances. These repeaters are placed at intervals along optical fibres, enabling long-distance qubit transmission. Finally, end nodes, which are quantum processors, connect to the network. These nodes can range from simple qubit handlers to large-scale quantum computers. A quantum internet is designed to complement, not replace, classical communication. Therefore, it is assumed that all nodes are capable of classical communication, such as via the classical internet, to exchange control information.

2.2.1. Secure Communication

A quantum internet has several useful applications. One of these is Quantum Key Distribution (QKD). QKD is a method that leverages the properties of quantum mechanics, such as the No Cloning Theorem [24], to allow two people to securely agree on a key. In this context, a key is a secret codeword that is shared only between you and the person with whom you are trying to communicate. This secret codeword can then be used to encrypt messages such that they can be transmitted without being read by a malicious third party. The most well-known QKD protocol is BB84 [25], which enables secure key distribution and is resistant to computational attacks, ensuring long-term security of the communication channel.

Significant milestones have already been achieved, such as the successful demonstration of QKD over practical distances [26, 27, 28, 29] and several companies are already making QKD commercially available [30, 31]. Another sophisticated application of the quantum internet which may be commercially available in the future is Blind Quantum Computing.

2.3. Blind Quantum Computing

Blind Quantum Computing (BQC) enables secure, cloud-based quantum computation where a client can delegate quantum computations to a remote quantum server without revealing the input, the computation, or the output to the server. This is achieved through protocols that leverage the principles of quantum mechanics to ensure that the server cannot access the sensitive data being processed. This makes BQC particularly significant because it addresses privacy concerns in quantum cloud computing, allowing users to perform complex quantum computations while maintaining privacy. This section delves into the BQC protocol in detail, exploring how it ensures secure and private quantum computations over the quantum internet.

2.3.1. BQC Protocol

The protocol presented here builds on those detailed in references [6, 7, 8], incorporating a variation where the client uses measurements for remote state preparation (RSP) on the server. In RSP, the sender measures part of an entangled state and sends a classical correction to establish a target state at the receiver. Unlike the previously referenced works, this study focuses on non-verifiable Blind Quantum Computing [32].

The protocol involves two parties: a server and a client. The server is a powerful quantum processor located some distance away, implemented as an Ion-Trap quantum processor based on the theory provided in [12]. The client, conversely, is a smaller quantum device capable only of performing measurements in arbitrary superposition bases.

Remote State Preparation phase

The protocol begins with the client sending a graph-state [A.1.4] description to the server over the classical channel. This description includes I (the number of qubits) and G (the graph's edges). Using this description, the server prepares a Bell pair for each qubit:

$$|\psi\rangle = |\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \tag{2.13}$$

and sends half of this state, namely a photonic qubit, to the client using the quantum channel. The server also transmits a unique qubit ID over the classical channel. Upon confirming the qubit's arrival, the client remotely prepares it in the $|+_\theta\rangle$ state, by measuring along the $|\pm_\theta\rangle$-basis as follows:

Upon receiving qubit i, the client selects a random integer value for k_i, which ranges from 0 to 8, and determines the measurement angle θ_i:

$$\theta_i = \frac{k_i \pi}{4} \tag{2.14}$$

The measurement outcome of qubit i, measured along $\pm_{\theta_i}$, is stored and denoted as $m_i \in \{0, 1\}$. This measurement of the client projects the state of the ion onto the $\pm_{\theta_i}$ basis ($+$ if $m_i = 0$ and $-$ if $m_i = 1$).

If a qubit is lost during transmission through the quantum channel, or if the qubit in the memory of the server has existed for too long and is therefore discarded to limit the decoherence from affecting the computation, the server discards its entangled partner and tries again. This process continues until all qubits for the graph have been remotely prepared. The server then notifies the client that the RSP phase is complete. This must occur within a cutoff time based on the coherence time of a single qubit, as qubits cannot remain indefinitely in memory. This challenge will be discussed in later chapters.

Graph Sate formation phase

Following the RSP phase, the server prepares a graph state $|G\rangle$ based on the client's description. Preparing a graph state from initialized qubits in the server memory involves applying $\wedge Z_{ij}$ gates to the edges ($e \in E$) defined by G. More details on graph states are stated in section A.1.4.

Measurement and Decoding phase

After forming the graph state, the client instructs the server to measure in the $\pm_{\delta_i}$ basis, with

$$\delta_i = \phi'_i + \theta_i + m_i\pi + r_i\pi \tag{2.15}$$

where $r_i \in_R \{0, 1\}$ is a randomly chosen bit. Furthermore, the angle ϕ'_i is defined as:

$$\phi'_i = (-1)^{s_{X,i}}\phi_v + s_{Z,v}\pi \tag{2.16}$$

More information about ϕ_i, s_{X_i} and s_{Z_i} can be found in [23]. The server measures qubit i in the $\pm_{\delta_i}$ basis to obtain the measurement outcome $b_i \in \{0, 1\}$, whereafter it sends this outcome to the client over the classical channel. The client then decodes this outcome:

$$s_i = mod(b_i, r_i). \tag{2.17}$$

where mod denotes the modulo operator, b_i the measurement outcome of the server and r_i the randomly chosen bit. The value for s_i is used to calculate ϕ'_{i+1} for the next qubit measurement. This process repeats until the final outcome s is obtained, representing the protocol's output.

2.4. Genetic Algorithms

This section explores the theory and nomenclature behind genetic algorithms, which are employed later in the project to solve an optimization problem.

The genetic algorithm is a metaheuristic optimization technique inspired by natural selection, the driving force behind biological evolution. It iteratively improves a population of potential solutions by selecting individuals from the current population to act as parents that are then used to generate a new generation of offspring. Over many iterations, the population evolves towards an optimal solution. But how does the algorithm determine which solution is optimal? This is where the cost function comes into play.

2.4.1. Cost Function

The cost function is a key part of the genetic algorithm. The main goal of the genetic algorithm is to minimize this cost function. A good cost function will have a lower value when the solution is better. The algorithm's designer needs to create a cost function that accurately represents the problem's requirements and goals, making sure it gives lower values to better solutions. Different parameters can be provided as input to the cost function, allowing it to evaluate various aspects of potential solutions. This setup helps the genetic algorithm effectively search through possible solutions to find the best one.

2.4.2. Population, Generations, Parents, Children, and Mutation

The genetic algorithm evolves an initial population of potential solutions to an optimization problem, where each individual in the population represents a possible solution, in this case, a set of five hardware parameters. The population size affects performance, with larger populations offering more diversity but requiring more computational resources.

The process takes place over multiple iterations known as generations. Each generation involves selection, crossover, and mutation to produce new individuals. Selection chooses parents based on their costs. Individuals with a lower cost have a higher chance of being selected. Moreover, selected individuals will become parents. Parents produce offspring, known as children, through crossover and mutation. Crossover combines genetic information from two parents, creating children with characteristics from both. Mutation introduces random changes to maintain diversity and prevent premature convergence to suboptimal solutions. In addition, by maintaining a diverse population and broadly exploring the search space, the genetic algorithm enhances its ability to find global optima. However, the global optimum is not guaranteed, as the algorithm can still get trapped in a local optimum.

Through successive generations, the population evolves as these processes iteratively refine the solutions, guiding the population toward optimal or near-optimal solutions. Inspired by natural selection, this approach allows the genetic algorithm to tackle complex optimization problems, such as the one in this report.

3

Setup and Methods

3.1. Simulation Methods

The protocol outlined in Section 2.3.1 is presented in a general form. To accurately simulate the challenges faced in real-world scenarios, it is essential to incorporate factors such as losses and noise into the simulation. In this section we will go over the setup that is simulated and how it was simulated to find the success probability of the protocol.

3.1.1. Simulation Inputs and Outputs

Computational Bases, Succes Probability and Confidence Interval

In this work, a simple graph state is simulated where $I = \{1, 2\}$ and $G = \{(1, 2)\}$. This means that the graph state consists of 2 qubits, q_1 and q_2, interconnected by one edge. Furthermore, the computational bases are chosen to be $\phi_1 = \pi/2$ and $\phi_2 = -\pi/2$, corresponding to the X basis: $|+\rangle$ and $|-\rangle$. In a noiseless scenario of the run of the protocol described in 2.3.1 the outcome of the computation is deterministically 1, therefore one can find the success probability by finding the fraction of the computation rounds that deviate from this outcome. This success probability is the principal output under investigation. Moreover, the confidence interval of the success probability is defined by Hoeffding's bound [33], where the $\alpha\%$ confidence interval is defined by

$$\epsilon_\alpha = \sqrt{\frac{ln(2/(1-(\alpha/100)))}{2n}} \tag{3.1}$$

with n the number of runs.

Hardware Parameters

Whether or not the protocol succeeds, depends on various hardware parameters given as input to the simulation. These parameters are divided into two subsets. Subset 1 is a set of parameters that are specific to the setup or do not significantly affect the success probability (e.g. gate durations), and are therefore not varied over during optimization. These parameters are listed in Table 3.1 as in [8].

Subset 2 consists of five parameters, listed in Table 3.2. These are the optimization parameters. Using the optimization methods described in section 3.2, these parameters are, in contrast to subset 1, varied over. Furthermore, in the second column of Table 3.2 the 'baseline values' are listed. These baseline values are taken from the current state-of-the-art long-distance trapped-ion experiments and will serve as the reference point from where the optimization will conduct its search.

Parameter	Value
Channel length	$50\ km$
Photon loss probability in fibre	$0.2\ dB/km$
Waveplate errors (fast axis tilt/retardation deviation)	0.001
Dark count probability of photon detectors	0.02%
Crosstalk in polarising beam splitter	0.0001
Qubit rotation duration	$12\ \mu s$ [34]
Entangling gate duration	$107\ \mu s$ [34]
Ion initialisation duration	$300\ ns$ [35]
Photon emission duration	$300\ ns$ [35]
Ion qubit readout duration	$100\ \mu s$

Table 3.1: Parameter set 1, as in [8]: These are the parameters that are specific to the setup or do not significantly affect the success probability of the protocol.

Parameter	Baseline Value
Initial loss probability (= 1 - emit $\times$ freq. conversion)	$0.8675\ (=\ 1-(0.53$ [12] $\times\ 0.25$ [36]$))$
Single-qubit gate fidelity	0.99 [37]
Entangling gate fidelity	0.95 [38]
Emission fidelity	0.947 [39]
Coherence time (ms)	62 [34]

Table 3.2: Parameter set 2 as in [8]: The baseline for long-distance trapped-ion experiments aligns with the current state of the art, as referenced in the table. These parameters are varied over during the optimization discussed in section 3.2. Server efficiency encompasses the overall efficiency of preparing an ion-qubit, emitting an ion-entangled photon, coupling to the fibre (represented as 'emit' in the table), and converting its frequency to the telecom C band at 1550 nm (indicated as 'freq. conversion' in the table). The emission fidelity denotes the fidelity of the ion-photon entangled pair at the moment of photon emission.

3.1.2. Server Modelling

The server is implemented as an "ion-trap quantum processor capable of emitting entangled photons" [14], following the model in [12]. This setup means that the qubits sent from the server to the client are photonic qubits and the bases of these photonic qubits are in the polarization basis. In the simulation, the trapped-ion server is modeled using the NetSquid-TrappedIons library [14]. Equivalent to [8] and [16], the decoherence of n trapped-ion qubits over time is modeled through a collective Gaussian dephasing process:

$$\rho \to \int_{-\infty}^{\infty} K_r \rho K_r^{\dagger} p(r) dr, \tag{3.2}$$

where

$$K_r = exp\left(-ir\frac{t}{\tau}\sum_{j=1}^{n} Z_j\right). \tag{3.3}$$

Here Z_j represents the Pauli Z operator [A.1.2] acting on qubit j. Furthermore, τ is the coherence time and t the storage time (i.e. the time spent in memory), and

$$p(r) = \frac{1}{\sqrt{2\pi}} e^{-r^2/2}. \tag{3.4}$$

Reference [8] states that this can be interpreted as follows: "all the qubits undergo Z rotations at a constant rate of $2r$ per time interval τ, where r is a random variable with probability distribution $p(r)$". More information on the dephasing process can be found in [8, 14, 16].

During the RSP phase described in section 2.3.1, some qubits are prepared and stored in the server's memory while waiting for the RSP phase to complete. This phase may take a while because multiple attempts are often needed to remotely prepare a single qubit, mainly due to photon loss in the fibre and the inefficiency of the server. As described above, the qubits in memory of the server decohere over time, resulting in a lower success probability. To maximize the success probability, the goal is to limit the time qubits spend in memory by implementing a cutoff time. Selecting an appropriate cutoff

time involves balancing the amount of RSP attempts and fidelity: a shorter cutoff time results in higher-quality qubits due to less decoherence but increases the number of discarded qubits, thereby increasing the number of RSP attempts consequently lowering the rate of successful RSP attempts. This work, similar to [8], focuses exclusively on minimizing error probability, which is dependent on the fidelity of the remotely prepared qubits, without considering the rate. Consequently, the cutoff time was set to half of the coherence time.

3.1.3. Client Modelling

A schematic overview of the general BQC setup is depicted in Figure 3.1. The figure illustrates a trapped-ion quantum server connected to a client via a quantum channel. The client uses half (H) and quarter (Q) waveplates to manipulate the polarization of incoming photonic qubits and measures them using a polarizing beam splitter (PBS) and single photon detectors (SPDs). The classical channel enables coordination between the client and the server. Additionally, the classical box (CB) on the client side is responsible for selecting a graph description, adjusting measurement angles, handling classical communication, and verifying the results sent by the server. The client setup is shown in detail, while the server setup is kept more general.

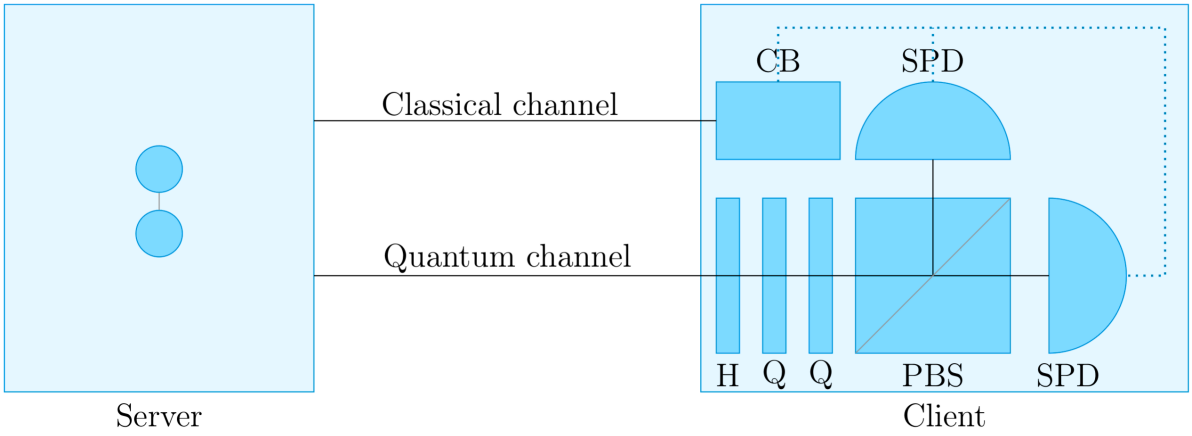


Figure 3.1: This figure is taken from [8] and edited in such a way that it describes the model discussed in this report, where a 2-qubit linear graph state is used. Overview of the setup: a trapped-ion quantum server, represented with a 2-qubit linear graph state, is connected to a client that manipulates the polarization of incoming photons through the quantum channel using half (H) and quarter (Q) waveplates. The client then measures the qubit with a polarizing beam splitter (PBS) and single photon detectors (SPDs). The classical channel facilitates coordination between the client and the server. The classical box (CB) on the client's side is used for communicating the graph description, measurement angle adjustments, handling classical communication, and verifying the results sent by the server. The client setup is depicted in more detail to illustrate its modeling in the simulation, while the server's exact configuration is kept general.

The quantum capabilities of the client are in this case restricted to solely making measurements. These measurements are performed by letting the photonic qubit pass through three waveplates (see Figure 3.1), which implement specific polarization rotations whereafter the qubit can be measured in a chosen basis. The three waveplates are modeled in NetSquid using Jones matrices. Waveplates induce a relative phase retardation between the fast axis and the slow axis. This phase retardation δ equals $\pi/2$ and π for a quarter wave plate (QWP) and a half wave plate (HWP) respectively. Furthermore, the waveplates' fast axes are angled at ϕ radians with respect to the x-axis, which lies in the plane of polarization for linearly polarized light. This angle specifies the precise rotation that is carried out. Naturally, these angles come with some form of error. As stated in Table 3.1, the retardation deviation $\Delta\delta$ and the fast axis tilt $\Delta\phi$ contribute to the waveplate errors. Using these expressions for the error, the Jones matrix [40] can be written as

$$J(\delta', \phi') = \begin{pmatrix} e^{\frac{i\delta'}{2}} \cos^2(\phi') + e^{-\frac{i\delta'}{2}} \sin^2(\phi') & \left(e^{\frac{i\delta'}{2}} - e^{-\frac{i\delta'}{2}}\right) \cos(x_p) \sin(\phi') \\ \left(e^{\frac{i\delta'}{2}} - e^{-\frac{i\delta'}{2}}\right) \cos(\phi') \sin(\phi') & e^{\frac{i\delta'}{2}} \sin^2(\phi') + e^{-\frac{i\delta'}{2}} \cos^2(\phi') \end{pmatrix} \tag{3.5}$$

with $\delta' = \delta + \Delta\delta$ and $\phi' = \phi + \Delta\phi$. In [41] it is demonstrated that any element of SU(2) can be achieved using two QWPs and one HWP in the configuration H-Q-Q. Using the coefficients $a = -\pi/2$ and $b = \theta_i$,

the fast axes of these waveplates can be defined as:

$$\phi_1 = \frac{\pi}{4} + \frac{a}{2} = 0 \tag{3.6}$$

$$\phi_2 = \frac{\pi}{4} + \frac{a+b}{2} = \frac{\theta_i}{2} \tag{3.7}$$

$$\phi_3 = -\frac{\pi}{4} + \frac{a+b}{4} = \frac{\theta_i}{4} - \frac{3\pi}{8} \tag{3.8}$$

This arrangement allows for measurements in the $|\pm_{\theta_i}\rangle$-basis, where θ_i is randomly chosen by the client, as described in equation (2.14). The measurement process involves using a Polarizing Beam Splitter (PBS) and two Single Photon Detectors (SPDs). Additionally, the SPDs are subject to errors, specifically dark counts. The dark count probability is calculated as $p_{dc} = 1 - e^{-R_{dc}\tau}$, where $R_{dc} = 1500\ Hz$ is the dark count rate, and $\tau = 12.5\ ns$ is the detection time window. These values are sourced from [7], as also referenced in [8].

3.1.4. Channel Length

To answer the first research question, the simulation must be run multiple times for various channel length values while keeping the other parameters constant, as specified in Tables 3.1 and 3.2. This ensures that any observed changes in success probability are solely due to variations in the channel length. The simulation script is run for $70,000$ runs for each channel length value to ensure a small confidence interval of ± 0.0051, as specified in equation (3.1). By monitoring the success rate for each run, data can be systematically gathered on how the channel length affects the protocol's performance. After collecting the simulation data, the results will be analyzed to determine the specific channel length at which the success probability reaches or falls below 0.7. This threshold is significant as it represents a practical benchmark for the feasibility of quantum communication over different distances.

3.2. Optimization Methods

3.2.1. Cost Function Design

The goal of the optimization is to identify a set of hardware parameters that, despite being significantly degraded from their baseline values, still achieve a success probability of at least 0.7. To facilitate this, a cost function has been designed to have its minimal value when the success probability p_{suc} is at least 0.7 and all parameters are maximally degraded from their baseline values. This section presents the cost function that was used to solve the problem at hand and motivates how each part of this cost function contributes to its objective.

The cost function is defined as:

$$C = a + w_1 u(0.7 - p_{suc}) + w_2 Hc \tag{3.9}$$

In this cost function, C represents the total cost associated with a specific combination of hardware parameters. The term a is a constant positive prefactor that ensures the cost remains non-negative, which can occur if rewards (i.e., negative costs) are implemented in the cost function. In the absence of rewards, the value of a does not affect the outcome since the theoretical minimum value of C will be equal to a. Therefore, $a = 0$ was chosen (some recommendations for potential rewards are discussed at the end of chapter 4). The weights w_1 and w_2 are used to specify the importance of different objectives within the cost function. There are two main objectives in this formulation. The first objective is to ensure that the success rate must be at least 0.7. This constraint is implemented in the cost function by multiplying the weight w_1 by the Heaviside step function $u(x)$, which is defined as:

$$u(x) := \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases} \tag{3.10}$$

The Heaviside step function $u(x)$ ensures that a penalty of w_1 is applied to the cost function whenever the success probability is less than 0.7. Given the importance of this constraint, the value of w_1 is chosen to be relatively high, specifically $w_1 \gg w_2$, with $w_1 = 10^5$ and $w_2 = 10$.

Parameter	Simulated Parameter	p_{NI}
Server efficiency η	Initial loss probability $p = 1 - \eta$	$1 - p$
Single-qubit gate fidelity F	single-qubit depolarization probability $p = 2(1 - F)$	$1 - p$
Entangling gate fidelity F	MS gate depolarization probability $p = 2(1 - F)$	$1 - p$
Emission fidelity F	Emission fidelity F	$\frac{1}{3}(4F - 1)$
Coherence time τ_c	Coherence time τ_c	$\exp(-(10^5/\tau_c)^2)$

Table 3.3: Optimization parameters along with their simulated parameters and probabilities of no imperfection. From top to bottom: The Server efficiency is translated into an initial loss probability and is defined as the initial probability of losing a photon in the quantum channel once it enters a channel. e.g. due to frequency conversion. The single-qubit gate fidelity is translated into the single-qubit depolarizing probability, a parameter characterizing depolarizing channel applied to a qubit when a single-qubit X rotation is performed. The entangling gate fidelity is translated into an MS gate depolarization probability, a parameter characterizing a depolarizing channel that is applied to all qubits participating in a multi-qubit MS gate [42], which can entangle qubits. The emission fidelity and coherence time parameters remain the same in the simulation.

The second objective is represented by Hc, which accounts for the hardware costs. This expression quantifies the deviation of each hardware parameter with respect to the baseline values defined in Table 3.2. The total hardware cost Hc of a parameter set $X = \{x_i\}_{0<i\leq N}$ is defined as a sum of the improvement factors k_i of all parameters and can be expressed as

$$Hc(X) = \sum_{i=1}^{N} k_i, \tag{3.11}$$

where the improvement factor k_i is defined as

$$k_i = \frac{ln(p_{NI}(b_i))}{ln(p_{NI}(x_i))}. \tag{3.12}$$

The improvement factor k_i depends on the ratio of the logarithms of the probabilities of no imperfection $p_{NI} \in (0, 1)$. The probability of no imperfection measures the likelihood that a system or process operates without any deviations from the theoretical optimal performance. This approach scales the parameters from zero to one, enabling the extraction of a dimensionless and thus comparable improvement factor k for each parameter. Table 3.3 lists all optimization parameters along with their corresponding simulated parameters, which represent the optimization parameters within the simulation, and their corresponding probabilities of no imperfection. Equation (3.12) showcases that when a parameter value equals the baseline value (i.e. $x_i = b_i$), the improvement factor for that parameter equals $k_i = 1$. Moreover, a parameter value that is 'worse' (i.e. results in a larger error) compared to the baseline value will correspond to a smaller value for p_{NI}, causing the improvement factor to decrease ($0 < k_i < 1$). In summary, when the improvement factor k of a parameter x with respect to the baseline b is between 0 and 1, the parameter x is considered 'worse' compared to the baseline b. A worse parameter value x induces a lower improvement factor k, leading to a lower hardware cost Hc and, consequently, a lower total cost C. This low value for the hardware cost along with a minimal value of 0.7 for the success probability is the objective that the genetic algorithm aims to achieve.

3.2.2. Parameter Space and Final Configuration

The search for the optimal minimal parameters is conducted in two phases: a search phase followed by a verification phase. Initially, the search phase explores the parameter space with a slightly broader confidence interval to find the location of the optimal solution. Subsequently, the verification phase checks whether or not the solution satisfies the 70% threshold while using a smaller confidence interval. This strategy was adopted to speed up computation times, which was a major challenge in this project and will be discussed later in this section. The parameter space is a five-dimensional space defined by the ranges of the five optimization parameters. Each parameter's range is bounded on one end by its baseline value, as listed in Table 3.2, and on the other end by its absolute minimal requirement. This requirement is determined by holding all parameters except the one under investigation at their baseline values and varying the parameter in question until the success probability falls below the 70% threshold, as illustrated in Figure 3.2.

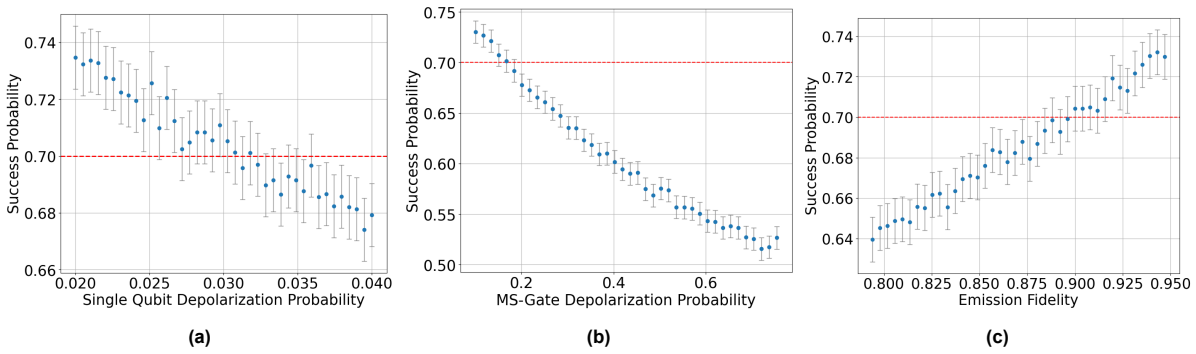


Figure 3.2: Absolute minimal requirements. Plots were created by varying only the parameter under investigation (x-axis) while keeping all other parameters constant at their baseline values. Absolute minimal requirements are determined by identifying the point where the success probability reaches 0.7, as indicated by the horizontal red line. It is important to note that the precise value of the absolute minimal requirement is not critical, as the genetic algorithm's selective nature will effectively manage minor variations. Consequently, an approximate value slightly below the exact threshold suffices, as it primarily aims to reduce the search space. (a) Single Qubit Depolarization Probability: the absolute minimal requirement is approximately 0.035. This is used as the upper bound for this parameter. (b) MS-Gate Depolarization Probability: the absolute minimal requirement is approximately 0.2. This is used as the upper bound for this parameter. (c) Emission Fidelity: the absolute minimal requirement is approximately 0.88. This is used as the lower bound for this parameter. These values are all listed in Table 3.4.

The lower bounds for the coherence time and initial loss probability are determined using a different method due to their significant impact on computation time, as illustrated in Figures 3.3 and 3.4. Figure 3.3a shows that a decrease in coherence time does not heavily influence the success probability. However, the computation time is significantly affected by this decrease, as seen in Figure 3.3b. Here, the average number of attempts to send a photonic qubit from the server to the client increases rapidly with decreasing coherence time.

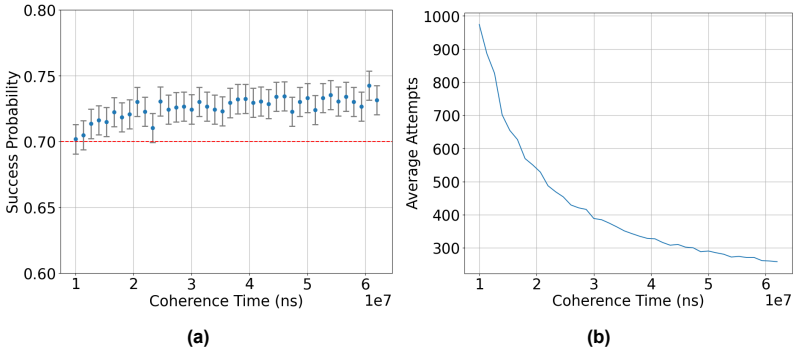


Figure 3.3: (a) The plot was generated by maintaining all parameters at their baseline values and varying only the coherence time. Each data point represents the result of $15,000$ runs, providing a confidence interval of ± 0.011. It shows that coherence time has minimal impact on success probability, with the absolute minimal requirement around the $10ms$ mark. (b) Conversely, decreasing coherence time significantly affects the average number of attempts. These attempts were tracked during the simulation and averaged over $15,000$ runs for each data point.

As described in section 2.3.1, if a qubit is lost during transmission through the quantum channel or if the qubit in the server's memory decoheres, the server discards its entangled partner and retries. In the simulation, a qubit in memory is discarded if the elapsed time since entanglement exceeds half the coherence time, as mentioned in section 3.1.2. A shorter coherence time leads to an increased number of discarded qubits, resulting in more RSP attempts and significantly prolonging both the computation and simulation rounds. This is undesirable given that the expected computation time for the first phase was already expected to be 12 days (on average a single run was observed to take 0.011 seconds and the optimization was planned to run for $18,444$ runs for a population of 243 over 20 generations, resulting in $89,637,840$ runs). Consequently, the lower bound for the coherence time was set to half the baseline value, specifically $31\ ms$. Additionally, the coherence time is not expected to significantly affect the success probability, as the discarding of qubits after the cutoff time is implemented to prevent decoherence from having a large impact.

The initial loss probability posed a similar issue because for large values an increased initial loss probability led to more average attempts without significantly affecting the success probability, as shown in Figure 3.4. As the initial loss probability approaches one, the average number of attempts theoretically approaches infinity. This makes sense because an initial loss probability of 1 indicates a server efficiency of $\eta = 0$. In such a case, every single photonic qubit the server attempts to emit is lost, prompting the server to retry indefinitely. The lower bound for the initial loss probability is set at the point where the average time for a single RSP equals the cutoff time. This is because when the time for an RSP attempt exceeds the cutoff time, the 'old' qubit, which has decohered as described by equation (3.2), is discarded. The server then creates a new entangled pair, resulting in a 'newer' (i.e., higher quality) qubit. Beyond this point, the time per RSP does not significantly impact the fidelity of the qubit in memory (and therefore the success probability of the computation), since the duration a qubit remains in memory will be uniformly distributed from zero to the cutoff time, averaging out any effects on the success probability. This point was calculated as follows: the average time per RSP is defined as

$$\text{time per RSP} = \text{attempts per RSP} \times \text{time per attempt} \tag{3.13}$$

where

$$\text{attempts per RSP} = \frac{1}{p_{arrival}} = \frac{1}{(1 - p_{loss,inital}) \times (1 - p_{loss,fibre})} \tag{3.14}$$

with $p_{arrival}$ representing the probability of a photon arriving after the server attempts to send it, $p_{loss,inital}$ denoting the initial loss probability (i.e., the initial chance of losing a photon once it enters the quantum channel), and $p_{loss,fibre}$ indicating the probability of photon loss during transmission through the fibre (i.e., the quantum channel). The probability of loss in the fibre is given as

$$p_{loss,fibre} = 1 - 10^{-\frac{\alpha * L}{10}} \tag{3.15}$$

where α is the attenuation coefficient of the fibre in dB/km and L is the channel length. According to Table 3.1, the values of α and L are $0.2\ dB/km$ and $50\ km$, respectively. The time per attempt is the sum of the ion initialization duration, photon emission duration, and fibre propagation duration. The speed of light in an optical fibre is approximately $200000\ km/s$. Using these values, the average time per attempt is $250600\ ns$. With this, one can determine the value of $p_{loss,initial}$ at which the time per RSP crosses the cutoff time since the time per RSP is a function of the initial loss probability. This is illustrated in Figure 3.4a.

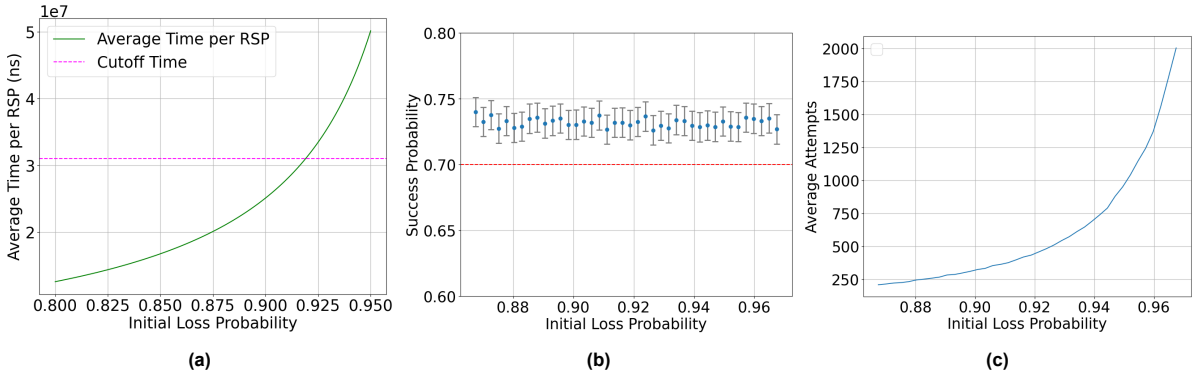


Figure 3.4: (a) The average time per RSP as a function of the initial loss probability is depicted according to equations (3.13 - 3.15). The intersection between the cutoff time and the average time per RSP occurs when the initial loss probability is approximately 0.92. (b) The success probability of the computation as a function of the initial loss probability is shown. This plot was generated by maintaining all other parameters at their baseline values while varying the initial loss probability. Using $15,000$ runs for a confidence interval of ± 0.011, it is evident that an increased initial loss probability does not significantly affect the success probability, as the success probability remains steady and does not fall below the red line, which represents the 0.7 success probability threshold. (c) The effect of increasing the initial loss probability on the average number of attempts is substantial. These average attempts were recorded during the simulation and averaged over $15,000$ runs for each point.

Simulated Parameter	lower bound	upper bound
Initial loss probability	0.8675	0.92
single-qubit depolarization probability	0.02	0.035
MS gate depolarization probability	0.1	0.2
Emission fidelity	0.88	0.974
Coherence time (ms)	31	62

Table 3.4: Parameter bounds. These upper and lower bounds span the parameter space and are given as input to the genetic algorithm.

With the upper and lower bounds of the optimization parameters established (as listed in Table 3.4), they can now be input into the genetic algorithm, which was executed using the YOTSE program [43]. Before optimization can commence, the algorithm requires an initial population. This is achieved by drawing 3 points for initial loss probability, 3 for single-qubit depolarization probability, 3 for MS gate depolarization probability, 3 for emission fidelity, and 3 for coherence time, resulting in an initial population of $3\times3\times3\times3\times3 = 243$ parameter sets. Every parameter set is run for $18,444$ rounds, yielding a confidence interval of ± 0.01. Each parameter set is then scored according to the cost function detailed in section 3.2.1. The best (i.e., lowest-costing) eight sets are selected to become the 'parents' of the next generation. These parents are combined with a mutation probability of 0.2 to produce the next generation. This process was planned to be repeated for 20 generations but was stopped after 12 generations due to increasing computation times per generation. Consequently, after $18,444 \times 3 \times 3 \times 3 \times 3 \times 3 \times 12 = 53,782,704$ runs of the simulation, an outcome is obtained.

These results are then passed on to the verification phase, which selects the set of parameters with the lowest cost that confidently meets the 70% threshold. The search phase runs each parameter set with a relatively large confidence interval, which means a set of parameters might reach the threshold due to this broader range. To address this, the verification phase runs simulations with the parameter set identified as having the lowest cost during the search phase, but with a significantly higher number of runs $(70,000)$ to achieve a small confidence interval of ± 0.0051. If this specific set of parameters does not meet the 70% threshold, the verification phase retries with the next lowest-cost set from the search phase, repeating the process until a solution that satisfies the threshold is found. The resulting set of parameters is considered minimal, meaning any further adjustments to lower the cost would no longer meet the required criteria.

4

Results and Discussion

The first results of this study reveal the dependence of the success probability for the non-verifiable Blind Quantum Computing (BQC) protocol on fibre length. In this protocol, the client's quantum capabilities are limited to making measurements only, while the server operates as a trapped-ion server. The client and server are separated by a distance and communicate over a quantum channel using an optical fibre with an attenuation coefficient of $0.2\ dB/km$.

A longer fibre implies that the photonic qubits sent from the server to the client must travel for a longer duration. This increased travel time raises the likelihood of photon loss within the fibre and extends the period during which a qubit in the server's memory undergoes decoherence. Consequently, this leads to a reduced success probability for the BQC computing round.

To determine the maximum separation distance that still yields a success probability of at least 70% for the BQC protocol, numerous simulations were conducted with varying fibre lengths. These findings are illustrated in Figure 4.1. The results indicate that, with current state-of-the-art hardware components, a separation distance of approximately 80 kilometers between the client and server maintains a success probability of 70%.

An additional search could be conducted by focusing more closely on the vicinity of the 80 km mark and increasing the number of runs per data point to achieve a smaller confidence interval. This approach was not pursued in this study because the results are highly dependent on the current state-of-the-art technology. The primary objective here was to determine the upper boundary of what is feasible with the existing technology.

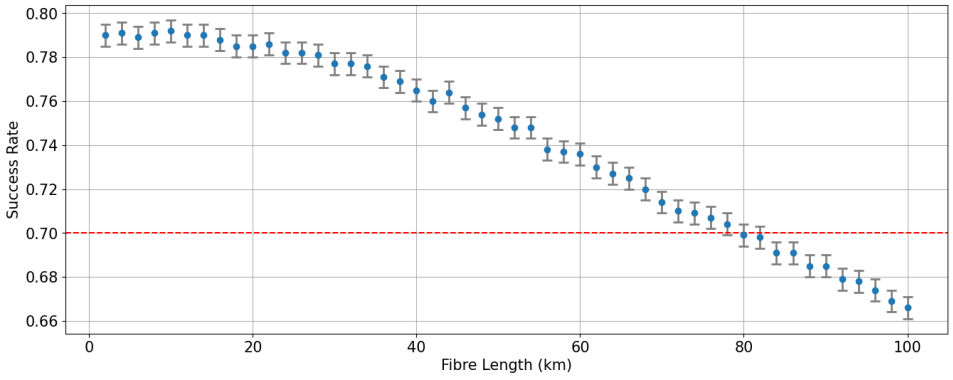
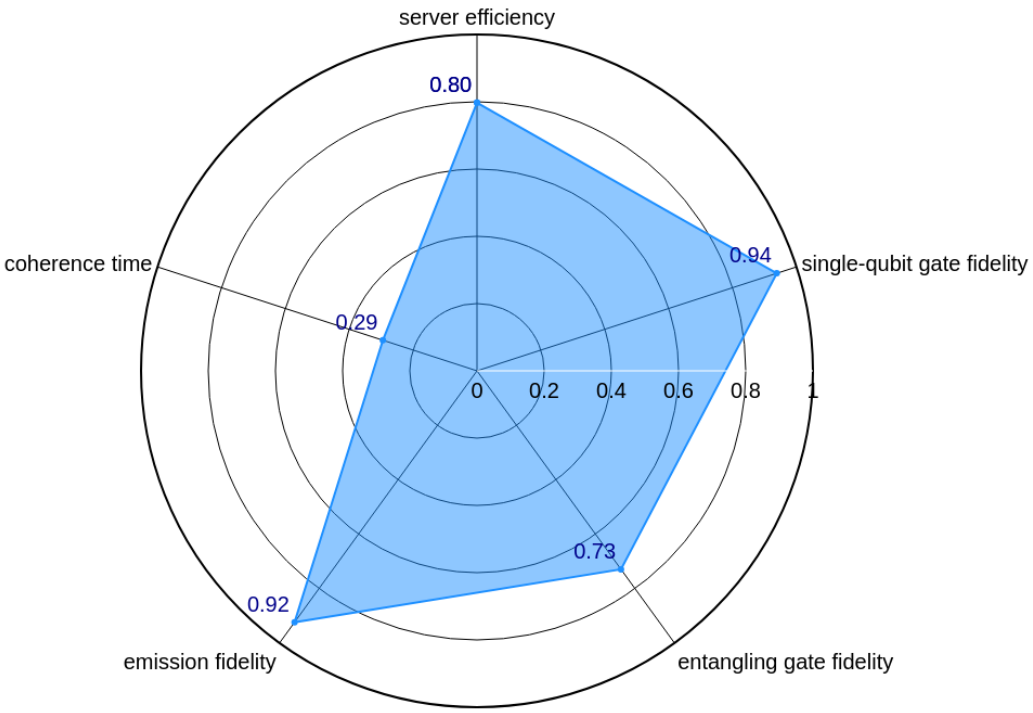


Figure 4.1: Plot of how the success probability depends on the fibre length (i.e. the length of the quantum channel connecting the client to te server). The data was obtained by varying only the fibre length while keeping all other parameters in the simulation at their baseline values. Furthermore, each datapoint was run and averaged over $70,000$ runs, resulting in a confidence interval of ± 0.0051 represented by the error bars. The red line depicts the 70% threshold for the success probability of the BQC protocol.

The second objective of this study was to determine the set of hardware parameters that can be maximally degraded from the current state-of-the-art values while still maintaining a minimum success probability of 70% for the BQC protocol. Identifying these parameters will establish the error margins for each hardware component when constructing the BQC setup in practice. This objective is essential for understanding the robustness and tolerances of the BQC protocol under suboptimal conditions. The results were obtained by employing a genetic algorithm with a custom cost function designed to search for the minimal values that hardware parameters can reach without the BQC protocol's success probability falling below 70%. To optimize computation time, the search was conducted in two phases. The first phase identified possible locations of the minimal parameter set, while the second phase verified the optimal location using more runs to achieve a smaller confidence interval. This approach resulted in a set of parameters considered minimal, meaning any further adjustments to lower the cost would no longer meet the required criteria.

Parameter	Baseline	Minimal value
Server Efficiency	0.1325	0.0807
Single-qubit gate fidelity	0.99	0.9893
Entangling gate fidelity	0.95	0.9327
Emission fidelity	0.974	0.9428
Coherence time (ms)	62	34

(a)



(b)

Figure 4.2: (a) Maximally degraded parameter set relative to baseline values that still achieve a minimum success probability of 70% for the non-verifiable Blind Quantum Computing protocol using a trapped-ion server and a measure-only client separated by 50 km of optical fibre. This set minimizes the cost function (3.9), representing the maximum allowable deviation from the state-of-the-art baseline while meeting the 70% success probability threshold. (b) Radar plot illustrating the improvement factors for various parameters. The radial axis denotes the improvement factor. A line closer to the center indicates a smaller improvement factor, meaning that the corresponding parameter can be degraded more with respect to its baseline value.

The results of this optimization are listed in Table 4.2a and visualized in Figure 4.2b. The values in Table 4.2a denote the maximally degraded parameter set relative to baseline values that still achieve a minimum success probability of 70% for the non-verifiable Blind Quantum Computing protocol using a trapped-ion server and a measure-only client separated by 50 km of optical fibre. This set minimizes the cost function (3.9), representing the maximum allowable deviation from the state-of-the-art baseline while meeting the 70% success probability threshold. Furthermore, it is important to note that the visualization in Figure 4.2b is not directly comparable across all parameters because coherence time is not represented on the same scale of zero to one as the other parameters. Looking at the results, it can be seen that the resulting minimal values for the coherence time and server efficiency approach the bounds that were set in section 3.2.2. It can also be seen that the coherence time shows greater tolerance for error compared to other parameters, meaning that a smaller coherence time does not significantly affect the success probability of the BQC protocol, which agrees with Figure 3.3a. Furthermore, the single-qubit gate fidelity has a tighter margin for error than other fidelities. This is because numerous single-qubit gates are executed to perform the CZ gate used in creating the graph state, making any imperfections in these gates more significant.

The genetic algorithm is known to be susceptible to finding local optima rather than the global optimum. To address this issue, this study employed a larger initial population while constraining the parameter space more tightly. Each parameter was bounded on one side by the baseline value and on the other by the absolute minimal requirements. This approach effectively created a densely populated search space, increasing the likelihood of finding the global optimum while reducing computation time.

On the other hand, the use of the two-phase strategy was less optimal since the search phase ran parameter sets over $18,444$ runs, resulting in a relatively large confidence interval of ± 0.01 (i.e., 1%). This larger confidence interval increased the likelihood of sub-optimal parameter sets being chosen as parents for the next generation of the algorithm, resulting in potential deviations from the true global optimum. Ideally, the optimization should have been run for $70,000$ runs over 20 generations with an initial population of 243, resulting in $340,200,000$ simulation rounds. This would have provided a smaller confidence interval and increased the accuracy of the optimization process, ultimately leading to a more robust identification of the minimal hardware parameters necessary for achieving the target success probability. Unfortunately, there was insufficient time to complete this, as the expected computation time for $340,200,000$ simulation rounds, with an average time of 0.01125 seconds per round, was approximately 26 days, which was more than the time there was left to finish this study. On top of that, each generation of the optimization was observed to take significantly longer than the previous generation, resulting in even larger computation times than initially expected (on average, the resulting set of parameters was observed to run for more than 26 minutes to finish $70,000$ runs. This means that $340,200,000$ runs would have taken more than 87 days). This was because the algorithm constantly tries to decrease the coherence time and server efficiency, resulting in an increase in computation time. Given this constraint, a two-phase search was implemented to significantly reduce computation time, albeit with an increased likelihood of finding a local optimum instead of the global optimum. Another debatable point in this study is the chosen lower boundary for the coherence time. The boundary could have been set lower, potentially allowing the algorithm to find solutions with a lower cost. However, the decision was made to limit the lower bound of the coherence time to half the baseline value. This is because a lower coherence time would lead to longer computation times due to an increased number of RSP attempts, which was undesirable for the same reasons previously mentioned. Therefore, the chosen lower bound for the coherence time was deemed appropriate.

These considerations naturally lead to several future work recommendations. First, the lower bound of the coherence time could be further reduced to its absolute minimal requirement to potentially discover a more optimal solution. Additionally, rerunning the algorithm with an increased number of runs (e.g., $70,000$) would reduce the confidence interval, potentially yielding a better solution. However, these recommendations will likely face the challenge of long computation times.
To address this issue, the code could be optimized by implementing state insertion, which can be achieved using a framework similar to NetSquid's entanglement generation Magic [44]. Another approach could involve modifying the cost function to include an argument that either rewards a smaller number of RSP attempts or penalizes a larger number of RSP attempts. This would ensure that the number of RSP attempts remains manageable, preventing the high number of RSP attempts observed in this work. Consequently, this adjustment would lead to shorter computation times both in the simula-

tion and in practical applications. This aspect was not explored in this work because the primary focus was on minimizing error probability, and the benefits of a lower rate were not immediately clear beyond the comfort of reduced computation times. Additionally, it was unclear how significant the penalty or reward should be to avoid creating a biased result where the rate becomes more important than the error probability. If the rate is considered important, the challenge lies in determining its relative importance compared to error probability.

References

[1] Peter W Shor. “Algorithms for quantum computation: discrete logarithms and factoring”. In: *Proceedings 35th annual symposium on foundations of computer science*. Ieee. 1994, pp. 124–134.

[2] Yudong Cao, Jhonathan Romero, and Alán Aspuru-Guzik. “Potential of quantum computing for drug discovery”. In: *IBM Journal of Research and Development* 62.6 (2018), pp. 6–1.

[3] Li Xi-Han et al. “Quantum secure direct communication with quantum encryption based on pure entangled states”. In: *Chinese Physics* 16.8 (2007), p. 2149.

[4] Joseph Fitzsimons and Elham Kashefi. “Universal blind quantum computation”. In: *Bulletin of the American Physical Society* 57 (2012).

[5] Tomoyuki Morimae and Keisuke Fujii. “Blind quantum computation protocol in which Alice only makes measurements”. In: *Physical Review A—Atomic, Molecular, and Optical Physics* 87.5 (2013), p. 050301.

[6] Joseph F Fitzsimons. “Private quantum computation: an introduction to blind quantum computing and related protocols”. In: *npj Quantum Information* 3.1 (2017), p. 23.

[7] P Drmota et al. “Verifiable blind quantum computing with trapped ions and single photons”. In: *Physical Review Letters* 132.15 (2024), p. 150604.

[8] Janice van Dam et al. “Hardware requirements for trapped-ion based verifiable blind quantum computing with a measurement-only client”. In: *arXiv preprint arXiv:2403.02656* (2024).

[9] Dominik Leichtle et al. “Verifying BQP computations on noisy devices with minimal overhead”. In: *PRX Quantum* 2.4 (2021), p. 040302.

[10] Stefanie Barz et al. “Demonstration of blind quantum computing”. In: *science* 335.6066 (2012), pp. 303–308.

[11] Chiara Greganti et al. “Demonstration of measurement-only blind quantum computing”. In: *New Journal of Physics* 18.1 (2016), p. 013020.

[12] J Schupp et al. “Interface between trapped-ion qubits and traveling photons with close-to-optimal efficiency”. In: *PRX quantum* 2.2 (2021), p. 020331.

[13] Tim Coopmans et al. “Netsquid, a network simulator for quantum information using discrete events”. In: *Communications Physics* 4.1 (2021), p. 164.

[14] Guus Avis. *NetSquid trapped-ions snippet*. 2023. URL: `https://docs.netsquid.org/snippets/netsquid-trappedions/modules/ion_trap.html`.

[15] Francisco Ferreira Da Silva et al. “Optimizing entanglement generation and distribution using genetic algorithms”. In: *Quantum Science and Technology* 6.3 (2021), p. 035007.

[16] Guus Avis et al. “Requirements for a processing-node quantum repeater on a real-world fiber grid”. In: *NPJ Quantum Information* 9.1 (2023), p. 100.

[17] Benjamin Schumacher. “Quantum coding”. In: *Phys. Rev. A* 51 (4 Apr. 1995), pp. 2738–2747. DOI: `10.1103/PhysRevA.51.2738`. URL: `https://link.aps.org/doi/10.1103/PhysRevA.51.2738`.

[18] Lov K Grover. “A fast quantum mechanical algorithm for database search”. In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 1996, pp. 212–219.

[19] Bova, Francesco, Goldfarb, Avi, and Melko, Roger G. “Commercial applications of quantum computing”. In: *EPJ Quantum Technol.* 8.1 (2021), p. 2. DOI: `10.1140/epjqt/s40507-021-00091-1`. URL: `https://doi.org/10.1140/epjqt/s40507-021-00091-1`.

[20] Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010.

[21] Robert Raussendorf and Hans J Briegel. “A one-way quantum computer”. In: *Physical review letters* 86.22 (2001), p. 5188.

[22] Robert Raussendorf and Hans Briegel. “Computational model underlying the one-way quantum computer”. In: *arXiv preprint quant-ph/0108067* (2001).

[23] Vincent Danos, Elham Kashefi, and Prakash Panangaden. “The measurement calculus”. In: *Journal of the ACM (JACM)* 54.2 (2007), 8–es.

[24] William K. Wootters and Wojciech H. Zurek. “The no-cloning theorem”. In: *Physics Today* 62.2 (Feb. 2009), pp. 76–77. ISSN: 0031-9228. DOI: 10.1063/1.3086114. eprint: https://pubs.aip.org/physicstoday/article-pdf/62/2/76/11300983/76_1_online.pdf. URL: https://doi.org/10.1063/1.3086114.

[25] Charles H Bennett and Gilles Brassard. “Quantum cryptography: Public key distribution and coin tossing”. In: *Theoretical computer science* 560 (2014), pp. 7–11.

[26] Momtchil Peev et al. “The SECOQC quantum key distribution network in Vienna”. In: *New Journal of Physics* 11.7 (2009), p. 075001.

[27] Masahide Sasaki et al. “Field test of quantum key distribution in the Tokyo QKD Network”. In: *Optics express* 19.11 (2011), pp. 10387–10409.

[28] Damien Stucki et al. “Long-term performance of the SwissQuantum quantum key distribution network in a field environment”. In: *New Journal of Physics* 13.12 (2011), p. 123001.

[29] Shuang Wang et al. “Field and long-term demonstration of a wide area quantum key distribution network”. In: *Optics express* 22.18 (2014), pp. 21739–21756.

[30] TOSHIBA. *Quantum Key Distribution Products*. 2024. URL: https://www.global.toshiba/ww/products-solutions/security-ict/qkd/products.html.

[31] Q*bird. *Quantum Key Distribution Products*. 2024. URL: https://q-bird.com/products/.

[32] Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. “Universal blind quantum computation”. In: *2009 50th annual IEEE symposium on foundations of computer science*. IEEE. 2009, pp. 517–526.

[33] Wassily Hoeffding. “Asymptotically Optimal Tests for Multinomial Distributions”. In: *The Annals of Mathematical Statistics* 36.2 (1965), pp. 369–401. ISSN: 00034851. URL: http://www.jstor.org/stable/2238145 (visited on 06/22/2024).

[34] Victor Krutyanskiy et al. “Telecom-wavelength quantum repeater node based on a trapped-ion processor”. In: *Physical Review Letters* 130.21 (2023), p. 213601.

[35] Laurent Stephenson. *Entanglement between nodes of a quantum network*. University of Oxford (United Kingdom), 2020.

[36] Viktor Krutyanskiy et al. “Light-matter entanglement over 50 km of optical fibre”. In: *npj Quantum Information* 5.1 (2019), p. 72.

[37] A Stute et al. “Quantum-state transfer from an ion to a photon”. In: *Nature photonics* 7.3 (2013), pp. 219–222.

[38] B. Casabone et al. “Enhanced Quantum Interface with Collective Ion-Cavity Coupling”. In: *Phys. Rev. Lett.* 114 (2 Jan. 2015), p. 023602. DOI: 10.1103/PhysRevLett.114.023602. URL: https://link.aps.org/doi/10.1103/PhysRevLett.114.023602.

[39] A Stute et al. “Tunable ion–photon entanglement in an optical cavity”. In: *Nature* 485.7399 (2012), pp. 482–485.

[40] R Clark Jones. “A new calculus for the treatment of optical systemsi. description and discussion of the calculus”. In: *Josa* 31.7 (1941), pp. 488–493.

[41] R. Simon and N. Mukunda. “Minimal three-component SU(2) gadget for polarization optics”. In: *Physics Letters A* 143.4 (1990), pp. 165–169. ISSN: 0375-9601. DOI: https://doi.org/10.1016/0375-9601(90)90732-4. URL: https://www.sciencedirect.com/science/article/pii/0375960190907324.

[42] Klaus Mølmer and Anders Sørensen. “Multiparticle Entanglement of Hot Trapped Ions”. In: *Phys. Rev. Lett.* 82 (9 Mar. 1999), pp. 1835–1838. DOI: 10.1103/PhysRevLett.82.1835. URL: https://link.aps.org/doi/10.1103/PhysRevLett.82.1835.

[43] Maxim Masterov, Ariana Torres, and David Maier. *YOTSE GitHub repository*. 2024. URL: https://github.com/SURFQuantum/yotse.

[44] Guus Avis et al. *NetSquid Magic snippet*. 2024. URL: https://docs.netsquid.org/snippets/netsquid-magic/.

A

Quantum Fundamentals

A.1. Linear Algebra for Quantum Meachanics

Here, we assume that the reader is familiar with the basic concept of vector space. In quantum mechanics, we consistently deal with vector spaces that are finite-dimensional and defined over the complex numbers. These vector spaces are *Hilbert spaces*, denoted as $\mathcal{H}$, characterized by an inner product, $\langle\psi|\phi\rangle$, where ψ and ψ are vectors.

A.1.1. Bra-Ket Notation

In order to implement Quantum Computing we would like control quantum degrees of freedom. To talk about these degrees of freedom, the bra-ket notation was developed. Bra-ket notation, also known as Dirac notation, is a notation for linear algebra and linear operators in complex vector space.

In Quantum Mechanics, it is custom to call elements of $\mathcal{H}$ "kets" wich we shall denote in the form $|\psi\rangle$. The dual vectors are called "bras" and are denoted as $\langle\phi|$. The combination of the two is an inner product and forms a so-called "braket": $\langle\psi|\phi\rangle \in \mathbb{C}$. The outer product, which results in an operator, is written as $|\psi\rangle\langle\phi|$. This operator can act on other kets or bras to produce new vectors or scalars. For example, applying the outer product to a ket $|\phi\rangle$ yields $(|\psi\rangle\langle\phi|)|\phi\rangle = |\psi\rangle\langle\phi|\phi\rangle$.

Bra-ket notation is extensively used in quantum mechanics to simplify expressions involving quantum states, operators, and measurements. It provides a concise and intuitive way to describe complex quantum phenomena, making it an essential tool for studying and implementing quantum computing.

A.1.2. Linear Operators and Gates

Linear operators are naturally associated with vector spaces, defined as linear maps from a vector space to itself. In finite-dimensional spaces, these operators are often represented by matrices.

Pauli Gates

The Pauli gates (X, Y, Z) correspond to the three Pauli matrices (σ_x, σ_y, σ_z) and act on a single qubit. These gates represent rotations around the x, y and z axes of the Bloch sphere by π radians, respectively.

The Pauli-X gate functions as the quantum equivalent of the classical NOT gate within the standard basis $|0\rangle$ and $|1\rangle$, which align with the z-axis on the Bloch sphere. Known as the bit-flip gate, it swaps $|0\rangle$ with $|1\rangle$ and vice versa. The Pauli-Y gate transforms $|0\rangle$ into $i|1\rangle$ and $|1\rangle$ into $-i|0\rangle$. The Pauli-Z gate leaves $|0\rangle$ unchanged but converts $|1\rangle$ into $-|1\rangle$. This operation earns the Pauli-Z gate the name phase-flip gate due to its specific action on the phase of the $|1\rangle$ state.

The matrix representation of the Pauli gates are represented as

$$X = \sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{A.1}$$

$$Y = \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \tag{A.2}$$

$$Z = \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{A.3}$$

Controlled Gates

The $\wedge Z_{ij}$ gate is a two-qubit gate commonly used in quantum computing. It operates on a pair of qubits, where one qubit serves as the control and the other as the target. Essentially, the $\wedge Z_{ij}$ gate applies a phase flip (a change in the relative phase) to the target qubit only when the control qubit is in the $|1\rangle$ state. If the control qubit is in the $|0\rangle$ state, the $\wedge Z_{ij}$ gate has no effect on the target qubit. The matrix representation of the $\wedge Z_{ij}$ gate has the following form:

$$\wedge Z_{ij} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \tag{A.4}$$

A.1.3. Tensor Products and Entanglement

The definition of the *tensor product* is as follows:

Given two Hilbert spaces $\mathcal{H}$ with basis vectors $\{a_i | 1 \leq i \leq n\}$ and $\mathcal{H}'$ with basis $\{b_j | 1 \leq j \leq m\}$ we define the tensor product, written $\mathcal{H} \otimes \mathcal{H}'$, as the vector space of dimension $n \cdot m$ with basis $a_i \otimes b_j$ [23]

More elegant, basis-independent methods exist for describing the tensor product, but the following definition suffices for our purposes. In Dirac notation, the symbol $\otimes$ is typically omitted; for example, $|\psi\rangle \otimes |\phi\rangle$ is written as $|\psi\phi\rangle$. Importantly, not all vectors can be expressed as the tensor product of other vectors. For instance, the vector $a_1 \otimes b_1 + a_2 \otimes b_2$, with a_i and b_i as basis vectors of two 2-dimensional Hilbert spaces, illustrates that general elements of $\mathcal{H} \otimes \mathcal{H}'$ cannot be decomposed into elements of $\mathcal{H}$ and $\mathcal{H}$'. This illustrates the concept of entanglement in mathematical terms.

A.1.4. Graph States

A particularly valuable type of quantum states is graph states, also referred to as cluster states. As indicated by their name, a graph state $|G\rangle$ is associated with a graph,

$$G = (V, E) \tag{A.5}$$

where the vertices ($v \in V$) represent qubits initialized into the $|+\rangle$ state, and edges ($e \in E$) represent the application of $\wedge Z_{ij}$ gates between respective vertices. A graph state can thus be expressed as

$$|G\rangle = [\prod_{(i,j)\in E} \wedge Z_{ij}] \, |+\rangle^{\otimes |V|} \tag{A.6}$$