UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE CIENCIAS FÍSICAS

TESIS DOCTORAL

Encriptación homomórfica cuántica de algoritmos cuánticos

Quantum homomorphic encryption of quantum algorithms

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

Pablo Fernández Ortiz

DIRECTOR

Miguel Ángel Martín-Delgado Alcántara

Encriptación homomórfica cuántica de algoritmos cuánticos

Quantum homomorphic encryption of quantum algorithms

Pablo Fernández Ortiz

Director: Miguel Ángel Martín-Delgado Alcántara

Facultad de Ciencias Físicas

Universidad Complutense de Madrid

Memoria para optar al título de

Doctor en Física

Acknowledgements

Me gustaría agradecer al director de la tesis, Miguel Ángel Martín-Delgado, por su dirección y por la gran libertad que me ha concedido para decidir las líneas de investigación durante la duración de la tesis. Me gustaría agradecer también la labor de Carmen Page, por su ayuda con todos los trámites que han surgido en el transcurso de esta tesis.

Quiero dar las gracias a Sergio Ángel Ortega, por su inestimable ayuda durante la colaboración que realizamos y su gran capacidad de trabajo. También me gustaría agradecer al resto de compañeros del grupo de investigación, Roberto Campos, Sara Giordano y Gabriel Escrig, por los buenos momentos que hemos pasado en varios congresos.

Por último me gustaría agradecer a mi familia y amigos por su apoyo durante estos años.

Contents

Abstract

Since its inception in the 20th century, quantum mechanics has revolutionized our understanding of nature, introducing new concepts such as the uncertainty principle and quantum interference. In the 21st century, we could say that we are living a new revolution thanks to the new quantum technologies based on this theory. Within these technologies, two branches of great importance are quantum computing and quantum communication. Quantum computing seeks to create algorithms with computational advantages over algorithms used in classical computers. On the other hand, quantum communication seeks to develop protocols for transmitting information with security based on the fundamental properties of quantum mechanics, superior to the security of current protocols.

At the intersection of both disciplines, a new technology that has recently been developed is homomorphic encryption. This technology allows a client to encrypt its data and send it to a remote server, which will perform operations on the encrypted data. Once the server completes the desired operations, it returns the data to the client for decryption. In this manner, the server gains no information about the client's data. It is expected that most quantum computers of the future will be accessed through the cloud. Therefore, quantum homomorphic encryption schemes are good candidates for preserving the security of the calculations performed on these computers. In this thesis, we have applied quantum homomorphic encryption schemes to various quantum algorithms, with the aim of demonstrating that the homomorphic implementation of these algorithms can be performed efficiently. In addition, we will perform simulations of some of the homomorphic implementations of the quantum algorithms studied in order to verify their correct functioning.

In the first chapter we will present a brief overview of quantum computing and quantum communication. We will give a historical perspective and present some of the most recent advances in these fields.

In Chapter 2, a brief introduction to quantum homomorphic encryption will be presented and one of these schemes will be described in great detail. This scheme is the basis of the thesis and it will be applied to different quantum algorithms in the following chapters.

In Chapter 3, we will describe the Bernstein-Vazirani algorithm, both in its standard and recursive versions. In Chapter 4, we will present our first publication, which deals

with the homomorphic implementation of the recursive Bernstein-Vazirani algorithm using the protocol presented in Chapter 2. We will study the cases in which such an implementation can be performed efficiently in detail.

In Chapter 5, we will describe Grover's quantum search algorithm. In Chapter 6, we will present our second publication, which will show a simulation of the homomorphic implementation of Grover's algorithm using Qiskit. Furthermore, it will be shown that such an implementation can be performed efficiently.

In Chapter 7, we will describe Szegedy walks algorithms, both in their quantum and semiclassical versions. Finally, in Chapter 8, we will present our third publication, which addresses the simulation of the homomorphic implementation of these algorithms using classical-quantum circuits in Qiskit. We will demonstrate that these implementations can be performed efficiently, and we will also present a Python library for the homomorphic simulation of a large number of quantum circuits.

Resumen

Desde su concepción en el siglo XX, la mecánica cuántica ha revolucionado nuestra forma de entender la naturaleza, presentando nuevos conceptos como el principio de incertidumbre o la interferencia cuántica. En el siglo XXI, podríamos decir que estamos viviendo una nueva revolución gracias a las nuevas tecnologías cuánticas basadas en esta teoría. Dentro de estas tecnologías, dos ramas de gran importancia son la computación cuántica y la comunicación cuántica. La computación cuántica busca crear algoritmos con ventajas computacionales frente a los algoritmos utilizados en ordenadores clásicos. Por otro lado, la comunicación cuántica busca desarrollar protocolos para transmitir información con una seguridad basada en las propiedades fundamentales de la mecánica cuántica, superior a la seguridad de los protocolos actuales.

En la intersección de ambas disciplinas, una nueva tecnología que se ha desarrollado recientemente es la encriptación homomórfica. Esta tecnología permite a un cliente encriptar sus datos y enviarlos a un servidor remoto que realizará operaciones sobre estos datos encriptados. Una vez que el servidor complete las operaciones deseadas, este devuelve los datos al cliente para que pueda desencriptarlos. De esta manera, el servidor no obtiene información sobre los datos del cliente. Se espera que la mayor parte de ordenadores cuánticos del futuro sean accedidos a través de la nube. Por ello, los protocolos de encriptación homomórfica cuántica son buenos candidatos para preservar la seguridad de los cálculos realizados en estos ordenadores. En esta tesis, hemos aplicado protocolos de encriptación homomórfica cuántica a diversos algoritmos cuánticos, con el objetivo de demostrar que la implementación homomórfica de estos algoritmos se puede realizar eficientemente. Además realizaremos simulaciones de algunas de las implementaciones homomórficas de los algoritmos cuánticos estudiados, con el objetivo de comprobar su correcto funcionamiento.

En el primer capítulo presentaremos una breve descripción general de la computación cuántica y la comunicación cuántica. Daremos una perspectiva histórica y presentaremos algunos de los avances más recientes en estos campos.

En el capítulo 2 se presentará una breve introducción a la encriptación homomórfica cuántica y se describirá en gran detalle uno de estos protocolos. Este protocolo es la base de la tesis y se aplicará a diferentes algoritmos cuánticos en los siguientes capítulos.

En el capítulo 3 describiremos el algoritmo de Bernstein-Vazirani, en su versión estándar y su versión recursiva. En el capítulo 4 presentaremos nuestra primera publicación, que

trata sobre la implementación homomórfica del algoritmo de Bernstein-Vazirani recursivo utilizando el protocolo presentado en el capítulo 2. Estudiaremos en detalle los casos en los que dicha implementación puede hacerse eficientemente.

En el capítulo 5 describiremos el algoritmo de búsqueda cuántica de Grover. En el capítulo 6 presentaremos nuestra segunda publicación, en la que se mostrará una simulación de la implementación homomórfica del algoritmo de Grover usando Qiskit. Además, se demostrará que dicha implementación puede realizarse eficientemente.

En el capítulo 7 describiremos los algoritmos de paseos de Szegedy, tanto en su versión cuántica como en su versión semiclásica. Finalmente en el capítulo 8 presentaremos nuestra tercera publicación, en la que se aborda la simulación de la implementación homomórfica de dichos algoritmos utilizando circuitos clásico-cuánticos en Qiskit. Se demostrará que estas implementaciones se pueden realizar de forma eficiente y además se presentará una librería en Python para la simulación homomórfica de una gran cantidad de circuitos cuánticos.

List of publications

1. P. Fernández and M. A. Martin-Delgado, "Homomorphic encryption of the $k = 2$ Bernstein–Vazirani algorithm", Journal of Physics A: Mathematical and Theoretical 57, 365301 (2024)

2. P. Fernández and M. A. Martin-Delgado, "Implementing the Grover algorithm in homomorphic encryption schemes", Physical Review Research 6, 043109 (2024)

3. S. A. Ortega, P. Fernández and M. A. Martin-Delgado, "Implementing semiclassical Szegedy walks in classical-quantum circuits for homomorphic encryption", Journal of Physics: Complexity 6, 025010 (2025)

Conference contributions

1. CISM-UniUD Joint Advanced School on Quantum Machine Learning: from Fundamentals to Applications (attendance), Udine, Italy, September 2022
2. QUANTUMatter2023 Conference (poster), Madrid, Spain, May 2023
3. 17th Granada Seminar (poster), Granada, Spain, September 2023
4. Primera Reunión Nacional del Plan Complementario de Comunicaciones Cuánticas (poster), Madrid, Spain, September 2023
5. The XXXIX RSEF Physics Biennial (talk), San Sebastián, Spain, July 2024
6. ETSI/IQC Quantum Safe Cryptography Conference 2025 (poster), Madrid, Spain, June 2025

List of Figures

List of Tables

1 Quantum Computation and Communication: an Overview

1.1 Quantum computation

Since it was introduced at the 20th century, Quantum mechanics has been an extremely successful theory. Along with general relativity, it is one of the current foundations of our understanding of physical reality. Quantum mechanics has been successfully applied to many different systems, such as elementary particles, the interior of the atom or the theory of semiconductors.

One of the applications that has generated a great amount of interest in recent decades is the field of quantum computation. In 1981, Richard Feynman presented a talk at a conference titled "Simulating physics with computers" [1]. He proposed using quantum computers to simulate quantum systems, as these systems are too complex to be simulated using conventional classical computers. This talk launched the study of quantum computation as a distinct field of study.

Then in 1985 David Deutsch presented a formal notion of a quantum computer, known as a quantum Turing machine [2]. The devices that he considered were quantum analogues of the classical machines defined almost fifty years ago by Turing [3]. Deutsch's work also presented an algorithm known today as Deutsch algorithm, which suggested that the computational power possessed by quantum computers may actually exceed the computational power that classical computers have. Then in 1993 Bernstein and Vazirani formulated a problem that could be solved using a quantum algorithm. This algorithm showed a superpolynomial speedup over a classical computer [4]. Soon after, Simon proposed a quantum algorithm that demonstrated an exponential speedup compared to classical algorithms [5].Simon's algorithm does not have any direct applications but it still served as inspiration for Peter Shor, who formulated two quantum algorithms, one for computing discrete logarithms and the other for efficient factorization of large numbers [6]. Shor's factoring algorithm runs in polynomial time, which is considerably faster than the best classical algorithms for this problem, as they work in non-polynomial time. Since Shor's algorithm had a direct impact on cryptanalysis, the interest in the field of quantum computation increased enormously. On the other hand, some physicists expressed concerns about the ability of quantum computers to work effectively in the future due to the negative effects of decoherence [7, 8]. Quantum states are loosely

coupled with their environments and they tend to lose information as they interact with them. This makes maintaining the desired quantum behaviours, such as superposition or entanglement, quite difficult.

Nevertheless, the next pivotal advances followed soon after with the formulation of quantum error-correcting codes [9,10] and of fault-tolerant methods for performing quantum computations using noisy hardware in a reliable manner [11]. Soon after, the so called "threshold theorem" was formulated [12–15]. It states that a quantum computer that has a physical error rate below a certain threshold can suppress the logical error rate to arbitrarily low levels by making use of quantum error correction schemes. Due to these results, it was thought that quantum computing could be scaled up to large devices that solve complex problems, assuming that errors that affect the hardware are not too strongly correlated or not too common. During these years, there were also proposals that used quantum optics and atomic physics to implement quantum logical operators [16]. The most significant applications of quantum computers are expected to be developed in the fields of quantum simulation, quantum chemistry and optimization problems [17]

Despite all this progress in recent decades, it is important to remark that quantum computers are still not fully functional [18]. This is because both the number of qubits and the accuracy of our quantum processors are still quite modest, despite the progress that different approaches for building quantum hardware have experimented. An indicator of the current status of quantum computing could be considered the milestone reached by Google in 2019 [19]. They demonstrated "quantum supremacy" [20], the idea that a quantum computer can solve a problem that no classical computer can solve in any feasible time, although this problem has no real application. Using superconducting quantum technology, they constructed a quantum computer known as Sycamore with 53 qubits arranged in a two-dimensional array. This way, entangling two-qubit quantum gates can be performed on adjacent qubits in the array. Then they executed up to 20 layers of two-qubit gates, and measured all the qubits once the circuit was completed. Since the hardware sometimes produces errors, the final measurement yields the correct result only once in 500 runs of the computation. Nevertheless, they managed to extract a statistically useful signal by repeating the same computation millions of times in just a few minutes. If best currently known classical methods were used, simulating what Sycamore achieved in a few minutes would take a few days at least for the most powerful existing classical supercomputer. Moreover, this problem would be far beyond the classical computer's reach if just a few more qubits could be added, since the cost of the classical simulation rises exponentially with the number of qubits. We remark that the problem solved by Sycamore has no particular interest for any purpose besides demonstrating quantum computational supremacy.

The current types of quantum computers have been called the "Noisy Intermediate-Scale Quantum" (NISQ) computers [21]. These quantum computers contain up to a thousand qubits. Since they are not error corrected, their computational power is lim-

ited by noise. Nevertheless, they are still capable of performing small-scale quantum algorithms, so they still serve as a platform for algorithm development. Besides this, NISQ devices are exciting for physicists because they provide new tools for studying the properties of complex many-particle quantum systems in a regime that has never been experimentally accessible before.

An interesting type of algorithm in the current NISQ era are the so called hybrid classical-quantum algorithms, since they could harness the current capabilities of classical computers and boost them using NISQ processors [22,23]. Different works [24] believe that hybrid algorithms could lead to pure quantum algorithms, since they can be used as an intermediate step in order to learn more about quantum information processing, mitigate errors of quantum processors and ultimately develop pure quantum algorithms.

Quantum simulation is an application worth of mentioning. It is interesting to distinguish between digital and analog quantum simulation. An analog quantum simulator is a system that contains many qubits whose dynamics resembles those of a model system that somebody wants to study. On the other hand, a digital quantum simulator is a universal quantum computer based on quantum gates that can be used to simulate any physical system of interest when correctly programmed. It can also be used for other purposes. Analog quantum simulation has been a highly active area of research for the past two decades [25], while digital quantum simulation using a general circuit-based quantum computer is currently getting started. Some of the same experimental platforms can be applied to both purposes, such as superconducting circuits and trapped ions, whereas other systems are more suited to be used as analog simulators, like molecules and trapped neutral atoms. Analog quantum simulators have become more sophisticated in recent years. For this reason they are already being used to analyse quantum dynamics in regimes that might be beyond the reach of classical simulators [26, 27]. They have also been used to create highly entangled equilibrium states of quantum matter with the objective of studying their properties [28].

Analog quantum simulators will eventually be surpassed by digital quantum simulators, since the former are difficult to control and the latter can be controlled using quantum error correction [18]. However due to the high cost of quantum error correction, the use of analog quantum simulators may continue for many years Thus, the potential power of analog quantum simulators should not be overlooked in the near term search of applications of quantum technology.

Classical computation can be thought as the manipulation of bits using logical gates such as AND, OR, XOR, etc. The bit is the basic unit of classical computing, and it can have two possible values, 0 or 1. In contrast, the basic unit of information in quantum computing is the qubit, which can have linear combinations of 0 and 1 due to its quantum mechanical nature. This property that allows the qubits to be in two states at the same time is known as superposition. Once the qubit is measured, it becomes either 0 or 1. This superposition of qubits provides access to a highly condensed representation of information. Another important property that quantum computing

presents is entanglement. It describes the effect in which a pair or group of quantum states are correlated. After they are entangled, they stop being independent from each other. If any of these quantum states is measured, it affects the remaining states, making them yield definite results when measured regardless of their distance from each other. For example, if Alice and Bob each possess one state of an entangled pair, the measurement of Alice's state instantly determines the outcome of Bob's measurement regardless of the distance they are separated by.

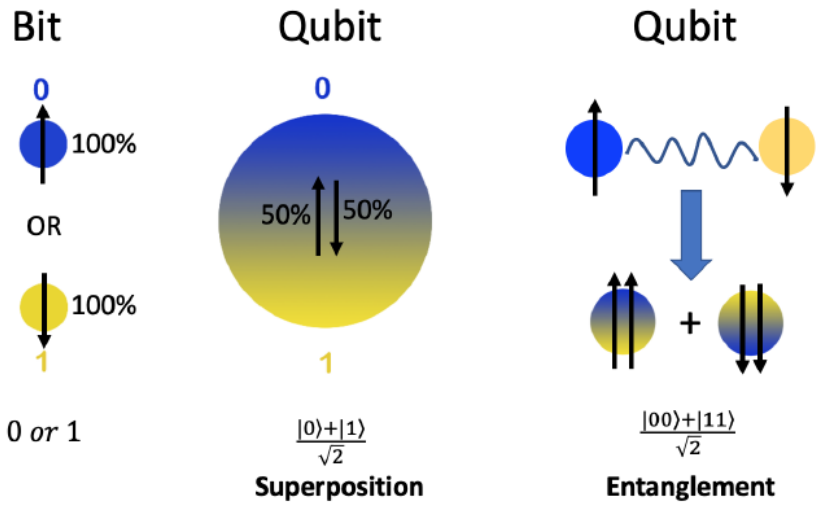


Figure 1.1: Image obtained from [29]. At the left we have the illustration of a bit which can take a value of 0 or 1 with 100% probability. At the middle we have a qubit that can be in a state of $|0\rangle$, $|1\rangle$ or in a superposition state of both $|0\rangle$ and $|1\rangle$. Here, a qubit is illustrated in a superposition state, in which there is a 50% of measuring the state $|0\rangle$ and 50% of measuring $|1\rangle$. At the right we have a illustration of two qubits in an entangled state. The properties of the two qubits in the entangled state are related to each other so that measuring one of them will reveal the state of the other qubit, even when they separated by large distances.

Superposition and entanglement are at the core of many quantum algorithms and protocols. They are represented in figure 1.1. Another important quantum mechanical property that has to be taken into account is the no-cloning theorem, which states that it is impossible to produce an identical copy of an arbitrary unknown quantum state [30].

Although quantum computing is still in its embryonic phase, its potential to revolutionize the classical infrastructure has been noticed by major companies and governments around the world. Reports have shown that the global investment in quantum computing reached almost 25 billion dollars in mid-2021 [31]. Companies such as IBM, Google, Intel, D-Wave, IonQ and Rigetti have shown significant interest in quantum computing over the past few years [32,33], continuously announcing their breakthroughs in quantum

hardware and software development kits (SDKs).

Different models have been used for building quantum computers, the most commonly used is the quantum logic gates model in which a quantum circuit is implemented based on quantum logic gates. This model leads to digital quantum computers that can perform quantum computations using a universal quantum logic gate set. Besides this, different types of technologies have been proposed to build quantum computers depending on the method used to create or encode the qubits, such as superconducting qubits [34–40]. Spin qubits are encoded in the spin state of a single electron and atomic qubits are encoded in the quantum state of an atom or ion. In the case of photonic qubits, these are encoded in the frequency, polarization or spatial mode of a photon. Superconducting qubits are encoded in the quantum state of a superconducting circuit. This is the type of technology chosen by IBM and Google. Each type of qubit has its particular advantages and disadvantages. Superconducting and photonic qubits are relatively simple to operate and transmit, but they are more susceptible to noise and decoherence. In contrast, atomic and spin qubits are more stable against noise, but they can be more demanding to scale [33]. A more extensive review about the different kind of qubits can be found in [41].

Currently, some of the quantum computers that have already been developed are available in the cloud, such as the IBM Q quantum computers. Researchers have performed plenty of experiments and different types of protocols on them, like obtaining a measurement of the topological Uhlmann phase for a topological insulator [42], implementing a great variety of representative open quantum systems models [43], implementing Shor's algorithm to factor the numbers $N = 15, 21$ and 35 using five, six and seven superconducting qubits respectively [44] or executing a quantum search algorithm over structured datasets [45].

Despite all the progress that the field of quantum computing has experimented, there are still critical challenges regarding the quantum hardware. These challenges are [33]:

- Noise: as we have already mentioned, the current quantum computers are considered to be in the NISQ era, so they have to deal with noise. Coherence between quantum states tends to disappear due to their interaction with the environment. Quantum states must be perfectly isolated from their outside environment in order to preserve quantum information. However, we need to interact with the states if we want to manipulate or measure them. These operations break the isolation of a quantum system and generate decoherence. This leads to errors which make computations random and incorrect. In order to prevent these errors caused by decoherence, quantum error correction algorithms are needed. Different types of error correction codes have been developed [46], such as surface codes [47] and color codes [48]. Another open problem about noise is correcting decoherence between entangled states. It has been shown that physical purification of quantum states [49] can be used to effectively correct multi-qubit states. However, this

process can only be applied to well-understood states like the Bell states.

- Hardware size: a quantum processor could have a size similar to a coin. However, the cryostat hardware required to provide a suitable environment for this processor is larger than a person [32]. There are multiple components to preserve quantum states, a vacuum chamber is needed, so a device to pump out the air is also required. Portals to the chamber are also needed to allow light sources like lasers. In order to retain the chamber in a cryogenic environment, specific materials are required to reduce the temperature, like liquid helium. Plenty of equipment is also required to control the qubits. Regarding their size, the situation of current quantum computers is similar to the early stages of classical computers that occupied a room [50, 51]. The cost for a single qubit in this age is around 10000$ by most estimates [52]. This cost should be greatly reduced in order to commercialize quantum computers, as the number of qubits still needs to be increased significantly compared to current quantum computers. For the development of a quantum Internet, lowering these costs and scaling up the connectivity is also a task of great importance [53].

- Design complexity: duplicating arbitrary qubits is impossible due to the non-cloning theorem. This fact produces inconveniences in algorithm designs and implementations because recovering lost data is difficult since there is no copy of it. We can recreate a state if we know its amplitudes, as the non-cloning theorem only forbids cloning arbitrary unknown states. However, no information can be obtained about an unknown state without measuring it and this measurement destroys its amplitude distribution. Even if it is measured, only one possible outcome is known. There are no redundant states that allow us to repeat measurements in order to recreate the amplitude distribution. Thus, traditional methods like creating redundancy and retransmission cannot simply be applied to improve system robustness and design algorithms [53]. This increases the complexity of designing quantum software and hardware. In addition, quantum programming differs significantly from classical programming due to the different computational models. Companies such as IBM, Google and Microsoft have developed programming toolkits, such as Qiskit [54] and Cirq [55]. Quantum programming is usually performed on a classical simulator. Once a program is completed, it is then uploaded to a real quantum computer. Then, the program is run a large number of times. The results show the distribution that the program generates. Still, different hardware can produce different distributions due to decoherence and noise. Despite all the difficulties, quantum programming and debugging software are being diligently investigated and developed [56].

Now that we have provided a brief introduction to quantum computation, we can move on to quantum communication, which is another active area of research that has gained much interest in recent decades.

1.2 Quantum communication

Quantum communication can be defined as the transfer of quantum information from one place to another [57]. Qubits are used for data transmission instead of classical bits. Quantum communication primarily focuses on applications such as quantum cryptography, quantum teleportation, quantum networks, and other related areas [58–63].

Quantum cryptography is one of the main branches of quantum communications. It is based on transforming plaintext data into encrypted data in order to preserve its security, taking advantage of the properties of quantum mechanics. Due to Shor's algorithm [6], the security of classical cryptographic protocols like RSA has been called into question. Since these classical protocols will become vulnerable to quantum attacks in the future, quantum cryptography has gained a lot of attention as a protection to these threats. The main advantage that quantum cryptography offers compared to the classical protocols is the promise of unconditional security, so that data is protected based on the fundamental laws of quantum mechanics. Thus, no attacker can break the protocol, independently of the resources used. Different security proofs of these protocols have been demonstrated [64, 65].

One of the most researched branches of quantum cryptography is quantum key distribution (QKD). This set of protocols allows two parties to generate a shared random secret key that is known only by them. This key can then be used to encrypt and decrypt messages. The first QKD protocol was proposed by Charles H. Bennett and Gilles Brassard in 1984 [66]. This protocol became the well known Bennett-Brassard 1984 (BB84) protocol. The original proposal did not originally receive that much attention but this changed five years later, when the protocol was implemented in practice [67]. Even though this was a crude proof of concept lab implementation over a distance of just 32.5 cm, it greatly increased the interest in the field. Both Bennett and Brassard also worked in other relevant problems regarding QKD, like their work on secret key post-processing. In particular, they studied the set of classical procedures required to transform the raw measurements obtained from the quantum signals into a functional secret key using information reconciliation and privacy amplification [68, 69].

Soon after, different experiments were performed that kept improving the field solidly. The distance improved from a just a handful of kilometres to a few hundreds out of the lab in field installations that used optical fiber [70]. The feasibility of the technology was demonstrated in networks using the DARPA network located in Boston [71]. In 2008, a breakthrough network was designed and implemented in Vienna by the European project SEcure COmmunication based on Quantum Cryptography (SECOQC) [72]. This project combined many different types of QKD links. A similar secure communication network with QKD in the metropolitan area of Tokyo was reported in 2010 [73]. Six different QKD systems were integrated into the network. They also demonstrated the world-first secure TV conference over a distance of 45 km using QKD links. In 2015, a QKD system capable of distributing provably secure cryptographic keys over 307 km

of optical fibre was presented [74]. In 2018, a QKD network was deployed in Madrid in collaboration with the telecommunications operator Telefónica. This network is managed through a Software Defined Networking (SDN) structure that integrates quantum and classical channels [75,76]. A variant of QKD known as twin-field QKD, which can achieve a longer secure distance than traditional QKD protocols has also been implemented experimentally. In 2022, a twin-field QKD was implemented over a secure distance of 833.8 km [77]. Then in 2023, another experimental twin-field QKD implementation managed to improve its distance to 1002 km [78]. The utilization of drones to improve free-space QKD infrastructures has been proposed [79]. Another variant of QKD known as continuous variable QKD (CV-QKD), which is based on encoding information in optical coherent states that can be transmitted by fiber using devices that are commonly employed in the current telecommunications industry [80], has been studied in depth too. Its main virtues are cost-effectiveness and ease of implementation. An issue that arises in CV-QKD is achieving accurate frequency synchronization, a problem commonly known as frequency locking. To address this issue, a pilot-tone-assisted frequency-locking algorithm was proposed to ensure phase and frequency coherence in CV-QKD systems [81]. This algorithm was validated using both simulations and experiments. Besides this, CV-QKD systems are extremely sensitive to optical and electronic impairments, since these can notably decrease the secret key rate. To address this issue, a CV-QKD system was modelled in order to simulate the adverse effects of individual impairments on the secret key [82]. These model and simulations serve as a tool for the initial calibration of a CV-QKD system, preventing the decrease of the secret key rate caused by commonly known experimental impairments. Recently this year, researchers from China and South Africa developed a quantum microsatellite capable of performing space-to-ground QKD making use of portable ground stations [83]. Using this set-up, they achieved the sharing of up to 1.07 million bits of secure keys during a single satellite pass. Furthermore, a secret key was created that enabled one-time pad encryption of images between China and South Africa at locations separated by over 12900 km on Earth.

Due to the rapid progress that the field has experimented, some commercial successes have been developed. Currently, there are several vendors of QKD equipment such as ID Quantique, Quintessence or Qaskey [70]. Besides this, many major companies and laboratories around the world have displayed equipment or are actively developing the technology, like Toshiba, Huawei, NEC, etc.

Besides the original BB84 protocol, more protocols have been developed. In 1991, a QKD protocol based on entanglement was published by Arthur Ekert, which is known as E91 [84]. These two protocols are the most famous, but other derivatives have been created since their invention, such as B92 [85], BBM92 [86], SARG04 [87], measurement-device-independent QKD (MDI-QKD) [88], twin-field QKD (TF-QKD) [89], etc.

We now proceed to summarize the key features of the current quantum cryptography applications [33]. As we briefly mentioned earlier, the main advantage they offer is in regards to security, as they are based on the laws of quantum mechanics and not on

the particular complexity of solving a certain mathematical problem like the classical protocols. Also, quantum mechanics cause the transmission of the quantum state to be sensitive. This means that attempting to tamper with the transmitted state will be noticed. In the case of QKD, it assures the detection of an eavesdropper. On the other hand, the current limitations of quantum networks negatively impact the development of quantum cryptography. In the case of QKD, the typical key generation rate of QKD networks is still inefficient compared to classical methods. Even though it is being improved steadily [90], the typical key generation rate of QKD networks is at the scale of Mbit/s whereas classical optical communications usually provide about 100 Gbit/s per wavelength channel [91]. Moreover the key efficiency, which describes how many of the original bit strings are preserved after the key generation, is worsened by decoherence since noises are introduced during quantum state transmission. Finally, as we have mentioned, since QKD is relatively simple to implement and is mature enough for real applications, QKD services have been commercialized successfully by different companies and research institutes.

In spite of all the progress that the field of quantum cryptography has experimented, as demonstrated by the commercialization of QKD services, there still are challenges to overcome in the future. These are [33]:

- Key rate: the key rate of QKD depends on the performance of the particular hardware used. Optical fibres generate less noise compared to transmissions performed in free space, like those achieved using satellites. On the other hand, transmission in free space can produce better photon quality than optical fibres. These parameters are really important, since if the communication between two parties in a QKD protocol is not stable and correct measurements are not guaranteed, the parties would wrongly believe that there is an intruder and would abandon a secure channel.

- Denial of service: If the parties in a QKD protocol presume that the communication channel is insecure, they would stop using it and switch to a different available quantum channel, assuming it exits. This means that denial-of-service attacks are possible in QKD. Attacks based on this strategy and strategies to defend against it have been proposed [92].

- Key efficiency: the key efficiency represents the proportion of the original bit strings that are preserved after key generation. It can be calculated dividing the length of the secret key by the length of the original bit string and it is related to the amount of time required to generate a key of fixed length. The length of the secret key should be long enough in order to prevent search attacks that rely on brute force. If the key efficiency is low, this indicates that the original bit string is long and that key generation requires a long time. QKD protocols typically have low key efficiency. As an example in the BB84 protocol [66], the sender and the receiver measure the same set of quantum states and retain those that have

the same measurement results, which are then mapped to the secret key. Since they choose between two measurement bases, they have about a 50% probability of selecting the same basis and then obtaining the same measurement results. Thus, about 50% of the quantum states are used to indicate the secret key, which is low compared to classical approaches.

Even though QKD is the most famous application of quantum cryptography, there exist many others, such as randomness generation, delegated quantum computation, secure two- and multi-party computation and quantum money [93]. Besides this, classical cryptographical alternatives known as post-quantum cryptography have been developed. These protocols are believed to be resistant to quantum attacks, taking advantage of computational problems that are considered too hard to solve even using a quantum computer [94]. They seem to be a great alternative to solve the security problem classically. For example, lattice-based cryptography [95, 96] is one of the possible alternatives to current classical cryptographic protocols based on factorization. It constructs its cryptographic primitives using lattices and it is assumed to be immune to both classical and quantum attacks [94]. Other proposals for post quantum cryptography are code-based [97], hash-based [98] and based on multivariate signatures [99]. Still, post-quantum cryptography is only immune to known quantum attacks, so new quantum algorithms developed in the future could pose problems to them.

To conclude this chapter, we want to mention two applications useful for delegated quantum computation that serve as a nexus between quantum computation and quantum communication: blind quantum computation (BQC) and quantum homomorphic encryption (QHE). Delegated quantum computation refers to the scenario in which a client with limited quantum resources delegates a task to a more powerful quantum server. BQC allows a client to delegate a quantum computing task to a remote server making use of interactive protocols, where the input, computation and final result are only known by the client [100]. QHE is a cryptographic technology that allows a remote server to perform quantum operations on encrypted quantum data so that the underlying plaintext data remains hidden from the server. The main difference between BQC and QHE is the fact that in BQC the server does not know the quantum operations performed, whereas in QHE the server is aware of these quantum operations [101]. QHE is the main focus of this thesis and will be introduced in great detail in the next chapter.

Finally, we want to mention a recent breakthrough in distributed quantum computing. Using a photonic network interface, two separate quantum processors were linked to form a single fully connected quantum computer [102]. This enables computations to be distributed across the network. This scalable architecture is based on modules, each containing only a small number of trapped-ion qubits that are linked together using optical fibers. The transmission of data between them is done using photons. This allows qubits in separate modules to be entangled. The significance of this study is that it constitutes the first demonstration of quantum teleportation of logical gates. This process is known as gate teleportation and it is the basis of the QHE schemes used in

this thesis. Thus this experimental demonstration represents an important step towards developing efficient QHE schemes.

The remaining chapters of the thesis are organized as follows. In chapter 2 we present a brief introduction to QHE and describe a QHE scheme in great detail. In chapter 3 we review the Bernstein-Vazirani algorithm in its standard and recursive versions. Then we analyse the homomorphic implementation of its recursive version in chapter 4. In chapter 5 we review Grover's algorithm and then analyse its homomorphic implementation in chapter 6. Finally, we review Szegedy walks algorithms in their quantum and semi-classical versions in chapter 7 and study their homomorphic implementations in chapter 8.

2 Background of Homomorphic Encryption

2.1 Background

In the current internet age, plenty of users take advantage of the services offered by the cloud, uploading enormous quantities of private data for many different services like storage. Thus, preserving the security of any computation performed on the cloud is a task of great significance. Similarly to how modern cryptography is applied to preserve the security of communications, computations performed on the cloud should be protected utilizing cryptographic technology that preserves the data's security. In order to accomplish this, a promising technology that has been developed in recent years is homomorphic encryption. It allows a client to encrypt data and then send it to a server that performs operations on this encrypted data. Once the computations are completed, the server returns the data back to the client so it can be decrypted. Thus, the client receives the evaluated data while the server never obtains any information about the actual data in which it operated on. A graphical representation of a simple homomorphic encryption scenario is shown in figure 2.1.

The notion of fully homomorphic encryption, which was originally called privacy homomorphism, was already introduced in 1978 [103]. Classical homomorphic encryption can be classified into three types regarding the number of allowed operations on encrypted data: Partially Homomorphic Encryption (PHE), Somewhat Homomorphic Encryption (SWHE) and Fully Homomorphic Encryption (FHE). PHE admits just one type of operation to be performed an unlimited number of times, SWHE admits some types of operations that can be performed a limited number of times and FHE admits an unlimited number of operations for an unlimited number of times [104]. Some examples of PHE and SWHE schemes include the RSA cryptosystem by Rivest, Shamir and Adleman (1978) [105], ElGamal (1985) [106], Paillier [107] (1999) and Boneh, Goh and Nissim [108] (2005). The last scheme in particular was the first capable of performing two operations: an arbitrary number of additions and only one multiplication, then an arbitrary number of additions again [109].

The first classical FHE scheme was proposed by Gentry [111] in 2009. It takes advantage of a technique known as bootstrapping, an intermediate process that allows to refresh a ciphertext with large error to a new one with smaller error. This way, more computations can be performed. Besides this, Gentry provided a method for constructing a FHE scheme from a SWHE one. Since then, more classical schemes have been

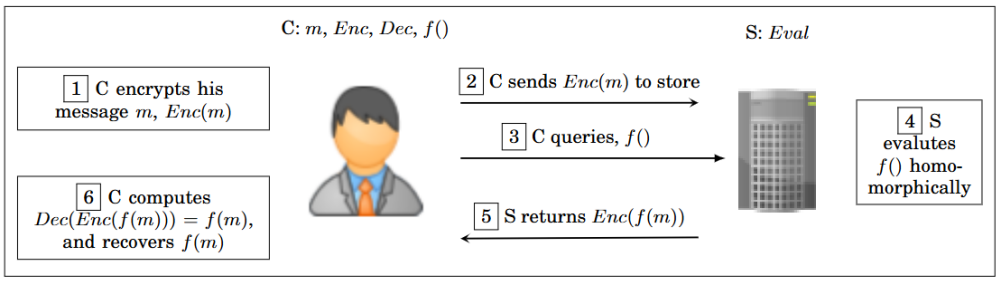


Figure 2.1: Homomorphic encryption scenario performed between a client and a server. Image obtained from [110].

proposed and perfected, such as BGV [112], FV [113], CKKS [114] and TFHE [115]. A extensive review about classical homomorphic encryption can be found in [109]. These schemes are computationally secure, which means that their security is based on the difficulty of solving complex mathematical problems, like ideal lattices [111] or the learning with errors problem (LWE) [112].

Regarding quantum computing and as we described in chapter 1, different companies have produced quantum computers that are available in the cloud, such as the IBM Q quantum computers. It is expected that most quantum computers will be accessible through the cloud in the near future. In order to preserve their security, different quantum homomorphic encryption schemes have been proposed. QHE allows a remote server to apply a quantum circuit, denoted by QC, on the encrypted quantum data $Enc(\rho)$ that a client has provided. Once the circuit is completed, the client decrypts the server's output and obtains the result of this quantum circuit, $QC(\rho)$. Contrary to the classical protocols, the security of QHE schemes can be based on the fundamental properties of quantum mechanics instead of the difficulty of solving complex mathematical problems.

QHE schemes can be classified as either interactive or non-interactive. If the client and server communicate during the execution of the circuit, the scheme is said to be interactive. If interactions are allowed the efficiency of the scheme decreases, since the server must wait until the client finishes some operations before it can resume the execution of the quantum circuit. For this reason interactive schemes are simpler to construct than non-interactive ones. As an example of these types of interactive protocols, a QHE scheme where the number of interactions between client and server is the same as the number of T-gates contained in the evaluated circuit was proposed in 2015 [116].

Nevertheless, we are more interested in studying non-interactive schemes. We provide some informal definitions about different properties of QHE schemes below. These will be expanded more rigorously in section 2.2.1. If a QHE scheme is $\mathcal{F}$-homomorphic then it can evaluate any quantum circuit homomorphically. A QHE scheme is said to be compact if the complexity of its decryption procedure does not depend on the evaluated circuit. Furthermore, for a scheme to be considered a quantum fully homomorphic encryption

(QFHE) scheme, it must be $\mathcal{F}$-homomorphic and compact. A scheme is perfectly secure if its security does not rely on the difficulty of solving complex mathematical problems, so the ciphertext does not convey any information about the contents of the plaintext.

The first proposals of QHE schemes appeared in 2012. A limited symmetric-key QHE scheme using the Boson sampling and multi-walker quantum walks was proposed by Rhode [117]. Since this scheme is not $\mathcal{F}$-homomorphic, it only allows the homomorphic evaluation of certain kinds of circuits. It was later implemented experimentally [118]. Liang [119] defined QFHE and proposed a weak scheme that allows local quantum computations to be performed on encrypted quantum data. However, since its evaluation algorithm depends on the secret key that the client possesses, it has a very limited application. A QHE scheme that allows a large class of quantum computations on encrypted data was proposed by Tan [120]. Yet, since it can only hide a constant fraction of the encrypted information, the scheme can not provide security in a cryptographic sense.

Questions about theoretical limits of QHE schemes have also been explored. A no-go result has been proven which states that any QFHE with perfect security must produce exponential storage overhead [121]. For this reason, constructing an efficient QFHE scheme with perfect security is impossible. An enhanced no-go result which states that constructing a QFHE scheme that is both information theoretically secure (ITS) and non-interactive is impossible was proved later [122]. Therefore, the best security a non-interactive QFHE scheme can achieve is computational security. Besides this no-go result, a compact, non-interactive ITS QHE scheme was constructed [122]. It is not $\mathcal{F}$-homomorphic due to their no-go result. This kind of scheme that is compact but not $\mathcal{F}$-homomorphic is called a quantum somewhat homomorphic encryption (QSHE) scheme.

Due to the no-go results mentioned no QHE scheme can have non-interaction, compactness, $\mathcal{F}$-homomorphism and perfect security simultaneously. Taking this into account, different schemes have been proposed with less strict conditions. For example, perfect security can be achieved if the QHE scheme is interactive, like in the one proposed by Liang [116]. Another approach is downgrading the security level from perfect security to computational security, like Broadbent and Jeffery [123], Dulek et al. [124] and Mahadev [125] did.

Two non-interactive quantum homomorphic schemes were proposed by Broadbent and Jeffery, combining quantum one-time pad (QOTP) and classical FHE. Their security and efficiency are limited by classical homomorphic encryption schemes. The first scheme they constructed was called EPR (named after Einstein, Podolski and Rosen [126]). Its properties are computational security, $\mathcal{F}$-homomorphism, non-interaction and quasi-compactness. A scheme is quasi-compact if the complexity of its decryption procedure scales sublinearly in the size of the evaluated circuit [123]. EPR is M^2-quasi-compact, where M is the number of $T/T^\dagger$ gates contained in the evaluated circuit. This means that the complexity of its decryption procedure scales with the square of the number of $T/T^\dagger$ gates contained in the evaluated circuit. The scheme proposed by Dulek [124] is

non-interactive too and relies on the implementation of an ancillary gadget which is based on classical FHE. For this reason, the scheme's security is also limited to computational security. Mahadev managed to construct a QFHE scheme based on the LWE problem [125], so it is built on classical FHE schemes and has computational security too.

Liang [127] constructed two non-interactive, perfectly secure and $\mathcal{F}$-homomorphic QHE schemes that are quasi-compact instead of compact. Both Broadbent's and Liang's schemes make use of quantum measurements and Bell states. Moreover, one of Liang's schemes called VGT, is M-quasi-compact, so it is superior in that regard compared to EPR. Since these schemes are quasi-compact, they do not contradict the no go-result that was previously presented.

Although they are quasi-compact, the main feature of Liang's schemes [127] is that they allow the efficient homomorphic evaluation of any quantum circuit with low $T/T^\dagger$-gate complexity with perfect data security. Due to this, they are suitable for circuits with a polynomial number of $T/T^\dagger$ gates. On the other hand, the decryption procedure would be inefficient for quantum circuits that contain an exponential number of $T/T^\dagger$ gates. These schemes have been used to implement a ciphertext retrieval scheme based on the Grover algorithm [128]. Also, a quantum ciphertext dimension reduction scheme for homomorphic encrypted data based on these schemes was constructed [129].

More QHE schemes and applications have been developed in recent years. The research of the homomorphic evaluation of non-Clifford gates has been focused on the $T/T^\dagger$ gates, so to improve this more non-Clifford gates have been studied. The homomorphic evaluations of the Clifford gates $\sqrt{X}$ (the square root of the X gate, called V in [130]), $\sqrt{X}^\dagger$, and controlled-Z gate have been given along the homomorphic evaluations of the following non-Clifford gates: controlled-$\sqrt{X}$ gate, controlled-$\sqrt{X}^\dagger$ gate and Toffoli gate [130]. Besides this, three QHE schemes for the single-qubit gates, the double-qubit gates, and the triple-qubit gate were respectively proposed [130]. A dynamic QFHE scheme based on universal quantum circuits was proposed to handle the volatility problem of the server [131], which refers to how a protocol should be aborted if a dynamic server wants to leave or join the protocol. To handle the situation in which many users want to begin computations at the same time or users do not want to trust just only one evaluator and prefer that a few evaluators collaborate to perform computations, a QHE scheme with a flexible number of evaluators based on (k, n)-threshold quantum state sharing was proposed [132]. In this scheme, the client can choose $d(k \leq d \leq n)$ evaluators in order to share encrypted states with them and cooperate to finish the desired operations. However, it can only perform some single-qubit unitary operations when $k \geq 2$. Also, the required quantum abilities of each evaluator can be a bit demanding. This scheme was later improved [133]. The improved version managed to extend the quantum operations that the evaluators can perform to include all the single-qubit unitary operations even when $k \geq 2$ and also managed to reduce the ability of at least $d-1$ evaluators at the expense of some flexibility. In order to solve the problem regarding the security of the quantum channel during the transmission process, in which the quantum

states may be attacked by eavesdroppers, a measurement- device-independent quantum homomorphic encryption (MDI-QHE) scheme has been proposed [134]. A two-round QHE scheme based on matrix decomposition and the circuit synthesis method was proposed [135]. In particular, matrix decomposition is used to construct the key updating function and obtain the corresponding decryption matrix. Then, the decryption matrix is decomposed into a quantum circuit making use of the circuit synthesis method. Based on the scheme in [128], an improved secure quantum homomorphic encryption ciphertext retrieval scheme was proposed [136]. In this new scheme, a trusted third party is introduced in order to cooperate with the server to execute the Grover algorithm. The trusted third party marks the solution on the plaintext state in each Grover iteration, then encrypts this marked state and sends it to the server. Next, the server performs the remaining operations of this Grover iteration on the encrypted state and returns the state to the trusted third party, who decrypts it and starts the whole process with the server once more until the whole algorithm is completed. At this point, the trusted third party encrypts the searched state and finally sends it to the client so they can decrypt and measure it to obtain the solution of the search problem. An efficient, error-correctable QHE scheme based on Calderbank–Shor–Steane codes has been recently developed [137]. This scheme takes advantage of the inherent properties of quantum error correction codes to perform encryption and error correction simultaneously. This eliminates the need for separate encoding steps, thus decreasing resource consumption. Lastly, we want to mention that a proof of concept implementation of a QHE scheme on a silicon photonic chip was experimentally demonstrated [138].

Liang's schemes are the basis of this thesis. We have applied them to the Bernstein-Vazirani algorithm, the Grover algorithm and Szegedy quantum and semiclassical walks algorithms. We proceed to review these schemes in this chapter.

2.2 Quantum Homomorphic Encryption scheme

2.2.1 Definitions

In this section, we review the rigorous definitions [123, 127] of some core concepts of QHE, such as compactness and homomorphism.

Definition 1 (Symmetric-key quantum homomorphic encryption) *A symmetric-key QHE scheme* `QHE` *contains the following four algorithms*

$$\texttt{QHE=(QHE.KeyGen,QHE.Enc,QHE.Eval,QHE.Dec)}.$$

1.Key Generation. $(sk, \rho_{evk}) \leftarrow \texttt{QHE.KeyGen}(1^n)$, *where sk is the secret key,* ρ_{evk} *is quantum evaluation key in* $D(\mathcal{H}_{evk})$. *The evaluation key is optional in symmetric QHE scheme.*

2.Encryption. $\texttt{QHE.Enc}_{sk} : D(\mathcal{H}_M) \to D(\mathcal{H}_C)$, *where* $D(\mathcal{H}_M)$ *and* $D(\mathcal{H}_C)$ *are the set of density operators in plaintext space and ciphertext space, respectively.*

3.Evaluation. $\texttt{QHE.Eval}^{QC} : D(\mathcal{H}_{evk} \otimes \mathcal{H}_C) \to D(\mathcal{H}_{C'} \otimes \mathcal{H}_{aux})$, *where* $\mathcal{H}_{C'}$ *is the result space of quantum computation on the space* $\mathcal{H}_C$*. For any quantum circuit* QC *(called evaluated circuit), with induced channel* $\Phi_{QC} : D(\mathcal{H}_M) \to D(\mathcal{H}_{M'})$*, we define a channel* $\texttt{Eval}^{QC}$ *that maps* $D(\mathcal{H}_C)$ *to* $D(\mathcal{H}_{C'})$ *with an additional auxiliary quantum state in space* $\mathcal{H}_{aux}$*. The evaluation key in* $D(\mathcal{H}_{evk})$ *is used up in the process.*

4.Decryption. $\texttt{QHE.Dec}_{sk} : D(\mathcal{H}_{C'} \otimes \mathcal{H}_{aux}) \to D(\mathcal{H}_{M'})$*. For any possible secret key* sk*,* $\texttt{Dec}_{sk}$ *is a quantum channel that maps ciphertext state together with auxiliary state to a plaintext state in* $D(\mathcal{H}_{M'})$*.*

Definition 2 (Compactness) *QHE scheme* QHE *is compact if the algorithm* QHE.Dec *is independent of the evaluated circuit* QC*.*

Definition 3 (Homomorphism) *Let* $\mathcal{L} = \{\mathcal{L}_\kappa\}_{\kappa\in\mathbb{N}}$ *be a class of quantum circuits. A quantum encryption scheme* QHE *is homomorphic for the class* $\mathcal{L}$ *if for any sequence of circuits* $\{C_\kappa \in \mathcal{L}_\kappa\}_{\kappa\in\mathbb{N}}$ *and input* $\rho \in D(\mathcal{H}_M)$*, there exists a negligible function negl such that:*

$$\Delta\left(\texttt{QHE.Dec}_{sk}\left(\texttt{QHE.Eval}^{C_\kappa}\left(\rho_{evk}, \texttt{QHE.Enc}_{sk}(\rho)\right)\right), \Phi_{C_\kappa}(\rho)\right) \leq negl(\kappa),$$

where $(sk, \rho_{evk}) \leftarrow \texttt{QHE.KeyGen}(1^\kappa)$ *and* Φ_{C_κ} *is the channel induced by quantum circuit* C_κ*.*

Given two quantum states ρ and σ, their trace distance used in definition 3 is defined as $\Delta(\rho, \sigma) = Tr(|\rho - \sigma|)$, where $|A|$ is given by $\sqrt{A^\dagger A}$.

Definition 4 (Quantum fully homomorphic encryption) *A QHE scheme is a quantum fully homomorphic encryption scheme if*

1. *it is compact and*
2. *it is* $\mathcal{F}$*-homomorphic (or homomorphic for* $\mathcal{F}$*), where* $\mathcal{F}$ *is the set of all quantum circuits over the universal quantum gate set* $\{X, Z, H, S, CNOT, T, T^\dagger\}$*.*

Naturally, any universal quantum gate set can be used in the previous definition instead of the set $\{X, Z, H, S, CNOT, T, T^\dagger\}$.

Definition 5 (Quasi-compactness) *Let* $\mathcal{L} = \{L_\kappa\}_{\kappa\in\mathbb{N}}$ *be the set of all quantum circuits over the universal quantum gate set* $\{X, Z, H, S, CNOT, T, T^\dagger\}$*. Let* $f : \mathcal{L} \to \mathbb{R}_{\geq 0}$

be some function on the circuits in $\mathcal{L}$. A QHE scheme is f-quasi-compact if there exists a polynomial p such that for any sequence of circuits $\{C_\kappa \in \mathcal{L}_\kappa\}_{\kappa\in\mathbb{N}}$ with induced channels $\Phi_{C_\kappa} : D(\mathcal{H}_M) \to D(\mathcal{H}_(M'))$, the circuit complexity of decrypting the output of $\texttt{QHE.Eval}^{C_\kappa}$ is at most $f(C_\kappa)p(\kappa)$.

Definition 6 (perfect security) *QHE scheme $\texttt{QHE}$ is perfectly secure if there exists a quantum state $\Omega^{\mathrm{A}'}$ such that for all states ρ^{AE} we have that:*

$$\| \texttt{QHE.Enc}(\rho^{\mathrm{AE}}) - \Omega^{\mathrm{A}'} \otimes \rho^{\mathrm{E}} \| = 0,$$

where $\texttt{QHE.Enc}$ is an encryption algorithm performed on the part A *of quantum state ρ^{AE}. Denote $\texttt{QHE.Enc}(\rho^{\mathrm{AE}})$ as a quantum ensemble over the probability distribution of the secret key and all the randomness in the quantum algorithm.*

2.2.2 Preliminaries

In the circuit model of quantum computation, quantum gates are the basic operations that compose any circuit that can be implemented [139, 140]. The usual Clifford gates, which are $\{X, Z, H, S, CNOT\}$, are used in the protocol. Gates X and Z are the single-qubit Pauli gates. The Hadamard gate is $H = \frac{X+Z}{\sqrt{2}}$. The matrix representation of these gates is given by:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{2.1}$$

The matrices for the phase gate S and the $CNOT$ gate are:

$$S = \sqrt{Z} = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \quad CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{2.2}$$

The $CNOT$ gate transforms a target qubit, t, according to the state of a control qubit c as:

$$CNOT|c\rangle|t\rangle = |c\rangle|t \oplus c\rangle, \forall c, t \in \{0, 1\}. \tag{2.3}$$

In order to perform universal quantum computation, a non-Clifford gate must be used along the Clifford gates. In this case, such gate is the T gate. Its conjugate is simply $T^\dagger$. Their matrix representation is:

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix} \quad T^\dagger = \begin{pmatrix} 1 & 0 \\ 0 & e^{-i\frac{\pi}{4}} \end{pmatrix}. \tag{2.4}$$

Then the complete set of gates used in Liang's schemes is $\mathcal{G} = \{X, Z, H, S, CNOT, T, T^\dagger\}$.

Regarding the homomorphic evaluation of a quantum circuit, the main issue is evaluating the T and $T^\dagger$ gates because these gates generate a S-error:

$$TX^aZ^b\left|\phi\right\rangle = (S^\dagger)^a X^a Z^{a\oplus b} T\left|\phi\right\rangle. \tag{2.5}$$

Thus, this error has to be corrected as efficiently as possible. In case of Liang's schemes, this error is corrected by making use of a generalization of quantum teleportation known as gate teleportation. This procedure is primarily used in measurement-based quantum computing [141, 142].

2.2.3 Gate teleportation

An EPR state is an entangled quantum state given by $\left|\Phi_{00}\right\rangle = \frac{1}{\sqrt{2}}(\left|00\right\rangle + \left|11\right\rangle)$. The state $\left|\Phi_{00}\right\rangle$ can be constructed by applying a H gate followed by a $CNOT$ gate, both acting on the state $\left|00\right\rangle$. From this entangled state, the well known four Bell states can be expressed in a compact expression:

$$\left|\Phi_{ab}\right\rangle = (Z^bX^a \otimes I)\left|\Phi_{00}\right\rangle, \forall a, b \in 0, 1. \tag{2.6}$$

Quantum teleportation [143] is a technique that allows the transferring of quantum states between a sender and a receiver, taking advantage of entanglement and a classical communication channel. In this procedure, Alice wants to send Bob a qubit in the state $\left|\psi\right\rangle$. To accomplish this, they first share an entangled pair of qubits in the state $\left|\Phi_{00}\right\rangle$. Thus, both have a qubit from the EPR pair. Then, Alice performs a quantum measurement in the Bell basis using the two qubits she possesses, the qubit in the state $\left|\psi\right\rangle$ and one qubit of the pair $\left|\Phi_{00}\right\rangle$. Due to entanglement, Bob can now recover the original state $\left|\psi\right\rangle$ applying the correct sequence of quantum gates to the qubit in his possession. These gates are a combination of X and Z gates, which Bob applies depending on the measurement results that Alice obtained.

For any single-qubit gate U, the "U-rotated Bell basis" can be defined as [142]: $\Phi(U) = \{\left|\Phi(U)_{ab}\right\rangle, a, b \in \{0, 1\}\}$, where $\left|\Phi(U)_{ab}\right\rangle = (U^\dagger \otimes I)\left|\Phi_{ab}\right\rangle = (U^\dagger Z^b X^a \otimes I)\left|\Phi_{00}\right\rangle$.

In case of a single qubit, we have the following expression for quantum teleportation:

$$\left|\alpha\right\rangle \otimes \left|\Phi_{00}\right\rangle = \sum_{a,b\in\{0,1\}} \left|\Phi_{ab}\right\rangle \otimes X^aZ^b\left|\alpha\right\rangle. \tag{2.7}$$

It can easily be extended for the "U-rotated Bell basis":

$$\left|\alpha\right\rangle \otimes \left|\Phi_{00}\right\rangle = \sum_{a,b\in\{0,1\}} \left|\Phi(U)_{ab}\right\rangle \otimes X^aZ^bU\left|\alpha\right\rangle. \tag{2.8}$$

where U is any single-qubit gate. Thus equation (2.8) describes gate teleportation. Similarly to the usual quantum teleportation protocol, Alice and Bob start by sharing

an entangled EPR pair in the state $|\Phi_{00}\rangle$. Then Alice prepares a qubit in the state $|\alpha\rangle$ and performs a "U-rotated Bell measurement". This is simply a quantum measurement in which the U-rotated Bell basis is selected as the measurement basis. Alice measures the two qubits she possesses, one in the state $|\alpha\rangle$ and one of the EPR pair in the state $|\Phi_{00}\rangle$. From this measurement Alice obtains the results a and b. After the measurement is completed, Bob's qubit transforms into the state $X^a Z^b U |\alpha\rangle$. Next, Alice tells Bob the results of her measurement, so Bob can apply the correct X and Z gates to finally obtain $U |\alpha\rangle$. The gate teleportation procedure is shown in figure 2.2.

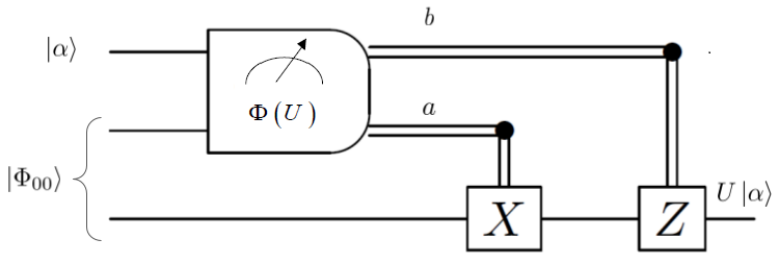


Figure 2.2: Quantum circuit for gate teleportation. The box represents the quantum measurement Alice performs on the qubit $|\alpha\rangle$ and one qubit of the pair $|\Phi_{00}\rangle$. The measurement basis selected is the U-rotated Bell basis $\Phi(U)$. Depending on the measurement results that Alice obtained, a and b, Bob applies X^a and Z^b in order to obtain $U |\alpha\rangle$.

Gate teleportation is clearly an extension of quantum teleportation, since if U were the identity then the U-rotated Bell basis $\Phi(U)$ would reduce to the standard Bell basis, and therefore gate teleportation would implement the standard quantum teleportation procedure. The basis of Liang's QHE schemes is this gate teleportation protocol, because it corrects the error that results from the homomorphic evaluation of a T gate.

2.2.4 Encryption, evaluation and decryption

Suppose that there is a quantum circuit composed of gates from the set $\mathcal{G} = \{X, Z, H, S, CNOT, T, T^\dagger\}$ that acts on n qubits. In total, there are l gates in the circuit. The quantum gates in the circuit are numbered starting from left to right. This way the first quantum gate is denoted by $G[1]$, the second by $G[2]$ and so on until the last gate, denoted by $G[l]$. The jth quantum gate is denoted by $G[j]$. The qubit it acts on is denoted by the subscript w. As an example the gate $G[j] = X_w$ represents a Pauli X-gate acting on the wth qubit of the circuit. The $CNOT$ gate acting on the wth control qubit and the w'th target qubit is denoted by $CNOT_{w,w'}$. Among the total number of gates l, the total number of T and $T^\dagger$ gates are denoted by M. Each T and $T^\dagger$ gate has its own number, j_i ($i \leq j_i \leq l$, $1 \leq i \leq M$) in the sequence of gates $\{G[j], j = 1, 2, .., l\}$. Then $G[j_i] = T/T^\dagger$ where $j_i < j_{i+1}$. If $G[j_i]$ ($1 \leq i \leq M$) is applied on the w_ith qubit

$(1 \leq w_i \leq n)$ then we can write $G[j_i] = T_{w_i}/T_{w_i}^{\dagger}$.

Liang proposed two schemes, one called GT and the other VGT. The first step in these schemes is encrypting the data that will be sent to the server. This is accomplished by applying a symmetric-key encryption scheme, known as quantum one time pad (QOTP), to the data. It is based on applying a combination of X^a and Z^b gates to each qubit. The bits a and b are randomly selected from $\{0,1\}$ and constitute the secret key of the client sk. If the plaintext data contains n qubits, the secret key sk requires $2n$ bits: $sk = (a_0, b_0)$, $a_0, b_0 \in \{0,1\}^n$. The wth plaintext qubit is encrypted using the secret bits $(a_0(w), b_0(w))$ such that: $|\alpha\rangle_w \rightarrow X^{a_0(w)} Z^{b_0(w)} |\alpha\rangle_w = |\rho_0\rangle_w$ where $|\rho_0\rangle_w$ represents the encrypted state of the wth qubit.

QOTP was proposed by Boykin and Roychowdhury [144]. They proved that if the bits a and b are randomly selected from $\{0,1\}$ and used only once, QOTP has perfect security. The reason for this is that if QOTP is applied to any arbitrary quantum state σ, the totally mixed state $\frac{I_{2^n}}{2^n}$ is obtained:

$$\frac{1}{2^{2n}} \sum_{a,b \in \{0,1\}^n} \left[\bigotimes_{i=1}^{n} X_i^{a(i)} Z_i^{b(i)} \right] \rho \left[\bigotimes_{i=1}^{n} X_i^{a(i)} Z_i^{b(i)} \right]^{\dagger} = \frac{I_{2^n}}{2^n}. \tag{2.9}$$

Next, the encrypted data will be sent to the server. In order to complete its designated quantum circuit, the server applies quantum gates from the set $\mathcal{G} = \{X, Z, H, S, CNOT, T, T^{\dagger}\}$. As long as the gates performed are Clifford and not T or $T^{\dagger}$, the homomorphic evaluation can be performed straightforwardly. Each time the server performs one of these gates on a qubit, the key is updated according to the algorithm shown in figure 2.3. After the jth $(1 \leq j \leq l-1)$ gate is performed, a new key (denoted by $(a_j, b_j), a_j, b_j \in \{0,1\}^n$) is obtained through key updating. This is the intermediate key. As an example of this process, the new key that would be obtained after applying a H-gate is $(a_1, b_1) = (b_0, a_0)$ assuming it was the first gate in the circuit so $j = 1$.

If the gate applied is a $T/T^{\dagger}$, its homomorphic evaluation is performed using the gate teleportation procedure explained in section 2.2.3. In this case, the S^a-rotated Bell measurement is used in order to correct the S-error described previously in Eq. (2.5). At the beginning of the circuit's evaluation an EPR source generates M Bell states, as many as $T/T^{\dagger}$-gates are in the circuit, denoted by $\{|\Phi_{00}\rangle_{c_i,s_i}, i = 1, \ldots, M\}$, where qubits c_i, $i = 1, \ldots, M$ and qubits s_i, $i = 1, \ldots, M$ are kept by the client and server respectively. If the server has to evaluate a $T/T^{\dagger}$-gate, it first applies this gate to the desired qubit, then performs a SWAP gate between this encrypted qubit and one of the entangled qubits that the server possesses, s_i. The next step is applying a S^a-rotated Bell measurement on the qubits s_i and c_i. This measurement is performed by the client. Using the results obtained from this measurement, r_a and r_b, the encryption key can be updated. In particular, from $TX^aZ^b |\alpha\rangle$,this whole process returns $X^{a \oplus r_a} Z^{a \oplus b \oplus r_b} T |\alpha\rangle$. When the evaluated gate is a $T^{\dagger}$-gate, the key will be updated from $T^{\dagger} X^a Z^b |\alpha\rangle$ to $X^{a \oplus r_a} Z^{b \oplus r_b} T^{\dagger} |\alpha\rangle$. This whole process is shown in figure 2.4 for a T gate. It will be

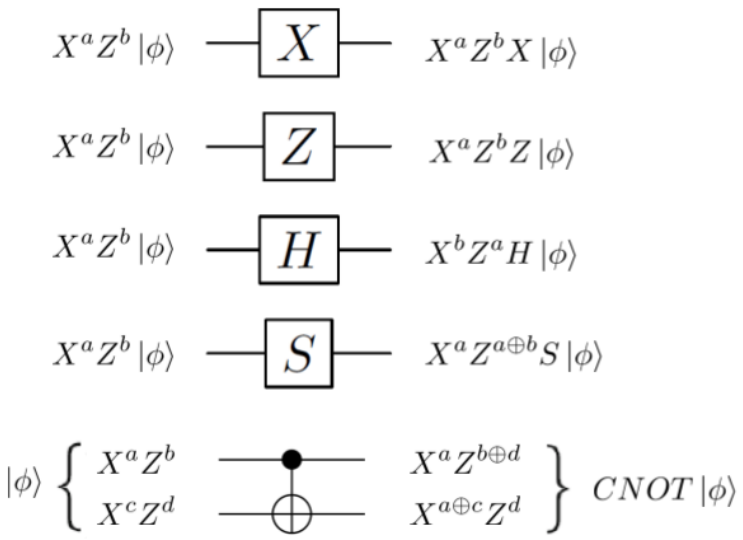


Figure 2.3: Key updating rules for the homomorphic evaluation of Clifford gates.

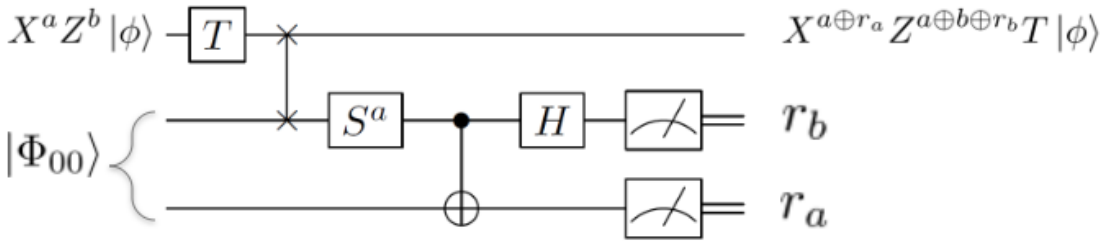


Figure 2.4: Homomorphic evaluation of T-gate. $|\Phi_{00}\rangle$ represents an EPR pair.

performed M times, as many as the number of $T/T^\dagger$-gates contained in the circuit.

These M measurements can be postponed until the server has completed all the quantum operations due to the principle of deferred measurement. Also, the server must generate the key-updating functions according to the key-updating rules of each gate from the set $\mathcal{G}$ that have already been explained. The key-updating functions are constructed differently in the schemes GT and VGT. VGT is more optimal than GT, since the latter is $M\log(M)$-quasi-compact whereas the former is M-quasi-compact. As VGT is the more optimal scheme, we focus on this one from this point onwards. For the scheme VGT, the server must generate $2M+1$ key-updating functions, $\{g_i\}_{i=1}^{M}$ and $\{f_i\}_{i=1}^{M+1}$. Once the quantum circuit has been completed, the server sends all the encrypted qubits, the key-updating functions based on the key-updating rules and all the ancillary s_i qubits to the client. Then the client updates its keys according to the key-updating functions and performs a S^a-rotated Bell measurement on qubits c_i and s_i. Since to perform the

correct S^a-rotated Bell measurement the updated a key is needed, the client has to alternate between updating the keys and measuring. These measurements must be performed in the pre-established order, as the updated key depends on the result of the previous measurement. Once all measurements are completed and all the keys are updated, the client obtains the final key $dk = (a_{\text{final}}, b_{\text{final}})$, $a_{\text{final}}, b_{\text{final}} \in \{0,1\}^n$. Finally the client decrypts the qubits that are in the state $|\rho_{\text{final}}\rangle$, which is the final state that the server outputs, making use of the final key to obtain the plaintext results in the state $|\alpha_{\text{final}}\rangle$: $X^{a_{\text{final}}} Z^{b_{\text{final}}} |\rho_{\text{final}}\rangle = |\alpha_{\text{final}}\rangle$.

The whole QHE scheme can be described in five steps: **Setup**, **Key Generation**, **Encryption**, **Evaluation**, **Decryption**.

1. **Setup**: an EPR source generates M Bell states, as many as $T/T^\dagger$-gates are in the circuit, $\{|\Phi_{00}\rangle_{c_i,s_i}, i = 1,\ldots,M\}$, where qubits $c_i, i = 1,\ldots,M$ and qubits $s_i, i = 1,\ldots,M$ are kept by the client and the server respectively.

2. **Key Generation**: Generate random bits a_0, $b_0 \in \{0,1\}^n$ and output the secret key $sk = (a_0, b_0)$.

3. **Encryption**: for any n qubit data $|\alpha\rangle$, client performs QOTP encryption making use of the secret key $sk = (a_0, b_0)$: $|\alpha\rangle \to X^{a_0} Z^{b_0} |\alpha\rangle = |\rho_0\rangle$.

4. **Evaluation**: The server applies the quantum gates $G[1]$, $G[2]$,..., $G[l]$ in order on the n encrypted qubits. For each $j \in \{1,\ldots,l\}$ there are two possible cases: if $j \notin \{j_1,\ldots,j_M\}$ then $G[j]$ is a Clifford gate and the server applies it. If $j = j_i$ $(1, \leq i \leq M)$ then the gate $G[j] = G[j_i] = T_{w_i}$ or $T^\dagger_{w_i}$, the server applies this gate $G[j]$ on the qubit w_i and then the server performs a SWAP gate on this w_ith qubit and one of the entangled qubits s_i. Taking $j_0 = 0$ and using the key-updating rules the server generates the polynomial $\{g_i\}_{i=1}^M$ for one key bit $a_{j_i-1}(w_i) = g_i(a_{j_{i-1}}, b_{j_{i-1}})$, $i = 1,\ldots,M$, with $a_{j_i-1}(w_i) \in \{0,1\}$. Likewise according to the key updating rules the server generates the polynomial $\{f_i\}_{i=1}^M$ for the intermediate key $(a_{j_i}, b_{j_i}) = f_i(a_{j_{i-1}}, b_{j_{i-1}}, r_a(i), r_b(i))$, $i = 1,\ldots,M$, with $(a_{j_i}, b_{j_i}) \in \{0,1\}^{2n}$. The final polynomial that the server generates is f_{M+1} for the final key $(a_{\text{final}}, b_{\text{final}}) = f_{M+1}(a_{j_M}, b_{j_M})$, with $(a_{\text{final}}, b_{\text{final}}) \in \{0,1\}^{2n}$. After the last gate is applied the server sends all the encrypted qubits, the key-updating functions and all the ancillary s_i qubits to the client.

5. **Decryption**: The client alternates between key updating and performing measurements. For each $i = 1,\ldots,M$, the client computes g_i and obtains the corresponding a for the ith S^a-rotated Bell measurement: according to the key $(a_{j_{i-1}}, b_{j_{i-1}})$ and the key-updating function g_i, the client obtains $a = g_i(a_{j_{i-1}}, b_{j_{i-1}})$. If $i = 1$, the secret key $sk = (a_0, b_0)$ is $(a_{j_{i-1}}, b_{j_{i-1}})$. Then using this measurement basis, the client performs a S^a-rotated Bell measurement on qubits c_i and s_i and obtains the measurement results $r_a(i)$ and $r_b(i)$. Then the client computes the intermediate key (a_{j_i}, b_{j_i}) according to the key updating function f_i:

$(a_{j_i}, b_{j_i}) = f_i(a_{j_{i-1}}, b_{j_{i-1}}, r_a(i), r_b(i))$. After the last round, $i = M$, of this process is performed the client obtains the intermediate key (a_{j_M}, b_{j_M}). Making use of this key and the last key-updating function f_{M+1} the client obtains the final key: $(a_{\text{final}}, b_{\text{final}})$. Finally the client performs QOTP decryption on the encrypted qubits to obtain the final states: $X^{a_{\text{final}}} Z^{b_{\text{final}}} |\rho_{\text{final}}\rangle = |\alpha_{\text{final}}\rangle$.

A remark about the setup step in which the client and server share M Bell states is in order. Actually, it is not necessary for them to pre-share the Bell states. Instead, a Bell state can be created by the server each time it evaluates a $T/T^\dagger$-gate. Then the server can generate the M Bell states required and send them back to the client along with all the encrypted qubits and the key-updating functions.

Therefore, this scheme can evaluate any quantum circuit homomorphically (making it $\mathcal{F}$-homomorphic) with perfect security. The server can never learn any information about the plaintext or the evaluation keys at any point of the scheme. The data the server receives is encrypted using QOTP so it is perfectly secure and the secret key sk used by the client is hidden perfectly too. There are no interactions between the client and the server during the evaluation process, so the server can not obtain any information about the plaintext or the secret key in that step either. Finally it is impossible to obtain any information once the data is sent back to the client for the decryption process, since the client performs quantum measurements and key-updating locally and does not interact with the server. At the last step, the client uses the final key to perform QOTP decryption which is perfectly secure, concluding the whole scheme.

Now that the whole scheme has been described, the reason why it is quasi-compact becomes clear. Unlike Clifford gates, the number of $T/T^\dagger$-gates contained in the evaluated circuit make the complexity of the decryption process grow due to the quantum measurements the client has to perform in succession. Thus, the efficiency of the scheme depends on the number of $T/T^\dagger$-gates. For this reason, the scheme we have described is only suitable for circuits with a polynomial number of $T/T^\dagger$-gates.

In the following chapters, we will apply this QHE scheme to different quantum algorithms and study their $T/T^\dagger$-gate complexity in order to analyse the efficiency of their QHE implementations.

3 The Bernstein-Vazirani (BV) Algorithm

3.1 Background

The algorithms proposed by Deutsch [2] and Deutsch-Jozsa [145] provided the first clues about the possible benefits of using quantum computers. After the introduction of these algorithms, Bernstein and Vazirani introduced a problem in which the query complexity of the quantum solution was better than what any classical computer could achieve. The Bernstein-Vazirani (BV) [4] algorithm is the first quantum algorithm that revealed the advantages that a quantum computer could have compared to a classical one. The nonrecursive version of the problem managed to improve the query complexity from $O(n)$, which was the best a classical computer could accomplish, to $O(1)$ for the case of the quantum computer. This improvement constitutes a polynomial speed-up. Due to its significance, this algorithm has been thoroughly studied. The algorithm was tested experimentally using an optical implementation of the circuit [146]. A probabilistic version of the BV problem and a quantum algorithm that solves it have been proposed that involves finding a certain number of secret keys from a set of multiple secret keys [147].

It has also been used as the basis of many different applications. One application is a quantum algorithm that finds the linear structures of a Boolean function [148]. In the realm of cryptography, the previous work was used as the basis of an algorithm that finds the linear structures of a vector function in order to attack block ciphers [149]. The BV algorithm was also used as a subroutine in a quantum algorithm for recovering the key of block ciphers in the quantum related-key attack model. [150]. Recently, the query complexity of the algorithms presented in [149] were improved further [151]. The BV and Grover algorithms were combined to generate quantum key-recovery attacks on different rounds of Feistel constructions [152]. A variant of the algorithm was studied for quantum learning robust against noise [153].

The nonrecursive BV algorithm is classically tractable because the complexity of the classical problem scales just polynomially in the problem size. Bernstein and Vazirani managed to produce a version of the problem which is classically intractable, but can be solved on a quantum computer. This version of the problem is called the recursive Bernstein-Vazirani problem. It possesses a superpolynomial query complexity in the classical realm but only a polynomial query complexity in the corresponding quantum version. Thus the recursive BV algorithm demonstrated a superpolynomial speed-up

compared to the best classical algorithms [154]. This algorithm was the first to show such a speed-up and for this reason it holds a distinguished position in the history of quantum computing. In this chapter we review both versions of the BV algorithm.

3.2 Bernstein-Vazirani algorithm review

In the Bernstein-Vazirani (BV) problem a n bit function $f : \{0,1\}^n \to \{0,1\}$ is given. This function satisfies the relation: $f_s(x) = s \cdot x = x_1 s_1 + x_2 s_2 + ... + x_n s_n \mod (2)$ where s is an unknown n bit string and the objective is finding s. The best classical algorithm that can be applied to solve this problem in the exact query complexity model takes n queries since the function only returns one bit, so the best algorithm is simply using as input a bit string where all bits are 0 except the ith bit which would be a 1 in order to retrieve s_i from the function. If bounded error is allowed, suppose that the function could be queried in a probabilistic manner so all of the bits of s could be obtained in less than n queries with some probability of failure that is bounded below 1/2. Still, this bounded error algorithm would need $\Omega(n)$ queries [154]. This contrasts with the Deutsch-Jozsa algorithm since a classical probabilistic algorithm makes the quantum advantage disappear.

In regards to the quantum algorithm, the main idea is implementing $f_s(x)$ in a reversible manner using a quantum oracle:

$$U_s = \sum_{x\in\{0,1\}^n} \sum_{y\in\{0,1\}} |x\rangle \langle x| \otimes |y \oplus (s \cdot x)\rangle \langle y| . \tag{3.1}$$

The quantum algorithm uses n qubits, all starting in the state $|0\rangle$. The first step is applying n H gates, one to each qubit. It results in an equal superposition of all possible n-bit strings:

$$H^{\otimes n} |00\ldots0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle . \tag{3.2}$$

This state serves as the input for the oracle. The oracle is implemented using an extra qubit in the state $|-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$. If a $CNOT$ gate is applied to the state $|-\rangle$ as the target qubit, a phase kickback occurs: the control qubit changes its sign and the target qubit remains unchanged. This is because if a $CNOT$ gate is applied to the state $|-\rangle$ with $|1\rangle$ as the control qubit: $CNOT[|1\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}] = |1\rangle \otimes \frac{|1\rangle - |0\rangle}{\sqrt{2}} = |1\rangle \otimes (-\frac{|0\rangle - |1\rangle}{\sqrt{2}}) = -|1\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}$. This means the phase of the terms where $s \cdot x = 1$ will be flipped. Thus after applying U_s:

$$|\phi\rangle_s = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{f_s(x)} |x\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{s\cdot x} |x\rangle . \tag{3.3}$$

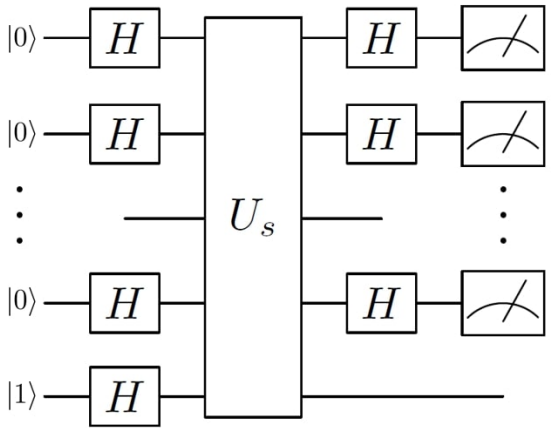


Figure 3.1: Nonrecursive Bernstein-Vazirani circuit. A layer of H gates is applied, followed by the application of U_s and another layer of H gates.

The action of the H gate on a general computational basis state of a n-qubit system is given by:

$$H^{\otimes n} |u\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{u \cdot x} |x\rangle , \tag{3.4}$$

and also the H gate is its own inverse:

$$H^{\otimes n} \left[\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{u \cdot x} |x\rangle \right] = |u\rangle . \tag{3.5}$$

Then if we apply n H gates once more, according to equation (3.5) the qubits will be in the state $|s\rangle$:

$$H^{\otimes n} |\phi\rangle_s = H^{\otimes n} \left[\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{s \cdot x} |x\rangle \right] = |s\rangle . \tag{3.6}$$

Finally after measuring the qubits, s is obtained with a 100% probability. Therefore the algorithm finds the value of s in just one query. This constitutes a polynomial speed-up compared to the classical case as we mentioned. The circuit that solves the BV problem is shown in figure 3.1.

We now move on to explain the recursive BV problem. In the nonrecursive BV problem, a function $f_s(x) = s \cdot x$ is given and s has to be obtained. In comparison in the recursive BV problem, what has to be obtained is some function $g : \{0,1\}^n \to \{0,1\}$ on s, $g(s)$. We refer to all the possible n bits strings that can be used as inputs to the function g by $s_{in} \in \{0,1\}^n$.

In order to construct the first level of the recursion, an oracle based on two n bits strings $x \in \{0,1\}^n$ and $y \in \{0,1\}^n$ is applied. Similarly to the nonrecursive BV algorithm the function queried takes these two bit strings as input and outputs $s_x \cdot y$, hence the function can be expressed as $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ given by $f(x,y) = s_x \cdot y$. The s_x are 2^n different bit strings labelled by $x \in \{0,1\}^n$ with one important property: the value of $g(s_x)$ must coincide with the product of $s \cdot x$ for some unknown s. The two level BV problem is then identifying this s and $g(s)$.

When $f(x,y)$ is queried, $s_x \cdot y$ is obtained. In order to find s_x for a fixed x, the original nonrecursive problem would have to be solved. Then after s_x has been found for a fixed x, s_x has to be used to calculate $g(s_x)$. However each $g(s_x)$ for different values of x is now part of a nonrecursive BV problem. This way the problem becomes more complex than the original nonrecursive version.

This process used to create the two level BV problem can be done k times to generate the k level BV problem. At the kth level the function f takes as input k strings of n bits denoted by x_i and calculates $f(x_1, x_2, \ldots, x_k) = x_k \cdot s_{x_1,x_2,\ldots,x_{k-1}}$. The secret bit string $s_{x_1,x_2,\ldots,x_{k-1}} \in \{0,1\}^n$ generates the next lower level problem since $g(s_{x_1,x_2,\ldots,x_{k-1}}) = x_{k-1} \cdot s_{x_1,x_2,\ldots,x_{k-2}}$ where like before $s_{x_1,x_2,\ldots,x_{k-2}} \in \{0,1\}^n$. This process can be continued until the final level is reached where $g(s_{x_1}) = s \cdot x_1$. Finally at this level s can finally be found.

From this point onwards, we will be discussing the recursive Bernstein-Vazirani problem for $k = 2$. We proceed to solve the problem for an arbitrary case in $k = 2$. As in the nonrecursive BV problem, we are given access to a unitary that computes $f(x_1, x_2) = s_{x_1} \cdot x_2$ in a reversible manner. If the usual BV algorithm is used, the value of a s_{x_1} for a fixed x_1 would be easy to obtain. The tricky part of the recursive BV problem is that when g is applied to these s_{x_1}'s we get a new BV problem that has to be solved. This implies that we have to use the BV algorithm over all x_1's too.

The oracle for the two level problem is:

$$U_s = \sum_{x_1,x_2 \in \{0,1\}^n} \sum_{y \in \{0,1\}} |x_1\rangle_1 \langle x_1|_1 \otimes |x_2\rangle_2 \langle x_2|_2 \otimes |y \oplus (s_{x_1} \cdot x_2)\rangle \langle y| \,. \tag{3.7}$$

If we start with two quantum registers, x_1 and x_2 with all their qubits in the state $|0\rangle$ and apply H gates to both of them we obtain the usual state that contains the superposition over all possible bitstrings on the x_1 and x_2 registers:

$$H^{\otimes 2n} |00...0\rangle = \frac{1}{2^n} \sum_{x_1,x_2 \in \{0,1\}^n} |x_1\rangle_1 \otimes |x_2\rangle_2 \,. \tag{3.8}$$

The two registers are labelled by a 1 and 2 subscript respectively. Applying the usual phase kick-back trick using a qubit in the state $|-\rangle$ as the target qubit, we can implement

the oracle U_s. After applying it we then have the following state:

$$\frac{1}{2^n}\sum_{x_1,x_2\in\{0,1\}^n}(-1)^{s_{x_1}\cdot x_2}\left|x_1\right\rangle_1\otimes\left|x_2\right\rangle_2\otimes\left|-\right\rangle. \tag{3.9}$$

By applying n Hadamard gates on the qubits of the second register, we see that they transform this state to:

$$\frac{1}{\sqrt{2^n}}\sum_{x_1\in\{0,1\}^n}\left|x_1\right\rangle_1\otimes\left|s_{x_1}\right\rangle_2\otimes\left|-\right\rangle. \tag{3.10}$$

The next step is applying g to s_{x_1} to solve the next order Bernstein-Vazirani problem, since $g(s_{x_1}) = s\cdot x_1$. The oracle for g is given by:

$$G=\sum_{x\in\{0,1\}^n}\sum_{y\in\{0,1\}}\left|x\right\rangle_2\left\langle x\right|_2\otimes\left|y\oplus g(x)\right\rangle\left\langle y\right|. \tag{3.11}$$

Attaching an extra ancilla qubit in the state $|-\rangle$, we can use the phase kick-back trick to apply this unitary and turn eq.(3.10) into:

$$\frac{1}{\sqrt{2^n}}\sum_{x_1\in\{0,1\}^n}(-1)^{g(s_{x_1})}\left|x_1\right\rangle_1\otimes\left|s_{x_1}\right\rangle_2\otimes\left|-\right\rangle\otimes\left|-\right\rangle. \tag{3.12}$$

This oracle acts only on the second register and the second extra qubit in the state $|-\rangle$. Of course, just one qubit in the state $|-\rangle$ is enough to implement both U_s and G, we are using two qubits just to separate each oracle more clearly. Next, we have to apply n H gates to the qubits of the second register and then we apply the oracle U_s again. After applying the H gates we have:

$$\frac{1}{2^n}\sum_{x_1,x_2\in\{0,1\}^n}(-1)^{g(s_{x_1})}(-1)^{s_{x_1}\cdot x_2}\left|x_1\right\rangle_1\otimes\left|x_2\right\rangle_2\otimes\left|-\right\rangle\otimes\left|-\right\rangle. \tag{3.13}$$

After the application of U_s we get another phase kickback:

$$\frac{1}{2^n}\sum_{x_1,x_2\in\{0,1\}^n}(-1)^{g(s_{x_1})}(-1)^{s_{x_1}\cdot x_2}(-1)^{s_{x_1}\cdot x_2}\left|x_1\right\rangle_1\otimes\left|x_2\right\rangle_2\otimes\left|-\right\rangle\otimes\left|-\right\rangle = \frac{1}{2^n}\sum_{x_1,x_2\in\{0,1\}^n}(-1)^{g(s_{x_1})}\left|x_1\right\rangle_1\otimes\left|x_2\right\rangle_2\otimes\left|-\right\rangle\otimes\left|-\right\rangle. \tag{3.14}$$

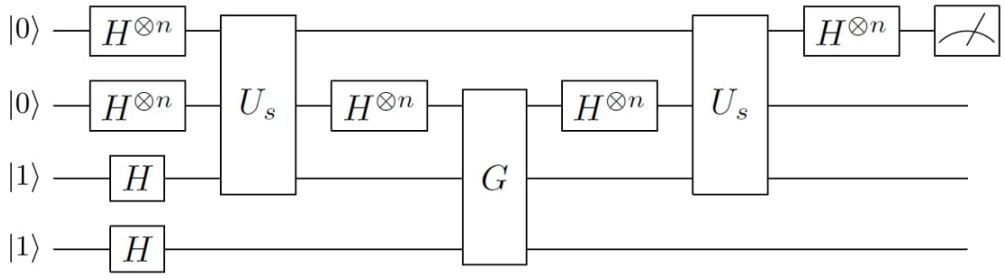


Figure 3.2: Recursive Bernstein-Vazirani circuit for level-2 recursion.

Then, applying H gates to all the qubits in the first register we get the following state:

$$\frac{1}{\sqrt{2^n}} \sum_{x_2 \in \{0,1\}^n} |s\rangle_1 \otimes |x_2\rangle_2 \otimes |-\rangle \otimes |-\rangle \,. \tag{3.15}$$

Finally, by measuring the qubits from the first register we obtain s with 100% probability. The recursive BV circuit is shown in figure 3.2.

4 Homomorphic Encryption of the BV algorithm

4.1 Motivation

The Bernstein-Vazirani algorithm is one of the first algorithms that showed the possible advantages of quantum computing. The recursive version in particular has historical importance because it was the first algorithm that showed a super-polynomial speed-up compared to what the best classical algorithms could achieve. We start our quest to find quantum algorithms that can be homomorphically evaluated in an efficient manner with the recursive BV algorithm due to its remarkable historical significance. We then found that the homomorphic implementation of this algorithm is not efficient in the general case and so we decided to study if there were particular cases where the homomorphic implementation could be performed efficiently. We give a positive answer to this question, as we managed to construct a class of circuits that solve the recursive BV problem for the second level of the recursion and can be implemented efficiently using Liang's [127] QHE schemes. The motivation for this work was theoretical rather than practical. We believe the work performed can lead to the application of quantum homomorphic encryption to a broader range of more complex and practical quantum algorithms and it also shows how particular efficient cases can be found even if the general case of an algorithm is not efficient.

In the recursive Bernstein-Vazirani algorithm the number of T gates required to solve the problem is not necessarily a polynomial in general. In our publication [155], we managed to reduce the number of T gates needed. In particular the circuits we have introduced, called T-linear circuits with reduced circuit complexity (RCC), allow the implementation of the oracle using a number of T gates that grows linearly with the number of qubits in the problem. We characterized these circuits in great detail, analysing all the possible cases in which they can be applied and their corresponding constraints. Since the number of T gates in T-linear circuits grows linearly, we proved that they constitute a perfect type of quantum circuits that can be homomorphically evaluated with perfect security and non-interaction in an efficient manner using QHE schemes. Besides these findings, we have also shown a possible extension of the previous results in a particular case that can be applied to any arbitrary level of the recursion. These circuits can also be homomorphically evaluated in an efficient manner, although not as efficiently as T-linear circuits with RCC.

4.2 T-linear circuits with reduced circuit complexity

In chapter 3 the recursive Bernstein-Vazirani algorithm was reviewed. In particular, the circuit that solves the recursive BV problem for $k = 2$ is shown in chapter 3, figure 3.2. In order to study the homomorphic evaluation of the algorithm, we need to implement this circuit with specific quantum gates. The oracle U_s for the recursive BV problem for $k = 2$ in the most general case is composed of multi-controlled $CNOT$ gates where n control qubits are used for the first register and one for the second. This way, the correct phase kickbacks are applied to the corresponding states.

As an example, we can consider the recursive BV problem for $k = 2$ where the secret bit string is $s = 11$ and the values of $g(s_{in})$ are selected as: $g(00) = 0$, $g(01) = g(10) = g(11) = 1$. Then the choice of values $s_{00} = s_{11} = 00$, $s_{01} = 01$, $s_{10} = 10$ is compatible with the fact that $g(s_{x_1}) = s \cdot x_1$. The circuit that solves this problem is represented using Qiskit [54] in figure 4.1, where the oracle U_s is implemented using multi-controlled $CNOT$ gates.

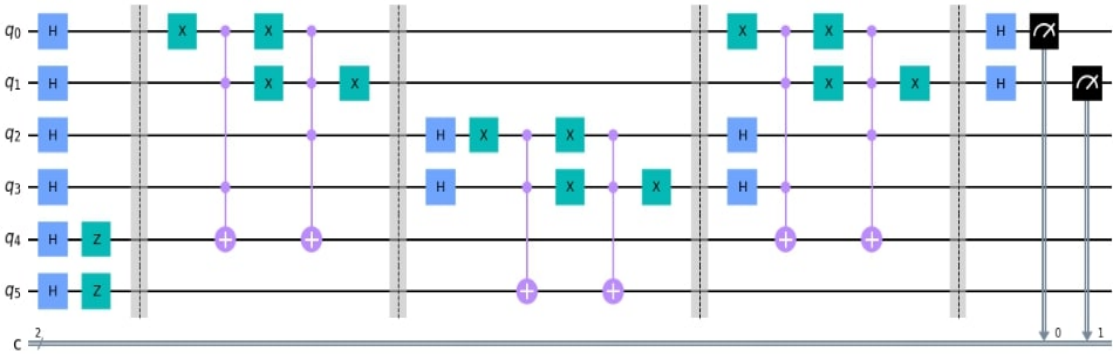


Figure 4.1: Recursive Bernstein-Vazirani circuit for two recursions for $s = 11$, represented in Qiskit.

However in certain situations the oracle U_s can be implemented using only Toffoli gates. We define the circuits where this is possible in the following section.

4.2.1 Definition and characterization

We start this section by defining T-linear circuits with reduced circuit complexity (RCC).

Definition 7 *T-linear circuits with RCC are quantum circuits that solve the Bernstein-Vazirani recursive problem when the oracle U_s can be constructed just with n Toffoli gates at most and the oracle G just with $CNOT$ gates.*

Here n is the same number as the qubits contained in each register of the algorithm.

It is also the number of bits needed to represent s. As a remark, the circuits allowed are those composed of H gates in some layers, a layer of Z gates after the first layer of H gates, Toffoli and X gates for the oracle U_s and $CNOT$ gates for G. It is useful to recall that the Toffoli gate performs the mapping $TOFF\,|a,b,c\rangle = |a,b,c \oplus (a \wedge b)\rangle$. We proceed to show for which values of $g(s_{in})$ and s_{x_1} a T-linear circuit with RCC can be constructed.

Ultimately, we want to answer the following question: given a mapping from $x_1 \in \{0,1\}^n$ to $s_{x_1} \in \{0,1\}^n$ and a function $g : \{0,1\}^n \to \{0,1\}$ such that $g(s_{x_1}) = s \cdot x_1$ for some fixed string $s \in \{0,1\}^n$. Is it possible to perform the mapping

$$|x_1\rangle_1 |x_2\rangle_2 |y\rangle \mapsto |x_1\rangle_1 |x_2\rangle_2 |y \oplus (s_{x_1} \cdot x_2)\rangle \quad \forall x_1, x_2 \in \{0,1\}^n, y \in \{0,1\}, \tag{4.1}$$

with at most n Toffoli gates?

We start with $g(s_{in})$. In the BV problem any value for $g(s_{in})$ can be selected on the condition that it equals 0 and 1 for a least one of its inputs. Let g be of the form $g(s_{in}) = s \cdot s_{in}$ and denote the Hamming weight of a string by $|s|$. The oracle G implementing $|x\rangle |y\rangle \mapsto |x\rangle |y \oplus g(x)\rangle = |x\rangle |y \oplus (\sum_{i=1}^{n} s_i x_i)\rangle$ in equation (3.11) can be constructed by a sequence of $|s|$ $CNOT$s, each one controlled by the qubit $|x_j\rangle$, $j \in \{\{1,\ldots,n\} : s_j = 1\}$, and targeting register $|y\rangle$. If the target $|y\rangle$ is in the state $|-\rangle$, the desired phase kickback is obtained. Thus, if the values of $g(s_{in})$ fulfil $g(s_{in}) = s \cdot s_{in}$ the oracle G can be implemented only using $CNOT$ gates. All T-linear circuits with RCC fulfil this condition. Moreover, this property of G is valid for any recursion k because G only acts on one register at a time.

Choosing $g(s_{in}) = s \cdot s_{in}$ restricts the circuits that can be analysed. Nevertheless, there is a good reason to restrict the function in this manner besides simplifying the problem, which is the fact that the homomorphic evaluation of $CNOT$ gates does not increase the complexity of the decryption process. If any other function g that is not balanced is used, implementing G using just $CNOT$ gates is impossible. In this case G would be implemented using multi-controlled $CNOT$ gates with n control qubits. Since these gates contain a larger number of $T/T^\dagger$ gates than the standard Toffoli gates, the decryption process of the QHE scheme would be more inefficient. The main focus of T-linear circuits with RCC is decreasing the complexity of this decryption process as much as is feasible.

Then if this g is chosen the problem narrows down to the mapping $x_1 \mapsto s_{x_1}$ with the restriction $s \cdot x_1 = s \cdot s_{x_1}$ for some string $s \in \{0,1\}^n$. Let $f : \{0,1\}^n \to \{0,1\}^n$ be such mapping so we have $s \cdot x_1 = s \cdot f(x_1)$. The mapping from equation (4.1) can be generically performed as

$$|x_1\rangle_1 |x_2\rangle_2 |y\rangle_{\mathtt{T}} \overset{1}{\mapsto} |f(x_1)\rangle_1 |x_2\rangle_2 |y\rangle_{\mathtt{T}} \overset{2}{\mapsto} |f(x_1)\rangle_1 |x_2\rangle_2 |y \oplus (f(x_1) \cdot x_2)\rangle_{\mathtt{T}} \overset{3}{\mapsto} |x_1\rangle_1 |x_2\rangle_2 |y \oplus (f(x_1) \cdot x_2)\rangle_{\mathtt{T}}. \tag{4.2}$$

Step 2 is simply the application of n Toffoli gates. The j-th Toffoli gate is controlled by the j-th qubit of registers 1 and 2 and it targets register $\mathtt{T}$, for $j \in \{1,\ldots,n\}$. The issue

is regarding Steps 1 and 3. Due to the fact that each step must be unitary, f must be a bijection. The problem is then implementing this map f applying only single-qubit gates. The reason for this restriction is that we want to implement U_s just with Toffoli gates that activate with either 0 or 1. Thus the only single-qubit gates that can be applied in the context of this function f are X gates.

The different mappings f that can be implemented are all valid T-linear circuits with RCC. The simplest mapping f that satisfies $s \cdot x_1 = s \cdot f(x_1)$ is $f(x_1) = x_1$, which is equivalent to the direct application of n Toffoli gates. This result is summarized in lemma 1:

Lemma 1 *A T-linear circuit with RCC can be constructed for $s_{x_1} = x_1$.*

Proof. As mentioned, finding a T-linear circuit amounts to selecting a f that fulfils the condition $s \cdot x_1 = s \cdot f(x_1)$. In this case $f(x_1) = x_1$ fulfils the condition simply by substitution.

□

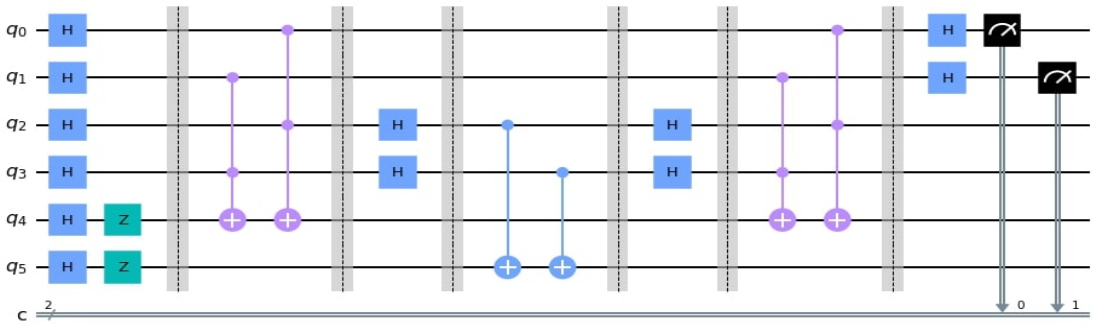


Figure 4.2: T-linear circuit with RCC for $s = 11$, $s_{00} = 00$, $s_{01} = 01$, $s_{10} = 10$ and $s_{11} = 11$, $g(00) = g(11) = 0$ and $g(01) = g(10) = 1$. This circuit is obtained from lemma 1.

In order to better illustrate this kind of circuits, we solve a circuit composed of 2 qubits per register, where $s = 11$. Since $g(s_{in})$ fulfils $g(s_{in}) = s \cdot s_{in}$, we have $g(00) = g(11) = 0$ and $g(01) = g(10) = 1$. For s_{x_1}, we have $s_{x_1} = x_1$ so: $s_{00} = 00$, $s_{01} = 01$, $s_{10} = 10$, $s_{11} = 11$. As these values were obtained from lemma 1, a T-linear circuit with RCC can be constructed. Thus, this circuit can be implemented making use of just $n = 2$ Toffoli gates for the oracle U_s and $CNOT$ gates for the oracle G. The circuit for this case is shown in figure 4.2. The qubits are denoted by q_0, q_1,..., q_{n-1}, where q_0 refers to the first one, q_1 to the second one and q_{n-1} to the n-th one. The first register is composed of two qubits, q_0 and q_1, and the second register has qubits q_2 and q_3.

We start by applying a layer of H gates to obtain the superposition of states of x_1 and x_2 from equation (3.8) and then U_s is applied to this superposition. Since $s_{01} = 01$,

the second qubit of each register, q_1 and q_3 in this case, is used as a control qubit for a Toffoli gate whose target is the first extra qubit in the state $|-\rangle$. This first Toffoli is denoted as $TOFF_{q_1,q_3,q_4}$ where q_1 and q_3 are control qubits and q_4 is the target qubit. A phase kickback occurs if both control qubits are in the state $|1\rangle$ and the target qubit is in the state $|-\rangle$: $TOFF\,|1,1,-\rangle = |1,1\rangle \left(\frac{|0\oplus(1\wedge1)\rangle - |1\oplus(1\wedge1)\rangle}{\sqrt{2}}\right) = -\,|1,1,-\rangle$. Applying this Toffoli gives us the state (see figure 4.2):

$$\begin{aligned} TOFF_{q_1,q_3,q_4}\Big[\frac{1}{4}\Big[\,&|00\rangle_1 \otimes (|00\rangle_2 + |01\rangle_2 + |10\rangle_2 + |11\rangle_2) \\ &+ |01\rangle_1 \otimes (|00\rangle_2 + |01\rangle_2 + |10\rangle_2 + |11\rangle_2) + |10\rangle_1 \otimes (|00\rangle_2 + |01\rangle_2 + |10\rangle_2 + |11\rangle_2) \\ &+ |11\rangle_1 \otimes (|00\rangle_2 + |01\rangle_2 + |10\rangle_2 + |11\rangle_2)\Big] \otimes |-\rangle\,\Big] = \frac{1}{4}\Big[\,|00\rangle_1 \otimes (|00\rangle_2 + |01\rangle_2 + |10\rangle_2 + |11\rangle_2) \\ &+ |01\rangle_1 \otimes (|00\rangle_2 - |01\rangle_2 + |10\rangle_2 - |11\rangle_2) + |10\rangle_1 \otimes (|00\rangle_2 + |01\rangle_2 + |10\rangle_2 + |11\rangle_2) \\ &+ |11\rangle_1 \otimes (|00\rangle_2 - |01\rangle_2 + |10\rangle_2 - |11\rangle_2)\Big] \otimes |-\rangle\,. \end{aligned} \tag{4.3}$$

The next s_{x_1} is $s_{10} = 10$, so now the first qubit of each register is selected as the control qubit for a Toffoli gate. Then, the control qubits are q_0 and q_2. This second Toffoli gate transforms the state into:

$$\begin{aligned} TOFF_{q_0,q_2,q_4}\Big[\frac{1}{4}\Big[\,&|00\rangle_1 \otimes (|00\rangle_2 + |01\rangle_2 + |10\rangle_2 + |11\rangle_2) \\ &+ |01\rangle_1 \otimes (|00\rangle_2 - |01\rangle_2 + |10\rangle_2 - |11\rangle_2) + |10\rangle_1 \otimes (|00\rangle_2 + |01\rangle_2 + |10\rangle_2 + |11\rangle_2) \\ &+ |11\rangle_1 \otimes (|00\rangle_2 - |01\rangle_2 + |10\rangle_2 - |11\rangle_2)\Big] \otimes |-\rangle\,\Big] = \frac{1}{4}\Big[\,|00\rangle_1 \otimes (|00\rangle_2 + |01\rangle_2 + |10\rangle_2 + |11\rangle_2) \\ &+ |01\rangle_1 \otimes (|00\rangle_2 - |01\rangle_2 + |10\rangle_2 - |11\rangle_2) + |10\rangle_1 \otimes (|00\rangle_2 + |01\rangle_2 - |10\rangle_2 - |11\rangle_2) \\ &+ |11\rangle_1 \otimes (|00\rangle_2 - |01\rangle_2 - |10\rangle_2 + |11\rangle_2)\Big] \otimes |-\rangle\,. \end{aligned} \tag{4.4}$$

The combination of both Toffolis also implements $s_{11} = 11$. The next step is applying two H gates to the second register so the state transforms into:

$$\begin{aligned} I \otimes H^2\Big[\frac{1}{4}[\,&|00\rangle_1 \otimes (|00\rangle_2 + |01\rangle_2 + |10\rangle_2 + |11\rangle_2) + |01\rangle_1 \otimes (|00\rangle_2 - |01\rangle_2 + |10\rangle_2 - |11\rangle_2) \\ &+ |10\rangle_1 \otimes (|00\rangle_2 + |01\rangle_2 - |10\rangle_2 - |11\rangle_2) + |11\rangle_1 \otimes (|00\rangle_2 - |01\rangle_2 - |10\rangle_2 + |11\rangle_2)] \otimes |-\rangle\,\Big] = \\ &= \frac{1}{2}\Big[\,|00\rangle_1 \otimes |00\rangle_2 + |01\rangle_1 \otimes |01\rangle_2 + |10\rangle_1 \otimes |10\rangle_2 + |11\rangle_1 \otimes |11\rangle_2\,\Big] \otimes |-\rangle\,. \end{aligned} \tag{4.5}$$

Then, G is applied using two $CNOT$ gates since $s = 11$, which gives the state:

$$I \otimes G\Big[\frac{1}{2}[|00\rangle_1 \otimes |00\rangle_2 + |01\rangle_1 \otimes |01\rangle_2 + |10\rangle_1 \otimes |10\rangle_2 + |11\rangle_1 \otimes |11\rangle_2] \otimes |-\rangle \otimes |-\rangle\Big] =$$
$$= \frac{1}{2}\Big[\,|00\rangle_1 \otimes |00\rangle_2 - |01\rangle_1 \otimes |01\rangle_2 - |10\rangle_1 \otimes |10\rangle_2 + |11\rangle_1 \otimes |11\rangle_2\Big] \otimes |-\rangle \otimes |-\rangle\,. \quad (4.6)$$

Since $g(01) = g(10) = 1$ a phase kickback is applied to the states $|01\rangle_2$ and $|10\rangle_2$. Next, H^2 and U_s are applied to the second register once more. After two H gates:

$$I \otimes H^2\Big[\frac{1}{2}\Big[\,|00\rangle_1 \otimes |00\rangle_2 - |01\rangle_1 \otimes |01\rangle_2 - |10\rangle_1 \otimes |10\rangle_2 + |11\rangle_1 \otimes |11\rangle_2\Big] \otimes |-\rangle \otimes |-\rangle\Big] =$$
$$= \frac{1}{4}\Big[\,|00\rangle_1 \otimes (|00\rangle_2 + |01\rangle_2 + |10\rangle_2 + |11\rangle_2) - |01\rangle_1 \otimes (|00\rangle_2 - |01\rangle_2 + |10\rangle_2 - |11\rangle_2)$$
$$-|10\rangle_1 \otimes (|00\rangle_2 + |01\rangle_2 - |10\rangle_2 - |11\rangle_2) + |11\rangle_1 \otimes (|00\rangle_2 - |01\rangle_2 - |10\rangle_2 + |11\rangle_2)\Big] \otimes |-\rangle \otimes |-\rangle\,. \quad (4.7)$$

After applying U_s once again:

$$U_s\Big[\frac{1}{4}[|00\rangle_1 \otimes (|00\rangle_2 + |01\rangle_2 + |10\rangle_2 + |11\rangle_2) - |01\rangle_1 \otimes (|00\rangle_2 - |01\rangle_2 + |10\rangle_2 - |11\rangle_2)$$
$$- |10\rangle_1 \otimes (|00\rangle_2 + |01\rangle_2 - |10\rangle_2 - |11\rangle_2) + |11\rangle_1 \otimes (|00\rangle_2 - |01\rangle_2 - |10\rangle_2 + |11\rangle_2)] \otimes |-\rangle$$
$$\otimes |-\rangle\Big] = \frac{1}{4}\Big[(|00\rangle_1 - |01\rangle_1 - |10\rangle_1 + |11\rangle_1) \otimes (|00\rangle_2 + |01\rangle_2 + |10\rangle_2 + |11\rangle_2)\Big] \otimes |-\rangle \otimes |-\rangle\,. \quad (4.8)$$

Finally, applying two H gates to the first register and measuring it returns the state $|11\rangle_1$ which is the correct value of s.

A different mapping f that fulfils $s \cdot x_1 = s \cdot f(x_1)$ is $f(x_1) = \overline{x_1}$ where $\overline{x_1}$ is the conjugate of x_1, with the restriction that $s \cdot x_1 = s \cdot \overline{x_1} \implies |s|$ is even. This is equivalent to flipping all the control qubits applying X gates that surround the n Toffoli gates or simply applying n Toffoli gates in which the control qubits activate with 0 instead of 1. We refer to such type of Toffoli gate by the name of negated Toffoli gate.

This result is summarized in lemma 2:

Lemma 2 *A T-linear circuit with RCC can be constructed for $s_{x_1} = \overline{x_1}$ provided $|s|$ is even.*

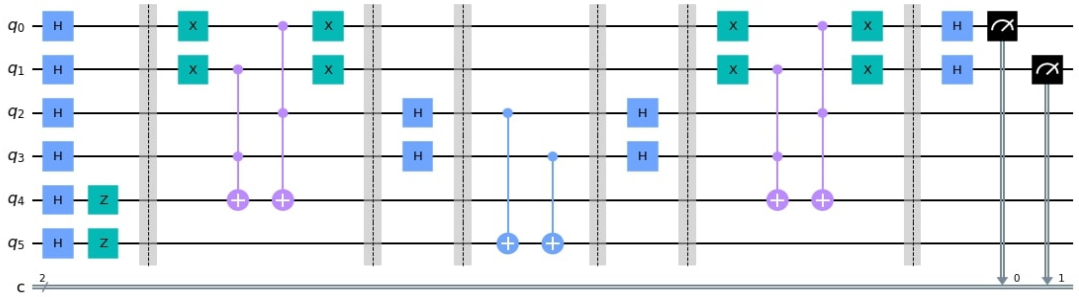


Figure 4.3: T-linear circuit with RCC for $s = 11$, $s_{00} = 11$, $s_{01} = 10$, $s_{10} = 01$, $s_{11} = 00$ and $g(00) = g(11) = 0$, $g(01) = g(10) = 1$. This circuit is obtained from lemma 2.

Proof. For $f(x_1) = \overline{x_1}$ we have $s \cdot \overline{x_1} = (\overline{x_1})_p \oplus \cdots \oplus (\overline{x_1})_q$ where each term of the sum has a corresponding value of $s_j = 1$ and $(\overline{x_1})_p$ is just the p-th bit of $\overline{x_1}$. If $|s|$ is even, the previous sum contains an even number of terms. The equation $\overline{a} \oplus \overline{b} = a \oplus b$ holds for arbitrary values of single bits a and b. Applying $\overline{a} \oplus \overline{b} = a \oplus b$ for every pair of bits in the previous sum and taking into account that there are no bits left unpaired as the sum contains an even number of terms we have $(\overline{x_1})_p \oplus \cdots \oplus (\overline{x_1})_q = (x_1)_p \oplus \cdots \oplus (x_1)_q$. Reintroducing the same s in the previous equation we arrive at $s \cdot \overline{x_1} = (\overline{x_1})_p \oplus \cdots \oplus (\overline{x_1})_q = (x_1)_p \oplus \cdots \oplus (x_1)_q = s \cdot x_1$ and so the equation $s \cdot \overline{x_1} = s \cdot x_1$ holds. Therefore $s_{x_1} = f(x_1) = \overline{x_1}$ fulfils $s \cdot x_1 = s \cdot f(x_1)$ if $|s|$ is even.

□

As an example of this lemma, if $s = 11$, then $s_{00} = 11$, $s_{01} = 10$, $s_{10} = 01$, $s_{11} = 00$ and the values of $g(s_{in})$ are $g(00) = g(11) = 0$, $g(01) = g(10) = 1$. In figure 4.3, the circuit that solves the BV problem for these values is represented. As it was mentioned, the control qubits of the first register of the Toffoli gates are surrounded by X gates, so they only activate if their state is a 0 instead of a 1.

More generally, we can have the mapping $f(x_1) = x_1 \oplus z$ for any string $z \in \{0,1\}^n$. In this case we have $s \cdot x_1 = s \cdot (x_1 \oplus z) \implies |s \wedge z|$ is even, where $s \wedge z$ is the bitwise AND operation between s and z. This is equivalent to substituting some of the n Toffoli gates by the same quantity of negated Toffoli gates, leaving the same control qubits that there were before. In particular, the j-th Toffoli gate is substituted by the negated Toffoli gate if $z_j = 1$. Of course, if $|z| = 0$ lemma 1 is recovered and if $|z| = n$ we have lemma 2.

This result is summarized in lemma 3:

Lemma 3 *A T-linear circuit with RCC can be constructed for $s_{x_1} = x_1 \oplus z$ for any string $z \in \{0,1\}^n$ provided $|s \wedge z|$ is even.*

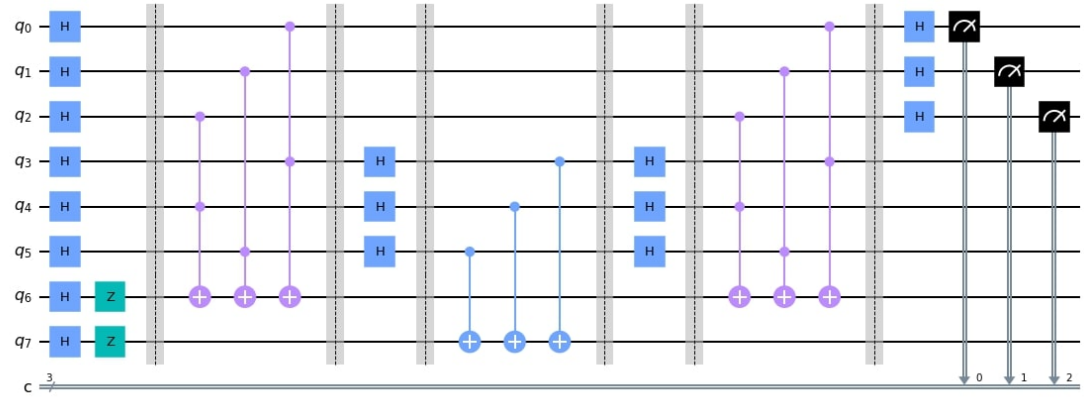


Figure 4.4: T-linear circuit with RCC for $s = 111$, $s_{000} = 000$, $s_{001} = 010$, $s_{010} = 001$, $s_{011} = 011$, $s_{100} = 100$, $s_{101} = 110$, $s_{110} = 101$, $s_{111} = 111$ and $g(000) = g(011) = g(101) = g(110) = 0$, $g(001) = g(010) = g(100) = g(111) = 1$. This circuit is obtained from lemma 1 after permuting the second and third control qubits of the second register.

Proof. For $f(x_1) = x_1 \oplus z$, we can expand the expression $s \cdot (x_1 \oplus z) = s_1((x_1)_1 \oplus z_1) \oplus \cdots \oplus s_n((x_1)_n \oplus z_n) \overset{\star}{=} s_1(x_1)_1 \oplus s_1 z_1 \oplus \cdots \oplus s_n(x_1)_n \oplus s_n z_n = s \cdot x_1 \oplus s \cdot z$ applying the distributive property $a(b \oplus c) = ab \oplus ac$ of single bits a, b and c on Step $\star$. If $|s \wedge z|$ is even, then $s \cdot z = \underbrace{1 \oplus \cdots \oplus 1}_{\text{even}} = 0$ and so if this is fact is applied in the previous equation we have $s \cdot (x_1 \oplus z) = s \cdot x_1 \oplus s \cdot z = s \cdot x_1 \oplus 0 = s \cdot x_1$. Therefore $s_{x_1} = f(x_1) = x_1 \oplus z$ fulfils $s \cdot x_1 = s \cdot f(x_1)$ if $|s \wedge z|$ is even. □

A different class of functions f that can be considered is moving part of the mapping f onto the Toffoli gates from Step 2 in equation (4.2). It is possible to permute the controls of the Toffoli gates and also delete some of the Toffoli gates. In particular, the valid permutations of the control qubits of the second register of the Toffoli gates also result in T-linear circuits with RCC. A valid permutation of the bits of $f(x_1)$, denoted by $\sigma(f(x_1))$, must fulfil $s \cdot x_1 = s \cdot \sigma(f(x_1))$ with the restriction that for each pair of bits that are permuted, $p \in \{1, \ldots, n\}$ and $q \in \{1, \ldots, n\}$, $s_p = s_q$ in each permutation. We can write the permutation of bits p and q of $f(x_1)$ explicitly as $\sigma(f(x_1)) = \sigma(f(x_1)_1, \ldots, f(x_1)_p, \ldots, f(x_1)_q, \ldots, f(x_1)_n) = (f(x_1)_1, \ldots, f(x_1)_q, \ldots, f(x_1)_p, \ldots, f(x_1)_n)$. This is equivalent to permuting the control qubits located in the second register of the pair of Toffoli gates p and q, leaving the control qubits of the first register unchanged. Any of the previous mappings is fit to be permuted, provided each of their respective restrictions are also taken into consideration. As an example if $f(x_1) = \overline{x_1}$, besides $s_p = s_q$ for the permuted bits p and q, $|s|$ must be even. For a given s the maximum number of valid permutations is $n! - 1$ (all the possible permutations except the starting configuration) and occurs when $|s| = n$.

As an example, if $s = 111$ and $s_{x_1} = f(x_1) = x_1$, then there are $3! - 1$ valid permutations available for the values of s_{x_1} obtained from lemma 1. The starting values of $s_{x_1} = x_1$ in this example are: $s_{000} = 000$, $s_{001} = 001$, $s_{010} = 010$, $s_{011} = 011$, $s_{100} = 100$, $s_{101} = 101$, $s_{110} = 110$, $s_{111} = 111$. The first permutation could be switching the position of bits 2 and 3 so we have $\sigma(f(x_1)) = \sigma(f(x_1)_1, f(x_1)_2, f(x_1)_3) = (f(x_1)_1, f(x_1)_3, f(x_1)_2)$: $s_{000} = 000$, $s_{001} = 010$, $s_{010} = 001$, $s_{011} = 011$, $s_{100} = 100$, $s_{101} = 110$, $s_{110} = 101$, $s_{111} = 111$. In this example the pair of bits permuted, 2 and 3, satisfy $s_2 = s_3$. Then the control qubits 2 and 3 of the second register of the Toffoli gates will be permuted. This can be seen in figure 4.4 which shows the circuit that solves this example. The remaining permuted circuits can be obtained using this process.

With respect to the erasure of Toffoli gates, this can be achieved by setting some bits of $f(x_1)$ to zero, i.e., by using the mapping $f(x_1) \wedge u$ for some string $u \in \{0,1\}^n$ instead of $f(x_1)$. In this case f is still a bijection that fulfils $s \cdot x_1 = s \cdot f(x_1)$. This is equivalent to erasing the j-th Toffoli gate if $s_j = 0$.

This result is summarized in lemma 4:

Lemma 4 *A T-linear circuit with RCC can be constructed for $s_{x_1} = f(x_1) \wedge u$ for some string $u \in \{0,1\}^n$ provided $u \wedge s = s$.*

Proof. For $f(x_1) \wedge u$ we have that $s \cdot (f(x_1) \wedge u) = \sum_{j=1}^{n} s_j f(x_1)_j u_j \overset{\star}{=} \sum_{j=1}^{n} s_j^2 f(x_1)_j = \sum_{j=1}^{n} s_j f(x_1)_j = s \cdot f(x_1) = s \cdot x_1$ if $u \wedge s = s$ (used on Step $\star$), i.e., if the bits of s whose value is 1 are also bits of u whose value is 1. Therefore $s_{x_1} = f(x_1) \wedge u$ if $u \wedge s = s$ and f a bijection are also valid mappings. □

Of course, the valid mappings for f that can be used in lemma 4 are all those from the previous lemmas such as $f(x_1) = x_1 \oplus z$ for any string $z \in \{0,1\}^n$ if $|s \wedge z|$ is even.

As an example of lemma 4, if $s = 001$, the available values for $u = \{001, 011, 101, 111\}$ due to the restriction $u \wedge s = s$. Taking $f(x_1) = x_1$ and $u = 011$ for this specific case, the values of s_{x_1} according to lemma 4 are $s_{x_1} = x_1 \wedge u$ thus: $s_{000} = 000$, $s_{001} = 001$, $s_{010} = 010$, $s_{011} = 011$, $s_{100} = 000$, $s_{101} = 001$, $s_{110} = 010$, $s_{111} = 011$. Since $g(s_{in})$ satisfies $g(s_{in}) = s \cdot s_{in}$ the values of $g(s_{in})$ are $g(000) = g(010) = g(100) = g(110) = 0$, $g(001) = g(011) = g(101) = g(111) = 1$. For these values a T-linear circuit with RCC can be implemented. The circuit that solves this example is represented in figure 4.5. Since a Toffoli gate has been erased, U_s is constructed using two Toffoli gates instead of three.

Naturally, there are other ways of setting $s \cdot (f(x_1) \wedge u) = s \cdot x_1$, e.g. dividing the strings $x_1 \in \{0,1\}^n$ into the two sets $\mathcal{X}_0 := \{x_1 \in \{0,1\}^n : s \cdot x_1 = 0\}$ and $\mathcal{X}_1 := \{x_1 \in \{0,1\}^n : s \cdot x_1 = 1\}$ and mapping $\mathcal{X}_0 \mapsto \{0^n\}$ and $\mathcal{X}_1 \mapsto \{0^{l-1}10^{n-l}\}$, where $l \in \{1, \ldots, n\}$ is any entry such that $s_l = 1$. Such mapping can be achieved by $u = 0^{l-1}10^{n-l}$ and $f(x_1) = x_1 \oplus \sum_{m \neq l}^{n} 0^{l-1}(x_1)_m 0^{n-l}$ where $m \neq l$ is any entry such that $s_m = 1$, the

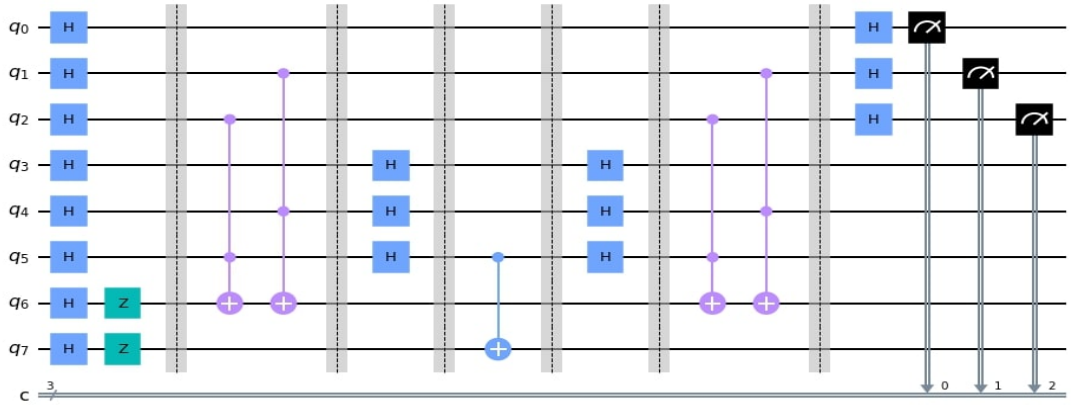


Figure 4.5: T-linear circuit with RCC for $s = 001$, $s_{000} = 000$, $s_{001} = 001$, $s_{010} = 010$, $s_{011} = 011$, $s_{100} = 000$, $s_{101} = 001$, $s_{110} = 010$, $s_{111} = 011$ and $g(000) = g(010) = g(100) = g(110) = 0$, $g(001) = g(011) = g(101) = g(111) = 1$. This circuit is obtained from lemma 4.

sum $\sum_{m\neq l}^{n} 0^{l-1}(x_1)_m 0^{n-l}$ refers to the bitwise sum of strings (each containing n bits) and $(x_1)_m$ is simply the m-th bit of x_1. If such entries do not exist, i.e. $|s| = 1$, then $f(x_1) = x_1 \oplus \sum_{m\neq l}^{n} 0^{l-1}(x_1)_m 0^{n-l} = x_1 \oplus 0^n = x_1$. This is equivalent to having every Toffoli gate use the same control qubit l in the second register while leaving the control qubits of the first register unchanged.

This result summarized in lemma 5:

Lemma 5 *A T-linear circuit with RCC can be constructed, given a string* $u = 0^{l-1}10^{n-l}$, *for* $s_{x_1} = \left[x_1 \oplus \sum_{m\neq l}^{n} 0^{l-1}(x_1)_m 0^{n-l}\right] \wedge u$, *provided* $s_l = 1$ *and* $s_m = 1$ *for any entry where* $m \neq l$ *in the sum.*

Proof. For this mapping we have $s_{x_1} = f(x_1) \wedge u = \left[x_1 \oplus \sum_{m\neq l}^{n} 0^{l-1}(x_1)_m 0^{n-l}\right] \wedge u$ so $s \cdot (f(x_1) \wedge u) = \sum_{j=1}^{n} s_j f(x_1)_j u_j \overset{1}{=} s_l f(x_1)_l$ if $u = 0^{l-1}10^{n-l}$ is used on Step 1. Next, substituting $f(x_1)$ we have $s_l f(x_1)_l = s_l[(x_1)_l \oplus \sum_{m\neq l}^{n}(x_1)_m] \overset{2}{=} (x_1)_l \oplus \sum_{m\neq l}^{n}(x_1)_m$ if $s_l = 1$ (used on Step 2). Then the previous equation is equal to $s \cdot x_1 = \sum_{j=1}^{n} s_j(x_1)_j = s_l(x_1)_l \oplus \sum_{m\neq l}^{n} s_m(x_1)_m \overset{3}{=} (x_1)_l \oplus \sum_{m\neq l}^{n}(x_1)_m$ if $s_l = 1$ and $s_m = 1$ for each term that appears in the sum in m (used on Step 3). Therefore $s_{x_1} = \left[x_1 \oplus \sum_{m\neq l}^{n} 0^{l-1}(x_1)_m 0^{n-l}\right] \wedge u$ is a valid mapping if $u = 0^{l-1}10^{n-l}$, $s_l = 1$ and $s_m = 1$ for any entry where $m \neq l$ in the sum. □

As a remark, if negated Toffoli gates were used instead, the results of lemma 5 remain the same as long as the corresponding restrictions are taken into account. If instead

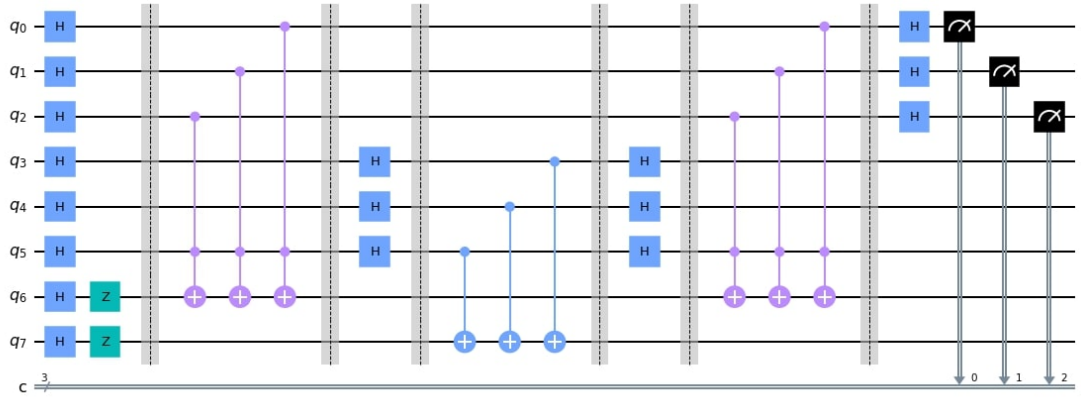


Figure 4.6: T-linear circuit with RCC for $s = 111$, $s_{000} = 000$, $s_{001} = 001$, $s_{010} = 001$, $s_{011} = 000$, $s_{100} = 001$, $s_{101} = 000$, $s_{110} = 000$, $s_{111} = 001$ and $g(000) = g(011) = g(101) = g(110) = 0$, $g(001) = g(010) = g(100) = g(111) = 1$. This circuit is obtained from lemma 5.

of $f(x_1) = x_1 \oplus \sum_{m\neq l}^{n} 0^{l-1}(x_1)_m 0^{n-l}$ a different mapping is used such as $f(x_1) = \overline{x_1} \oplus \sum_{m\neq l}^{n} 0^{l-1}(\overline{x_1})_m 0^{n-l}$ with $|s|$ even or $f(x_1) = x_1 \oplus z \oplus \sum_{m\neq l}^{n} 0^{l-1}(x_1 \oplus z)_m 0^{n-l}$ with $|s \wedge z|$ even and $z \in \{0,1\}^n$, the same results are obtained.

As an example of lemma 5, if $s = 111$ and taking $l = 3$ since $s_3 = 1$, we have that $u = 0^{3-1}10^{3-3} = 001$. Then $m = 1, 2$ since $s_1 = s_2 = 1$. Thus $s_{x_1} = [x_1 \oplus \sum_{m\neq l}^{n} 0^{l-1}(x_1)_m 0^{n-l}] \wedge u = [x_1 \oplus 00(x_1)_1 \oplus 00(x_1)_2] \wedge 001$ so: $s_{000} = 000$, $s_{001} = 001$, $s_{010} = 001$, $s_{011} = 000$, $s_{100} = 001$, $s_{101} = 000$, $s_{110} = 000$, $s_{111} = 001$. The resulting circuit is shown in figure 4.6. As it can be seen, the control qubit of the second register for all Toffoli gates is the third qubit.

In lemma 5 all Toffoli gates use the same control qubit l but naturally this lemma can be generalized further if only some of the Toffoli gates use the same control qubit. In this scenario, a valid transformation of $f(x_1)$, denoted by $\tau(f(x_1))$, applied on its bits $p \in \{1, \ldots, n\}$ and $q \in \{1, \ldots, n\}$ can be written as: $\tau(f(x_1)) = \tau(f(x_1)_1, \ldots, f(x_1)_p, f(x_1)_q, \ldots, f(x_1)_n) = (f(x_1)_1, \ldots, 0_p, f(x_1)_p \oplus f(x_1)_q, \ldots, f(x_1)_n)$. The transformation must fulfil $s \cdot x_1 = s \cdot \tau(f(x_1))$ with the restriction that $s_p = s_q = 1$ like in lemma 5. This transformation may be performed for as many bits as desired. If it is applied $n - 1$ times, selecting all the possible values of p except one that acts as q each time the transformation is applied, we recover lemma 5 assuming that τ is applied to $f(x_1) = x_1$.

Next, in a similar manner to lemma 5, we can divide the strings $x_1 \in \{0,1\}^n$ into the two sets $\mathcal{X}_0 := \{x_1 \in \{0,1\}^n : s \cdot x_1 = 0\}$ and $\mathcal{X}_1 := \{x_1 \in \{0,1\}^n : s \cdot x_1 = 1\}$ and map $\mathcal{X}_0 \mapsto \{0^n\}$ and $\mathcal{X}_1 \mapsto \{1^n\}$ where once more $l \in \{1, \ldots, n\}$ is any entry such that $s_l = 1$. Such mapping can be achieved by $f(x_1) = (x_1 \wedge u) \oplus \sum_{m\neq l}^{n} 0^{m-1}(x_1)_l 0^{n-m} = \sum_{m=1}^{n} 0^{m-1}(x_1)_l 0^{n-m}$ if $u = 0^{l-1}10^{n-l}$, where $m \in \{1, \ldots, n\}$, and $(x_1)_l$ is simply the

l-th bit of x_1. This is equivalent to having every Toffoli gate use the same control qubit l in the first register while leaving the control qubits of the second register with no changes.

This result is summarized in lemma 6:

Lemma 6 *A T-linear circuit with RCC can be constructed for* $s_{x_1} = (x_1 \wedge u) \oplus \sum_{m\neq l}^{n} 0^{m-1}(x_1)_l 0^{n-m} = \sum_{m=1}^{n} 0^{m-1}(x_1)_l 0^{n-m}$*, provided* $u = 0^{l-1}10^{n-l}$*,* $s_l = 1$ *and also* $|s| = 1$*.*

Proof. For this mapping we have that $s \cdot f(x_1) = s \cdot [(x_1 \wedge u) \oplus \sum_{m\neq l}^{n} 0^{m-1}(x_1)_l 0^{n-m}] \overset{1}{=} s \cdot [\sum_{m=1}^{n} 0^{m-1}(x_1)_l 0^{n-m}] = \sum_{j=1}^{n} s_j(x_1)_l$ if $u = 0^{l-1}10^{n-l}$ (used on Step 1). If $s_l = 1$ and $|s| = 1$, then all $s_j = 0$ except s_l and the previous equation can be expanded as $\sum_{j=1}^{n} s_j(x_1)_l = s_l(x_1)_l = (x_1)_l$. Thus the previous equation is equal to $s \cdot x_1 = \sum_{j=1}^{n} s_j(x_1)_j \overset{2}{=} (x_1)_l$ if $s_l = 1$ and $|s| = 1$ (used on Step 2). Both sides of the equation $s \cdot f(x_1) = s \cdot x_1$ are equal and therefore $s_{x_1} = (x_1 \wedge u) \oplus \sum_{m\neq l}^{n} 0^{m-1}(x_1)_l 0^{n-m}$ is a valid mapping if $u = 0^{l-1}10^{n-l}$, $s_l = 1$ and $|s| = 1$. □

Unlike in lemma 4 and lemma 5, not all the Toffoli gates can be substituted by negated Toffoli gates. In this instance the mapping $f(x_1) = (\overline{x_1} \wedge u) \oplus \sum_{m\neq l}^{n} 0^{m-1}(\overline{x_1})_l 0^{n-m} = \sum_{m=1}^{n} 0^{m-1}(\overline{x_1})_l 0^{n-m}$ can not be implemented due to the incompatibility between the conditions $|s| = 1$ and $|s|$ even. The mapping $f(x_1) = ((x_1 \oplus z) \wedge u) \oplus \sum_{m\neq l}^{n} 0^{m-1}(x_1 \oplus z)_l 0^{n-m} = \sum_{m=1}^{n} 0^{m-1}(x_1 \oplus z)_l 0^{n-m}$ can be applied as long as $|s \wedge z|$ is even which implies $z_l = 0$ due to $|s| = 1$. Therefore, the mapping $\mathcal{X}_0 \mapsto \{0^n\}$ and $\mathcal{X}_1 \mapsto \{1^n\}$ can only be implemented with regular Toffoli gates like in lemma 6.

As an example of lemma 6, if $s = 100$ then $l = 1$ since $s_1 = 1$. We have $u = 100$ which leads to $s_{x_1} = \sum_{m=1}^{n} 0^{m-1}(x_1)_l 0^{n-m} = (x_1)_1 00 \oplus 0(x_1)_1 0 \oplus 00(x_1)_1$ so: $s_{000} = 000$, $s_{001} = 000$, $s_{010} = 000$, $s_{011} = 000$, $s_{100} = 111$, $s_{101} = 111$, $s_{110} = 111$, $s_{111} = 111$. The resulting circuit is shown in figure 4.7. As it can be seen, the control qubit of the first register for all Toffoli gates is the first qubit.

In lemma 6 all Toffoli gates use the same control qubit of the first register l. Like in lemma 5, this can be also applied if only some Toffoli gates use the same control qubit. In such case, a valid transformation of $f(x_1)$, denoted by $\beta(f(x_1))$, applied on its bits $p \in \{1, \ldots, n\}$ and $q \in \{1, \ldots, n\}$ can be expressed as: $\beta(f(x_1)) = \beta(f(x_1)_1, \ldots, f(x_1)_p, f(x_1)_q, \ldots, f(x_1)_n) = (f(x_1)_1, \ldots, f(x_1)_p, f(x_1)_p, \ldots, f(x_1)_n)$. The transformation must fulfil $s \cdot x_1 = s \cdot \beta(f(x_1))$ with the restriction that $s_p = 1$ and $s_q = 0$. This transformation may be performed for as many bits as desired. Once more, if it is applied $n - 1$ times, selecting all possible values of q except one that acts as p each time the transformation is applied, lemma 6 is recovered assuming that β is applied to $f(x_1) = x_1$.

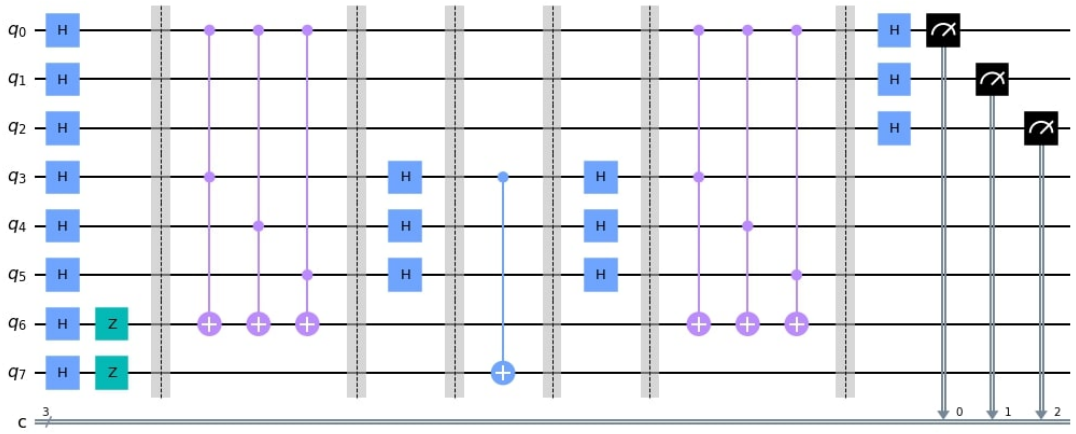


Figure 4.7: T-linear circuit with RCC for $s = 100$, $s_{000} = 000$, $s_{001} = 000$, $s_{010} = 000$, $s_{011} = 000$, $s_{100} = 111$, $s_{101} = 111$, $s_{110} = 111$, $s_{111} = 111$, and $g(000) = g(001) = g(010) = g(011) = 0$, $g(100) = g(101) = g(110) = g(111) = 1$. This circuit is obtained from lemma 6.

By combining all the previous lemmas and transformations that have been presented, the remaining valid T-linear circuits with RCC can be implemented, i.e., a circuit that contains regular and negated Toffoli gates, where some of their control qubits are permuted, some others act on the same qubit of the first register, others on the same qubit of the second register and some Toffoli gates are deleted altogether is also a T-linear circuit with RCC, provided the corresponding restrictions are considered for the qubits involved.

For the function $g(s_{in})$ selected, the degrees of freedom of the problem are: which type of Toffoli gate is applied regarding if it activates with a 1 or a 0, the number of Toffoli gates used, and the control qubits used. Thus all possible cases are: applying n Toffoli gates, negated Toffoli gates or a combination of both types of gates (lemmas 1, 2, and 3 respectively), erasing some of these n Toffoli gates (lemma 4), all the possible permutations of the control qubits of the second register for all the previous lemmas (given by $\sigma(f(x_1))$), using the same control qubit of the second register for all the Toffoli gates (lemma 5) or only for some of them (given by $\tau(f(x_1))$) and lastly using the same control qubit of the first register for all the Toffoli gates (lemma 6) or only for some of them (given by $\beta(f(x_1))$). Therefore the circuits obtained from all the possible combinations of all these cases constitute the complete set of T-linear circuits with RCC for $k = 2$. As a remark, the circuits allowed here are those composed of H gates in some layers, a layer of Z gates after the first layer of H gates, Toffoli gates and X gates for the oracle U_s and $CNOT$ gates for G. In regard to the number of Toffoli gates that are required for T-linear circuits with RCC, at worst n gates are needed. The best case scenario occurs for a circuit obtained from lemma 4 where all Toffoli gates are erased

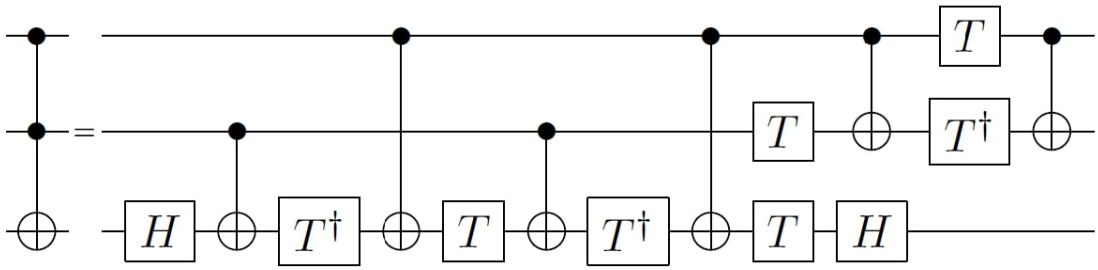


Figure 4.8: A Toffoli gate can be implemented using 7 $T/T^\dagger$.

but one.

4.2.2 Homomorphic implementation and extended results

The main feature of T-linear circuits with RCC is the fact that the oracle U_s is simplified from multi-controlled $CNOT$ gates to just n Toffoli gates at most and the oracle G is also simplified so it can be constructed using $CNOT$ gates instead of multi-controlled $CNOT$ gates. The importance of this simplification is related to the number of T gates needed to implement each operation.

For T-linear circuits with RCC, like the one in figure 4.2, the number of T gates grows linearly with the number of qubits of each register (the same number of bits needed to represent s), n, required to solve the problem. As it has already been discussed, the reason for this is that in the worst case n Toffoli gates are required to implement U_s. Since a Toffoli gate can be decomposed in 7 $T/T^\dagger$ gates as shown in figure 4.8 ($T^\dagger$ gates are counted as T gates) and taking into account that U_s must be applied twice to solve the recursive BV problem for $k = 2$, the number of T gates required is $14 \cdot n$, which means that this number grows linearly with n, thus the number of T gates grows as $O(n)$.

In chapter 2 it was shown that Liang's QHE schemes are only suitable for circuits with a polynomial number of $T/T^\dagger$ gates. Thus, this fact means that T-linear circuits are a perfect example of an algorithm that can be evaluated in an efficient manner using Liang's QHE schemes. Summarizing in corollary 1:

Corollary 1 *The number of $T/T^\dagger$ gates required to implement T-linear circuits with RCC grows linearly with the number of qubits of each register needed to solve the problem: $O(n)$. Therefore T-linear circuits with RCC can be implemented efficiently using Liang's QHE schemes.*

On the other hand, for the $k = 2$ case, if U_s has to be constructed using only multi-controlled $CNOT$ gates, the number of T gates grows differently. Since $g(s_{x_1}) = s \cdot x_1$,

half of the values of $g(s_{x_1})$ will be 1, so half of all s_{x_1} strings will have $|s_{x_1}| \neq 0$ (they will not be equal to 0^n). Then U_s will require at least a multi-controlled $CNOT$ gate for half of the s_{x_1} strings, because in contrast to Toffoli gates, a multi-controlled $CNOT$ gate is restricted to just one value of x_1 each time it is applied. Thus, the number of multi-controlled $CNOT$ gates will be 2^{n-1} at least, since there are 2^n s_{x_1} strings and we have to take half of that number.

Taking into account that U_s is applied twice, the number of multi-controlled $CNOT$ gates would be 2^n. Assuming for simplicity that each multi-controlled $CNOT$ gate could be constructed using 7 $T/T^\dagger$ gates like the Toffoli gates, the number of T gates grows exponentially with the number of qubits of each register n, thus the number of T gates grows as $\Omega(2^n)$. Therefore this implementation of U_s would not be appropriate to be evaluated homomorphically applying the QHE schemes presented in chapter 2.

Assuming that the multi-controlled $CNOT$ gates contain the same number of T gates as the Toffoli gates is an extreme simplification. Still since the number of T gates is exponential even in this case, it is enough for our purpose: studying if the number of T gates contained in the circuit is exponential or polynomial.

We want to mention some cases where U_s must be implemented using multi-controlled $CNOT$ gates because Toffoli gates are not enough. A group of circuits where this occurs are those where $2^{n-1}+1$ of the s_{x_1} strings are mapped making use of any of the previous lemmas but the remaining strings are mapped to different values that are still valid. For instance it is possible to construct a BV problem in which one half of the s_{x_1} strings are mapped to x_1 and the other half of the s_{x_1} strings are mapped to $s_{max} = 1^n$ since $g(s_{max}) = s \cdot s_{max} = 0 \ \forall s$ if $|s|$ is even and $g(s_{max}) = s \cdot s_{max} = 1 \ \forall s$ if $|s|$ is odd (assuming $g(s_{in})$ satisfies $g(s_{in}) = s \cdot s_{in}$).

As an example of such case if $s = 111$, then $s_{000} = 000$, $s_{001} = 111$, $s_{010} = 111$, $s_{011} = 011$, $s_{100} = 111$, $s_{101} = 101$, $s_{110} = 110$, $s_{111} = 111$ and $g(000) = g(011) = g(101) = g(110) = 0$, $g(001) = g(010) = g(100) = g(111) = 1$. Since $|s|$ is odd, $g(s_{max}) = s \cdot s_{max} = s \cdot 111 = 1$ which means that s_{max} is a valid value for all s_{x_1} strings where $g(s_{x_1}) = s \cdot x_1 = 1$. This example is basically what would be obtained from using lemma 1 for $s = 111$ but substituting half of the values of s_{x_1} for s_{max}. Nevertheless, it is impossible to construct this circuit just using n Toffoli gates, because $s_{001} = s_{010} = s_{100} = 111$, but according to lemma 6 we have that $\mathcal{X}_1 \mapsto \{1^n\}$ only if $|s| = 1$. This means that if s_{001} is mapped to s_{max}, then $s_{010} = s_{100} \neq s_{max}$ so this particular case can not be implemented with Toffoli gates and requires multi-controlled $CNOT$ gates. For this particular example, using $s_{x_1} = x_1$ as in lemma 1 to apply 3 Toffoli gates would implement U_s for all the s_{x_1} strings where $g(s_{x_1}) = 0$ and would implement $s_{111} = 111$ too. This is performed in order to minimize the number of multi-controlled $CNOT$ gates as much as is feasible. Still, for the remaining values (s_{001}, s_{010}, s_{100}) two multi-controlled $CNOT$ gates would be required for each s_{x_1} so U_s implements $s_{001} = s_{010} = s_{100} = s_{max}$.

In the general case for an arbitrary n where $\mathcal{X}_1 \mapsto \{1^n\}$, $\mathcal{X}_0 \mapsto \{x_1\}$ and assuming that n Toffoli gates are employed to map $\mathcal{X}_0 \mapsto \{x_1\}$ so the maximum number of phase kickbacks are applied, minimizing the amount of multi-controlled $CNOT$ gates utilized in these circuits, $2^{n-1} - 1$ s_{x_1} strings still have to be mapped to s_{max} after the Toffoli gates are applied. Then each of these strings would require at least a multi-controlled $CNOT$ gate (most strings would require more) in order to obtain the desired mapping, so at minimum $2^{n-1} - 1$ multi-controlled $CNOT$ gates would be required to implement U_s. Thus, as it has been discussed previously, the best possible number of T gates for these types of circuits grows as $O(2^n)$. If Toffoli gates were not used then each of the 2^{n-1} s_{x_1} strings that must be mapped to s_{max} would require n multi-controlled $CNOT$ gates, making the number of T gates in the circuit grow exponentially once more (worse than the former n Toffoli gates case).

Finally we want to briefly discuss how the results obtained for T-linear circuits with RCC could be extended for different functions $g(s_{in})$ than the one we used and for $k > 2$.

A simple function g could be $g(s_{in}) = s \cdot \overline{s_{in}}$. This function is simply the application of $|s|$ negated $CNOT$ gates. Then we have that f must obey $s \cdot x_1 = s \cdot \overline{f(x_1)}$. This means that all the previous results presented for T-linear circuits with RCC still apply here, taking into consideration that the restrictions that negated Toffoli gates had are now the restrictions of the usual Toffoli gates. As an example if $f(x_1) = x_1 \implies s \cdot x_1 = s \cdot \overline{f(x_1)} = s \cdot \overline{x_1}$, so by using the mapping f of lemma 1 we recover the condition of lemma 2. Generalizing further, any balanced function $g(s_{in}) = s \cdot (s_{in} \oplus z)$ for any string $z \in \{0,1\}^n$ can be selected, taking into account the corresponding restrictions when selecting a specific $f(x_1)$. All these functions can still be implemented with just $|s|$ $CNOT$ gates, mixing usual and negated $CNOT$ gates.

Next we can investigate functions g that are not balanced. As we mentioned previously, these functions have to be implemented with multi-controlled $CNOT$ gates with n control qubits that activate with 0 or 1 as required and target the state $|-\rangle$ to generate phase kickbacks. A multi-controlled $CNOT$ gate with n control qubits can be decomposed into $2(n-1)$ Toffoli gates using $(n-1)$ extra ancilla qubits. An example of this decomposition [139] is presented in figure 4.9 for any multi-controlled U gate.

We are still interested in implementing U_s with n Toffoli gates at most for these kind of functions g. For example the function g defined as $g(s_{in}) = 1$ if $s_{in} = 0^{n-1}1$ and $g(s_{in}) = 0$ for the remaining inputs can be constructed using only one multi-controlled $CNOT$ gate that activates for the state $\left|0^{n-1}1\right\rangle$ and targets the state $|-\rangle$. Making use of lemma 5 we can map $\mathcal{X}_0 \mapsto \{0^n\}$ and $\mathcal{X}_1 \mapsto \{0^{n-1}1\}$ taking $l = n$. It is important to notice that for this type of function g, the restriction $s_l = 1$ from lemma 5 does not apply any more and instead we have $l \in \{\{1, \ldots, n\} : g(0^{l-1}10^{n-l}) = 1\}$, so in this example $g(0^{l-1}10^{n-l}) = g(0^{n-1}1) = 1$ and $l = n$. The mapping for f is now $f(x_1) \wedge u = \left[\sum_m^n 0^{l-1}(x_1)_m 0^{n-l}\right] \wedge u$ for $u = 0^{l-1}10^{n-l}$ and any entry m such that $s_m = 1$. Then for the example chosen $l = n$ and we have that $f(x_1) \wedge u = \left[\sum_m^n 0^{n-1}(x_1)_m\right] \wedge 0^{n-1}1 = \sum_m^n 0^{n-1}(x_1)_m$. From this we obtain a valid implementation of U_s for this particular

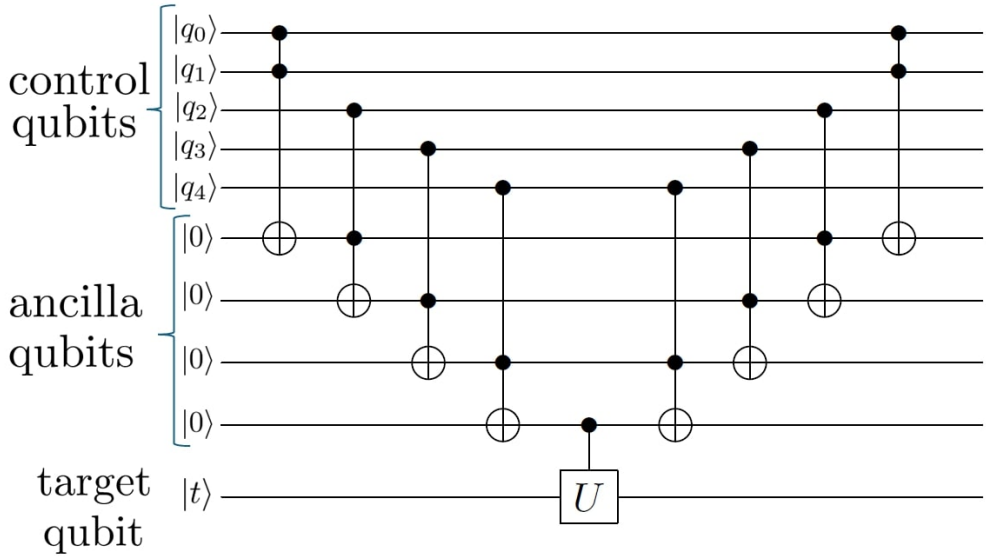


Figure 4.9: A multi-controlled single qubit gate U with $n = 5$ control qubits, decomposed into 8 Toffoli gates using 4 ancilla qubits in the state $|0\rangle$. If $U = X$ we have a multi-controlled $CNOT$ gate.

function g that still uses only n Toffoli gates at most. Nevertheless after decomposing the multi-controlled $CNOT$ gate into $2(n-1)$ Toffoli gates, the resulting circuit now needs $n-1$ ancilla qubits and contains $2n+2(n-1)$ Toffoli gates at most. The number of T gates then still grows as $O(n)$. Thus the decryption procedure of this circuit is still efficient provided $n-1$ extra ancilla qubits are employed in the QHE scheme. Naturally, this result also works for a function g defined as $g(s_{in}) = 1$ if $s_{in} = 0^{l-1}10^{n-l}$, $g(s_{in}) = 0$ for the remaining inputs and any value of l.

Making use of the previous example we can easily generalize the result for every function g that satisfies:

$$g(s_{in}) = \begin{cases} 1 & \text{if } |s_{in}| = 1 \\ 0 & \text{if } |s_{in}| \neq 1. \end{cases} \tag{4.9}$$

This function g is implemented using n multi-controlled $CNOT$ gates, the l-th gate activating for the state $\left|0^{l-1}10^{n-l}\right\rangle$, where $l \in \{1, \ldots, n\}$. As usual the target is the state $|-\rangle$. For this type of g we can obtain valid mappings for U_s if we use lemma 5 again, mapping $\mathcal{X}_0 \mapsto \{0^n\}$ and $\mathcal{X}_1 \mapsto \{0^{l-1}10^{n-l}\}$. The restriction $s_l = 1$ is not required any more and instead any value of l can be selected, since any l satisfies $g(0^{l-1}10^{n-l}) = 1$. The mapping f is then $f(x_1) \wedge u = \left[\sum_m^n 0^{l-1}(x_1)_m 0^{n-l}\right] \wedge 0^{l-1}10^{n-l} = \sum_m^n 0^{l-1}(x_1)_m 0^{n-l}$ for any entry m such that $s_m = 1$. These circuits need $n-1$ extra ancilla qubits and contain $2n + n \cdot 2(n-1)$ Toffoli gates at most. Then the number of T gates grows as

$O(n^2)$. The decryption process of these circuits then still has polynomial complexity but it is now quadratic instead of linear. Therefore their homomorphic evaluation is still feasible, even though it is less efficient compared to T-linear circuits with RCC.

In general, lemma 5 can be utilized to implement a valid U_s for these types of functions g that are constructed with multi-controlled $CNOT$ gates. As long as $g(s_{in}) = 1$ if $s_{in} = 0^{l-1}10^{n-l}$ for a certain l and $g(s_{in}) = 0$ for $s_{in} = 0^n$, the remaining values of $g(s_{in})$ are not important and the mapping obtained from lemma 5 can be used, taking into consideration that the restriction $s_l = 1$ is no longer needed. The reason that this lemma can be applied is simply that $g(0^n) = 0 \implies \mathcal{X}_0 \mapsto \{0^n\}$ and $g(0^{l-1}10^{n-l}) = 1 \implies \mathcal{X}_1 \mapsto \{0^{l-1}10^{n-l}\}$.

Another type of function g that can be considered in order to expand the previous results further is:

$$g(s_{in}) = \begin{cases} 1 & \text{if } |s_{in}| = n \\ 0 & \text{if } |s_{in}| \neq n \end{cases} \tag{4.10}$$

This function g is constructed using one multi-controlled $CNOT$ gate that activates for the state $|1^n\rangle$, where the target is the state $|-\rangle$ as usual. If lemma 6 is used to map $\mathcal{X}_0 \mapsto \{0^n\}$ and $\mathcal{X}_1 \mapsto \{1^n\}$ a valid implementation for U_s and this g is obtained that can be used for the special case where $|s| = 1$. These circuits contain $2n+2(n-1)$ Toffoli gates and $n-1$ ancilla qubits once more, so their number of T gates grows as $O(n)$. However, this implementation of U_s given by lemma 6 is limited by the condition $|s| = 1$. In order to go beyond this restriction, a different implementation of U_s that can be considered is applying n^2 Toffoli gates. Each of these Toffoli gates is controlled using the l-th qubit of the first register and the m-th qubit of the second register, for $l, m \in \{1, \ldots, n\}$. This way each qubit in the first register acts as the control qubit for n Toffoli gates and each of these gates makes use of a different control qubit from the second register. Applying this implementation of U_s we obtain the mapping $\mathcal{X}_0 \mapsto \{0^n\}$ and $\mathcal{X}_1 \mapsto \{1^n\}$ with the restriction that $|s| = n$. The mapping f is then $f(x_1) = \sum_{l=1,m=1}^{n} 0^{m-1}(x_1)_l 0^{n-m}$, where the sum is performed over all the values of l and m. These circuits need $n-1$ ancilla qubits and contain $2n^2 + 2(n-1)$ Toffoli gates. Thus, the number of T gates grows as $O(n^2)$ which means that the decryption procedure of these type of circuits still has polynomial complexity. Therefore their homomorphic evaluation, while less efficient than the evaluation of T-linear circuits with RCC, is still feasible. For an arbitrary s and this g, the implementation of U_s is performed making use of $n \cdot |s|$ Toffoli gates. Here, each of these Toffoli gates is controlled using the l-th qubit of the first register and the m-th qubit of the second register, for any entry $l \in \{\{1, \ldots, n\} : s_l = 1\}$, all the available values of $m \in \{1, \ldots, n\}$ and the mapping $f(x_1) = \sum_{l,m=1}^{n} 0^{m-1}(x_1)_l 0^{n-m}$. Once the usual gate decomposition is completed we find that the number of T gates grows as a polynomial, since at worst for $|s| = n$ it grows as $O(n^2)$ as previously explained.

Finally we can generalize these results for a g that satisfies $g(s_{in}) = 1$ if $s_{in} = 0^{n-i}1^i$ for $i \in \{1, \ldots, n\}$ and $g(s_{in}) = 0$ for the remaining values, i.e., $g(s_{in}) = 1$ just for one string s_{in} with any number of ones. Then g can be constructed using one multi-

controlled $CNOT$ gate that activates for the state $\left|0^{n-i}1^i\right\rangle$ and targets the state $|-\rangle$ once again. In this case U_s is implemented using $i \cdot |s|$ Toffoli gates. Each of these Toffoli gates is controlled using the l-th qubit of the first register, the m-th qubit of the second register, for any $l \in \{\{1,\ldots,n\} : s_l = 1\}$, any $m \in \{1,\ldots,n\}$ such that $(s_{in})_m = 1$ when $g(s_{in}) = 1$ and the mapping $f(x_1) = \sum_{l,m}^{n} 0^{m-1}(x_1)_l 0^{n-m}$. Of course the mapping $\mathcal{X}_0 \mapsto \{0^n\}$ and $\mathcal{X}_1 \mapsto \{0^{n-i}1^i\}$ is obtained from this U_s. Like in the previous case, the homomorphic evaluation of the circuits that result from this U_s and g is feasible because the number of T gates grows as a polynomial. This is because after the decomposition of the Toffoli gates and the multi-controlled $CNOT$ gate, the number of T gates grows as $O(n^2)$ at worst. This corresponds with the $|s| = i = n$ case described by the function 4.10 and implemented making use of n^2 Toffoli gates for U_s. Naturally the results obtained here also apply to a g where $g(s_{in}) = 1$ only for one string s_{in} whose ones are localized in any position, not necessarily restricted as they were in $s_{in} = 0^{n-i}1^i$.

In regards to the extension of these results for $k > 2$, the mapping that U_s implements for the general k case is

$$|x_1\rangle_1 \ldots |x_k\rangle_k |y\rangle \mapsto |x_1\rangle_1 \ldots |x_k\rangle_k \left|y \oplus (s_{x_1,\ldots,x_{k-1}} \cdot x_k)\right\rangle \quad \forall x_1,\ldots,x_k \in \{0,1\}^n, y \in \{0,1\}. \quad (4.11)$$

We want to implement this mapping using n multi-controlled $CNOT$ gates with k control qubits. We will only consider the function $g(s_{in}) = s \cdot s_{in}$, hence each application of G can be implemented with $|s|$ $CNOT$ gates at most just like in T-linear circuits with RCC. Then by selecting this g the problem narrows down to the mapping $x_1,\ldots,x_{k-1} \mapsto s_{x_1,\ldots,x_{k-1}}$ with the condition $s_{x_1,\ldots,x_{k-2}} \cdot x_{k-1} = s \cdot s_{x_1,\ldots,x_{k-1}}$ for some string $s \in \{0,1\}^n$. Let $f : \{0,1\}^n \to \{0,1\}^n$, $f(x_1,\ldots,x_{k-1})$, be this mapping. Then we have $s_{x_1,\ldots,x_{k-2}} \cdot x_{k-1} = s \cdot f(x_1,\ldots,x_{k-1})$. Like in the $k = 2$ case, the mapping from equation (4.11) can be generically performed as:

$$\begin{aligned} |x_1\rangle_1 \ldots |x_k\rangle_k |y\rangle_{\mathtt{T}} &\overset{1}{\mapsto} |f(x_1,\ldots,x_{k-1})\rangle_1 \ldots |x_k\rangle_k |y\rangle_{\mathtt{T}} \overset{2}{\mapsto} \\ &|f(x_1,\ldots,x_{k-1})\rangle_1 \ldots |x_k\rangle_k |y \oplus (f(x_1,\ldots,x_{k-1}) \cdot x_k)\rangle_{\mathtt{T}} \overset{3}{\mapsto} \\ &|x_1\rangle_1 \ldots |x_k\rangle_k |y \oplus (f(x_1,\ldots,x_{k-1}) \cdot x_k)\rangle_{\mathtt{T}}. \end{aligned} \quad (4.12)$$

Step 2 is just the application of n multi-controlled $CNOT$ gates with k control qubits. The j-th multi-controlled $CNOT$ gate is controlled using the j-th qubit of registers 1, 2,..., k and it targets register $\mathtt{T}$, for $j \in \{1,\ldots,n\}$. The issue is in regards to Steps 1 and 3. As in the $k = 2$ case, f must be a bijection since each step must be unitary. Also like before, we want to implement this map f using just single-qubit gates, so the X gates are allowed. Then the multi-controlled $CNOT$ gates that can be applied may be activated with either 0 or 1.

Studying this mapping in general was out of the scope of our publication [155], so we just give an brief example of a possible extension in the general k case. We can simply make use of the map $f(x_1, x_2, \ldots, x_{k-1}) = x_1 \wedge x_2 \wedge \cdots \wedge x_{k-1}$. In order to fulfil the

condition $s_{x_1,\ldots,x_{k-2}} \cdot x_{k-1} = s \cdot f(x_1, \ldots, x_{k-1})$ we require $|s| = n$ and we assume that $s_{x_1,\ldots,x_{k-2}} = x_1 \wedge \cdots \wedge x_{k-2}$, $s_{x_1,\ldots,x_{k-3}} = x_1 \wedge \cdots \wedge x_{k-3}$, $\ldots$, $s_{x_1} = x_1$. Then we have $s_{x_1,\ldots,x_{k-2}} \cdot x_{k-1} = s \cdot f(x_1, \ldots, x_{k-1}) = s \cdot (x_1 \wedge \cdots \wedge x_{k-1}) \implies (x_1 \wedge \cdots \wedge x_{k-2}) \cdot x_{k-1} = s \cdot (x_1 \wedge \cdots \wedge x_{k-1}) \implies |s| = n$. This mapping f is equivalent to simply applying n multi-controlled $CNOT$ gates with k control qubits. A single multi-controlled $CNOT$ gate with k control qubits can be decomposed into $2(k-1)$ Toffoli gates. If the n multi-controlled $CNOT$ gates are decomposed we then have $n \cdot 2(k-1)$ Toffoli gates and $k-1$ extra ancilla qubits, so the number of T gates grows as $O(n \cdot k)$. Then the decryption process of the QHE for these circuits still has polynomial complexity. Thus their homomorphic evaluation is still efficient, just not as much as T-linear circuits with RCC in the $k = 2$ case. Therefore even in the general k case some extensions of T-linear circuits with RCC, relevant in the context of QHE, can still be found.

We conclude this chapter with a possible application of T-linear circuits with RCC. These circuits could be used as a test to asses the correct functioning of a quantum computer. The simulation of these circuits can not be performed efficiently using classical computers due to the T gates they contain. Nevertheless such circuits require less computational power than circuits containing an exponential number of T gates. We are currently in the so called "noisy intermediate-scale quantum" era (NISQ) in which the best quantum computers available contain at most a few hundred qubits but have not achieved fault-tolerance yet and are still not large enough to use the full computational power offered by quantum mechanics. Seeing that the number of T gates grows as $O(n)$ for T-linear circuits with RCC, these circuits could be implemented in quantum computers with a substantial number of qubits in order to certify their correct functioning by ensuring that the correct s is returned and it would be less computationally complex than implementing other algorithms where the number of T gates is higher than linear. Besides this application, T-linear circuits with RCC could be great circuits in which to test QHE experimentally due to their linear number of T gates.

4.3 Conclusions

- In this chapter we have defined and characterized a certain group of quantum circuits, called T-linear circuits with RCC, that solve some particular cases of the recursive Bernstein-Vazirani problem for $k = 2$. The main characteristic of these circuits is simplifying the quantum oracle required to solve the problem from multi-controlled $CNOT$ gates to Toffoli gates. This decreases the T gate complexity of the oracle. In particular, the number of T gates required to solve the problem grows linearly with the number of qubits used to represent the secret key s. This feature is important in the realm of quantum homomorphic encryption.

- In Liangs's [127] QHE schemes the decryption procedure depends only on the number of $T/T^\dagger$ gates in the circuit, so the schemes are only efficient for circuits

with a polynomial number of $T/T^\dagger$ gates. Since the number of $T/T^\dagger$ gates in T-linear circuits with RCC grows linearly as $O(n)$, they constitute a perfect example of quantum circuits that can be evaluated homomorphically with non interaction and perfect security in an efficient manner.

- On the other hand, a circuit that solves the recursive BV problem for $k = 2$ that can not be constructed using a polynomial number of $T/T^\dagger$ gates would not be suitable to be implemented applying these QHE schemes, unlike T-linear circuits with RCC. As it was shown such a circuit requires an exponential number of $T/T^\dagger$ gates, which means its decryption procedure would be inefficient due to the quasi-compactness of the QHE schemes. Still, the homomorphic implementation of the recursive BV algorithm for general cases could be performed using a different QHE scheme that has compactness in exchange of less restrictive conditions, such as downgrading the security level from perfect security to computational security. A scheme that satisfies these requirements is the one proposed by Mahadev [125], so studying the particular application of this scheme to the general cases of the recursive BV problem could serve as a future line of research.

- We analysed some other possibilities of implementing the oracle G making use of different functions g. In these cases, we found that some implementations of U_s using only Toffoli gates instead of multi-controlled $CNOT$ gates can still be constructed. Their number of $T/T^\dagger$ gates grows polynomially so their homomorphic evaluation is still efficient, although not as much as in T-linear circuits with RCC. Moreover, these circuits need $n-1$ extra ancilla qubits due to the decomposition of the multi-controlled $CNOT$ gates used to implement G. An extension of T-linear circuits for the general case k of the recursion was also found for a simple case. In these circuits the number of $T/T^\dagger$ gates grows as $O(n \cdot k)$. Therefore their homomorphic implementation is efficient too, taking into account that $k - 1$ ancilla qubits are needed in this case.

- Studying the extension of T-linear circuits with RCC for higher orders of the recursion in more detail and analysing the efficiency of their homomorphic encryption could serve as future work.

5 The Grover algorithm

5.1 Background

One of the most basic tasks that a computer can perform is searching. There are plenty of different applications and problems based on searching, such as retrieving data from a database, finding the maximum or minimum value in an array, factoring an integer, problems about constraint satisfaction (like the Boolean satisfiability problem), problems about combinatorial optimization and many more.

Perhaps the simplest search problem can be considered the unstructured search problem. It consists in finding a desired element in a database with no structure. Due to this condition, for a classical computer there is no better algorithm than checking each element in the database until the desired element is found. Assuming there are N elements in this database, this process takes at most $O(N)$ queries. Surprisingly, it turns out quantum mechanics offer a better solution to this problem, which is the quantum algorithm proposed by Grover [156]. Grover's search algorithm is considered one of the most successful algorithms in quantum computing, since it showed the advantage quantum computers have over classical computers in the unstructured database search problem. Grover's algorithm improved the query complexity from $O(N)$, offered by the best classical algorithm, to just $O(\sqrt{N})$. This constitutes a quadratic speed-up.

Due to its importance, Grover's algorithm has been studied extensively. In [157] a family of quantum search algorithms is investigated, showing that Grover's algorithm holds a distinguished place in this family, as it is the most optimal algorithm in this family. Also, the algorithm has been used as the basis for many different quantum algorithms and applications, such as quantum counting [158], which is an algorithm that efficiently counts the number of solutions for a given search problem, or the collision problem [159]. Surprisingly it also shares an isomorphism with classical kinematics collisions [160]. Grover's algorithm also has application regarding symmetric encryption, since it can be applied to the problem of finding the secret key. The quadratic speed-up that the algorithm offers over a classical exhaustive key search seems to be the most relevant quantum cryptanalytic impact for block ciphers [161]. In this chapter we will review the Grover algorithm in order to study its homomorphic implementation in the next chapter in great detail.

5.2 Grover's algorithm review

As we have mentioned in the unstructured search problem, a list of N unordered elements is given. The components of this list are labelled from 0 to $N-1$ without any loss of generality. There is an element in this list that has to be found denoted by w. Since there is no structure in this list all the elements have to be checked in order to find the desired element in the classical scenario. This process takes on average $N/2$ attempts and in the worst case scenario it takes N tries, making the complexity of the classical problem $O(N)$.

As we mentioned, in the quantum algorithm given by Grover the searched element can be found in just $O(\sqrt{N})$ steps which constitutes a quadratic speedup. For simplicity, the elements of the list can be written as $N = 2^n$ for some integer n. This way we can represent the N elements of the list using n qubits.

The algorithm begins with n qubits in the state $|0\rangle$. The first step in Grover's algorithm is state preparation. In this step, n Hadamard gates are applied to each qubit in order to obtain an uniform superposition of all possible n bit strings:

$$|s\rangle = H^{\otimes n} \left|0^{\otimes n}\right\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle \,. \tag{5.1}$$

In the next step, the oracle expressed by $U_w = I - 2\,|w\rangle\langle w|$ is applied. Given any state $|x\rangle$ as input, the oracle will output:

$$U_w\,|x\rangle = \begin{cases} |x\rangle & \text{if } x \neq w \\ -\,|x\rangle & \text{if } x = w \end{cases} \tag{5.2}$$

This oracle changes the amplitude of the searched state while leaving the rest unaffected. The action of the oracle on the superposition state $|s\rangle$ is given by:

$$U_w\,|s\rangle = \frac{1}{\sqrt{N}} \sum_{\substack{x=0 \\ x\neq w}}^{N-1} |x\rangle - \frac{1}{\sqrt{N}}\,|w\rangle \,. \tag{5.3}$$

Next, the diffusion operator given by $U_s = 2\,|s\rangle\langle s| - I$ is applied to $U_w\,|s\rangle$. This operator flips the amplitudes around the mean. Thus, the amplitude of each state decreases except in the case of searched state that experiments an increase in its amplitude. This process of amplifying the amplitude of the desired state is known as amplitude amplification. Grover's algorithm consists on applying the operators U_w and U_s iteratively.

The algorithm has a well known geometric interpretation based on two reflections that lead to a rotation around an angle θ in a plane. In this interpretation, an orthonormal coordinate system is used. As the superposition state $|s\rangle$ is not orthogonal to the searched

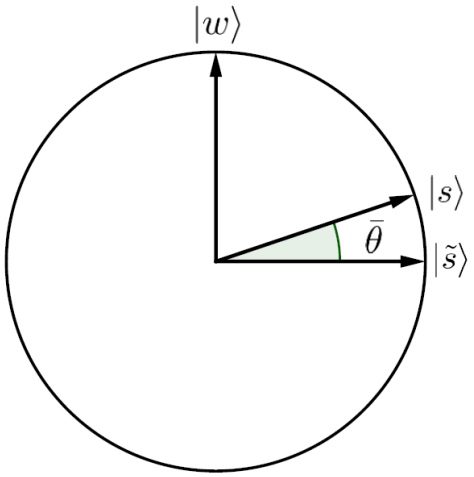


Figure 5.1: Circle defined by eq. (5.5). The basis is given by the states $|\tilde{s}\rangle$ and $|w\rangle$. The angle $\bar{\theta}$ that this state forms with the horizontal axis is given by equation (5.6).

state $|w\rangle$, we introduce a new state called $|\tilde{s}\rangle$ that is perpendicular to $|w\rangle$ defined by:

$$|\tilde{s}\rangle = \frac{1}{\sqrt{N-1}} \sum_{\substack{x=0 \\ x\neq w}}^{N-1} |x\rangle \,. \tag{5.4}$$

The state $|\tilde{s}\rangle$ is then obtained from the state $|s\rangle$ by removing the desired state $|w\rangle$ and rescaling so that the state $|\tilde{s}\rangle$ is still normalized. The states $|\tilde{s}\rangle$ and $|w\rangle$ form a basis and so any state can be expressed in terms of an angle θ:

$$|\theta\rangle = \cos(\theta)\,|\tilde{s}\rangle + \sin(\theta)\,|w\rangle \,. \tag{5.5}$$

Both the U_w and U_s operators keep the resulting state in the circle defined by eq (5.5). This circle is shown in figure 5.1. The first step of the algorithm is also represented in figure 5.1 since the state shown is the starting superposition state $|s\rangle$. The angle $\bar{\theta}$ that this state forms with the horizontal axis is given by equation (5.6).

$$\bar{\theta} = \arcsin \langle s|w\rangle = \arcsin \frac{1}{\sqrt{N}}. \tag{5.6}$$

The next step is applying the oracle U_w. Geometrically this is equivalent to a reflection about the $|\tilde{s}\rangle$ axis since the amplitude of the state $|w\rangle$ experiments a change of its sign. The last step to complete an iteration of the algorithm is applying U_s. This operator reflects the state about the state $|s\rangle$ as shown in figure 5.2.

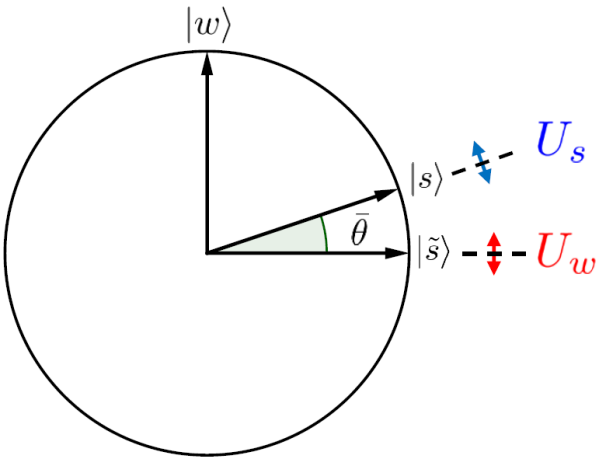


Figure 5.2: The oracle U_w reflects about $|\tilde{s}\rangle$ and the diffusion operator U_s reflects abound $|s\rangle$.

Two consecutive reflections constitute a rotation. The angle corresponding to this rotation is $2\bar{\theta}$ as it can be seen in figure 5.3. Thus, the combined action of both operators on any state is given by eq. (5.7).

$$U_s U_w |\theta\rangle = \left|\theta + 2\bar{\theta}\right\rangle . \tag{5.7}$$

If both operators are applied to the initial state $|s\rangle$, we obtain $U_s U_w |s\rangle = \left|\bar{\theta} + 2\bar{\theta}\right\rangle$.

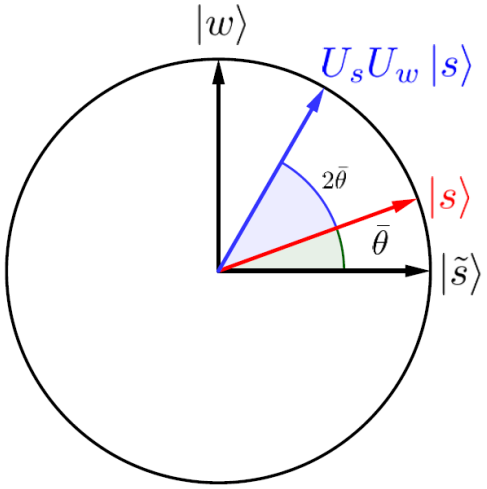


Figure 5.3: The combination of U_w and U_s rotate the initial state by an angle of $2\bar{\theta}$.

The combined action of the operators is then repeated iteratively in order to rotate the initial state $|s\rangle$ closer to the desired state $|w\rangle$. This means performing a rotation from $\theta = \bar{\theta}$ to $\theta = \pi/2$ in steps of $2\bar{\theta}$. After $O(\sqrt{N})$ iterations, the amplitude of the

desired state $|w\rangle$ reaches its maximum and so the probability of obtaining $|w\rangle$ in a quantum measurement approaches 1. Thus, the final step of the algorithm is performing a quantum measurement of the n qubits that returns the searched element of the list, w, with high probability.

In case there are m elements that have to be found in the list instead of just one, the steps of the algorithm remain unchanged: preparation of the superposition state $|s\rangle$, iterative application of the operators U_w and U_s and measurement of the n qubits. In this case, the number of steps required to obtain the desired elements is bounded by $O\left(\sqrt{\frac{N}{m}}\right)$ [162].

6 Homomorphic Encryption of the Grover Algorithm

6.1 Motivation

Grover's algorithm is one of the most successful quantum algorithms. It serves as the basis of many other algorithms and applications. Following our quest to find quantum algorithms that can be homomorphically encrypted in an efficient manner, it is natural to study Grover's algorithm next, due to its great importance and wide variety of applications. Furthermore, as most quantum computers will be accessible through the cloud, the security of the computations processed in these environments will become paramount. If homomorphically encrypted quantum search could be performed efficiently it would represent an important development, since not only we would have a faster search than what any classical computer can perform, but also a perfectly search algorithm protected by the fundamental properties of quantum mechanics.

Taking into account these motivations, we have combined Liang's QHE schemes and Grover's algorithm to perform homomorphically encrypted quantum search. Recall that the main feature of Liang's schemes [127] is that despite being quasi-compact, they allow the efficient homomorphic evaluation of any quantum circuit with low $T/T^\dagger$-gate complexity with perfect security. Due to this trait, they are suitable for circuits with a polynomial number of $T/T^\dagger$-gates. Previous works that applied these QHE schemes to Grover search were performed by Gong et al. [128]. In particular, they implemented a ciphertext retrieval scheme based on the Grover algorithm using these QHE schemes. They performed simulations and experiments using Qiskit and IBM's quantum computers as an example of the scheme. However, the Grover search simulations they constructed were made on a circuit containing two qubits that only needed Clifford gates. Also, no analysis of the homomorphic implementation of a general Grover circuit was done. Regarding the implementation of $T/T^\dagger$ gates, only one of these gates was implemented

In our publication [163], we have improved these previous results by proposing a homomorphic implementation of Grover's algorithm using Liang's schemes that allows the homomorphic evaluation of any Grover circuit efficiently, provided we have access to $n-2$ ancilla qubits besides the n qubits used in Grover search. As an example, a homomorphic Grover search circuit for 3 qubits has been simulated using Qiskit. Our simulation of the homomorphic evaluation of a Grover circuit is more complex than

those performed in [128] because it contains $T/T^\dagger$ gates. Besides this, we showed how to apply classically controlled-S gates in the Qiskit simulation to correctly evaluate multiple $T/T^\dagger$ gates and account for every possible measurement result. The results obtained from our simulation can always be correctly decrypted after obtaining the final key for each qubit involved. We also discuss the $T/T^\dagger$-gate complexity of the algorithm unlike previous works [128]. The fact that this number grows slower than any linear function will demonstrate that our proposal for a general Grover homomorphic implementation is efficient. In the particular case of a database with only one marked element, different quantum search algorithms that reduce the number of quantum gates needed to solve the problem have been proposed by Arunachalam and de Wolf [164] and Briański et al. [165]. We also show that these algorithms can be homomorphically evaluated in a more efficient manner than Grover's algorithm. Finally, we study the efficiency of the homomorphic evaluation of more practical quantum search algorithms used in quantum key attacks for symmetric encryption schemes.

6.2 Homomorphic Grover simulation in Qiskit

6.2.1 Qiskit setup

In this section a simulation of a homomorphic evaluation of a Grover circuit is shown. The simulation has been performed in IBM Qiskit [54].

Suppose that the client wants to solve the unstructured search problem using Grover's algorithm. He wants to do it without the server learning anything about his data so the client decides to use quantum homomorphic encryption to preserve its security. In this particular case, he wants to find two marked elements in a list containing eight elements.

The circuit that the client wants to evaluate is shown in figure 6.1. The qubits are denoted by q_0, q_1, ..., q_{n-1}. Notice that in Qiskit the states are represented as $|q_{n-1}...q_1q_0\rangle$ so the circuit in figure 6.1 should be read starting from q_2 and finishing at q_0. The length of the list is $N = 8$ which means that $n = \log_2 N = 3$ qubits are needed. This circuit finds two elements in the list: the states $|011\rangle$ and $|101\rangle$ (Qiskit notation is being used here). Then Grover's algorithm for two marked elements is applied. The reason why this circuit was selected is related to the complexity of the simulation. For $N = 8$ and $m = 2$ only one iteration is needed and the circuit still requires the use of $T/T^\dagger$-gates. This makes the homomorphic evaluation not trivial but still simple enough to be simulated.

All the steps of the circuit are separated by a grey barrier in figure 6.1. The first step applies 3 H-gates to each qubit in order to create the superposition state $|s\rangle$. The next step is the application of the oracle U_w, so the states $|011\rangle$ and $|101\rangle$ are marked using two controlled-Z gates. The third step is the application of the diffusion operator U_s using a CCZ gate (a controlled Z-gate that has 2 control qubits and one

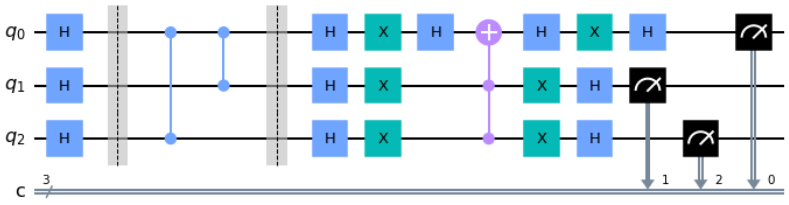


Figure 6.1: The quantum circuit that the client wants to evaluate. It implements a Grover search for $n = 3$ qubits, $N = 8$ elements and $m = 2$ marked items in Qiskit. The two marked items are $|011\rangle$ and $|101\rangle$. Each step of the algorithm is separated by a grey barrier.

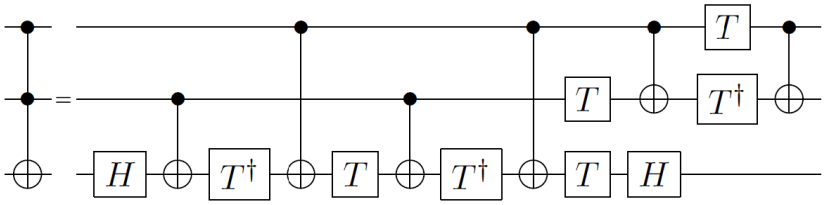


Figure 6.2: Toffoli gate decomposition into 7 $T/T^\dagger$-gates.

target qubit) surrounded by H and X gates. In figure 6.1, the CCZ gate has already been decomposed into a Toffoli gate which has its target qubit surrounded by two H-gates. Using the relation $Z = HXH$, any Z-gate can be implemented using an X gate surrounded by two H gates. The reason for this decomposition is that in order to implement this circuit homomorphically, it has to be decomposed into gates of the set $\mathcal{G} = \{X, Z, H, S, CNOT, T, T^\dagger\}$. This means that each controlled-Z gate from the oracle U_w has to be substituted by a $CNOT$ gate surrounded by two H-gates just like the CCZ gate was substituted by a Toffoli gate surrounded by H-gates. The last gate that has to be transformed is the Toffoli gate. Recall that it can be decomposed into 7 $T/T^\dagger$-gates as seen in figure 6.2.

After decomposing the two controlled-Z gates, the CCZ gate and the Toffoli gate into gates of the set $\mathcal{G}$, a new circuit represented in figure 6.3 is obtained. This circuit is now composed only of gates from the set $\mathcal{G}$ and so it can be evaluated homomorphically using the QHE scheme described in chapter 2. The total number of gates of the circuit is $l = 35$. The sequence of gates is $G[1] = H_0$, $G[2] = CNOT_{2,0}$, ... and so on until the last three gates which are $G[33] = H_0$, $G[34] = H_1$, $G[35] = H_2$.

The first step of the QHE scheme that the client must perform is generating the secret key, so the client starts by generating the random bits a_0, $b_0 \in \{0,1\}^3$ to obtain his secret key $sk = (a_0, b_0)$. In the simulation $n = 3$. We chose $a_0 = a_0(0)a_0(1)a_0(2) = 111$ and $b_0 = b_0(0)b_0(1)b_0(2) = 111$ where $a_0(n-1)$ and $b_0(n-1)$ refer to the values of the key

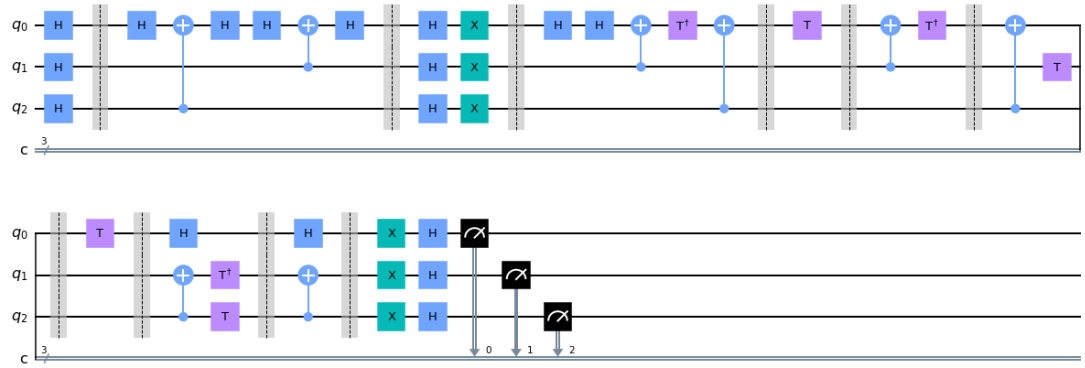


Figure 6.3: Compilation of the circuit that the client wants to evaluate decomposed using gates from the set $\mathcal{G}$.

for q_{n-1}. This way we have that for q_0 the keys are $(a_0(0), b_0(0)) = (1, 1)$. For q_1 and q_2 the keys are $(a_0(1), b_0(1)) = (1, 1)$ and $(a_0(2), b_0(2)) = (1, 1)$ respectively. Of course, any value for the keys can be chosen at this stage. Next, the first step of Grover's algorithm, state preparation, is performed using three H gates to obtain the superposition state $|s\rangle$. Then $|s\rangle$ is encrypted using the secret key sk. This means that all the qubits are encrypted using X^1Z^1 since the keys for each qubit are $(1, 1)$. Now that the data has been encrypted by the client, it is sent to the server.

The server starts by generating as many EPR pairs as $T/T^\dagger$ are in the circuit. Since $M = 7$ then seven entangled pairs have to be generated. An EPR pair can be easily constructed just using a H-gate on the first qubit and a $CNOT$ gate in which the target is the second qubit. Since 7 entangled pairs are used this means that 14 additional qubits are then needed in the simulation.

The simulation we performed contains 17 qubits in total. An image containing such a number of qubits would be too large to show properly so we show an equivalent circuit to the one that we used in our Qiskit simulation in figure 6.4. Besides the Qiskit simulation, we also calculated the value of each key before any S^a-rotated Bell measurement is performed using a Python script that takes the initial keys and calculates the final keys for all possible S^a-rotated Bell measurement results, using the key updating rules of the gates from the set $\mathcal{G}$. There are some details that need further explanation. As we mentioned, 14 extra qubits are needed to evaluate the 7 $T/T^\dagger$-gates. However in figure 6.4 there are only two extra qubits denoted by q_3 and q_4. Instead of using 7 EPR pairs and performing 7 S^a-rotated Bell measurements, we make use of Qiskit reset operation which is denoted by the gate $|0\rangle$ in the circuit. This gate simply turns the state back to $|0\rangle$, so we can perform a S^a-rotated Bell measurement, use the reset operation on the qubits that were measured and entangle them again using a H gate and a $CNOT$ gate. The reason this is done is simply to obtain a more compact image, because instead of using 17 qubits only 5 are needed. The number of S^a-rotated Bell measurements remains

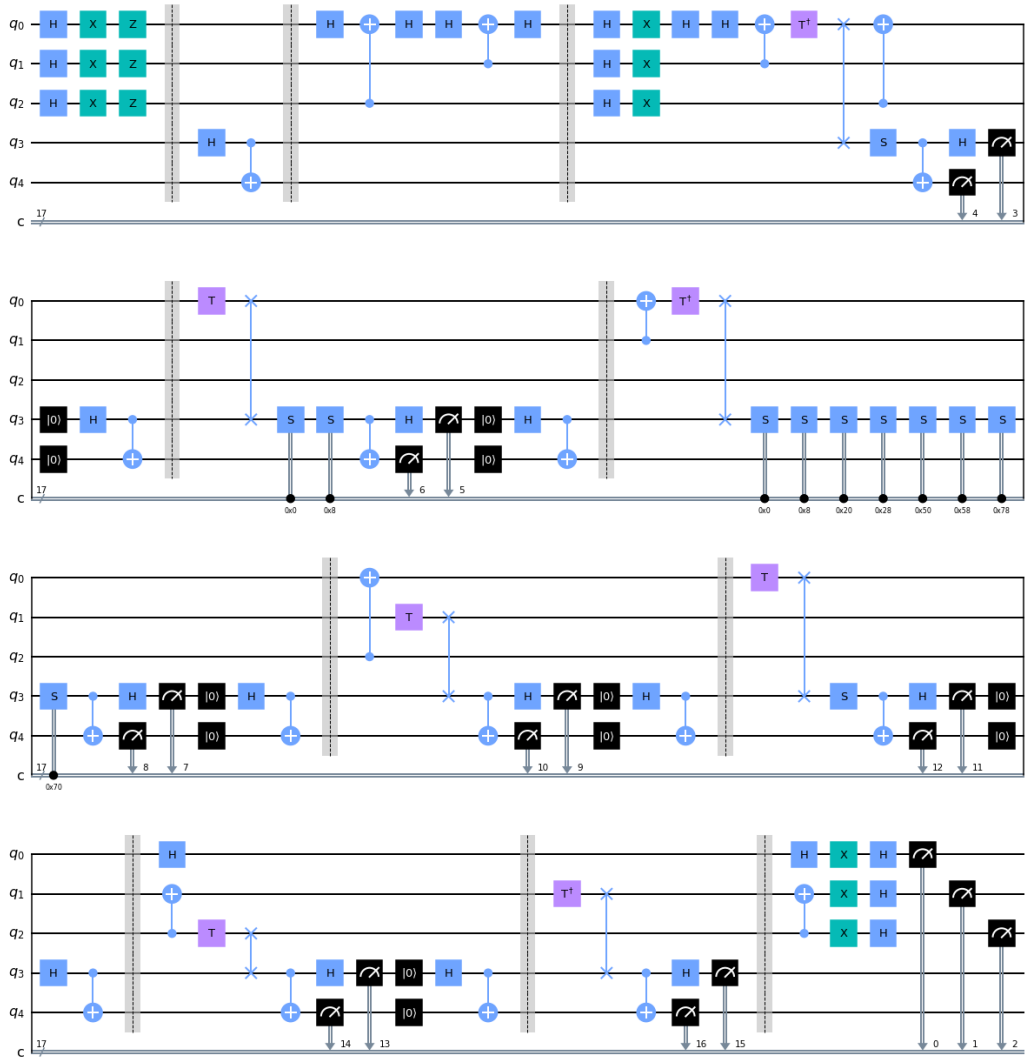


Figure 6.4: Simulated circuit in Qiskit. This circuit is a simplified version of the simulation performed in order to make the image more compact.

the same: $M = 7$.

The main issue that has to be solved is correcting the phase errors and for this purpose classically controlled S-gates are used in the circuit. These gates are denoted by a S-gate on the target qubit and a control located in the register denoted by c in the circuit, below q_4. This c register contains the results of all the quantum measurements so it is made of classical bits. An example of these classically controlled S-gates are the two gates that act on q_3 after the first T (second $T/T^\dagger$) gate of the circuit. In chapter 2 it was explained that in the decryption process the client needs to obtain the

corresponding value of a_j before performing the correct S^a-rotated Bell measurement. This means that to perform the correct simulation one cannot simply apply S^1-rotated Bell measurements in all cases, since there would be cases where no S gate should be applied in the measurement. Instead each possible measurement result from all the previous S^a-rotated Bell measurements has to be taken into account and used to calculate the corresponding a for the next measurement so the simulation returns the correct values for all possible scenarios. Using the key updating Python script mentioned previously, the a_j relative to a measurement is calculated, one for each possible result of all the previous measurements. Then for each $a_j = 1$ a controlled S gate is used, where the control is a classical bit string sequence stored in c that contains one result of all the previous measurements. As an example notice that for the second $T/T^\dagger$ in figure 6.4 there are two control-S gates. This is because in the previous $T/T^\dagger$-gate four measurement results are obtained. After key updating two of them lead to $a_{17} = 1$ for this second $T/T^\dagger$ gate (which is the 18th gate of the circuit), so two classical bit strings stored in c have to be used as the control for two control-S gates. The third $T/T^\dagger$ in figure 6.4 contains 8 control-S gates for the same reasons, only now there are 16 possible measurement results because there are two previous measurements. The second reason why the circuit shown in figure 6.4 is not exactly the circuit that was simulated is the number of these control-S gates that were used. For the last $T/T^\dagger$ of the circuit 2048 control S gates were used, so the resulting image would be too large to show here. The only two differences between the circuit shown in figure 6.4 and the simulated circuit are that the latter used 17 qubits with no reset operation and that it contained all the necessary control S gates in the measurements. Nevertheless the circuit in figure 6.4 is still useful to be presented here because it illustrates all the relevant elements of the simulation: the encryption using X and Z gates according to the secret key, the 7 S^a-rotated Bell measurements along the entangled qubits and some classically controlled S gates that are needed in these measurements.

We want to remark that the main issue about evaluating the $T/T^\dagger$ gates is related to the particular homomorphic encryption scheme used and not to the implementation of the gates on the IBMQ platform. It is already assumed that the $T/T^\dagger$ gates can be implemented with zero error in the QHE scheme. However, all the S^a-rotated Bell measurements must be performed in order by the client. Thus the complexity of the decryption process depends necessarily on the number of $T/T^\dagger$ gates contained in the evaluated circuit, even if the $T/T^\dagger$ gates are implemented with zero error.

We can proceed with the remaining steps now that these details have been properly discussed. After evaluating all the quantum gates in the circuit using the key updating rules, the server would send the encrypted qubits back to the client, the key-updating functions and all the entangled qubits. We want to remark that the 7 S^a-rotated Bell measurements in the protocol are performed by the client once he receives all the data from the server and updates its keys accordingly. The last step of the scheme is measuring the qubits q_0, q_1 and q_2 and using the final key dk to decrypt the measurement results.

6.2.2 Simulation results

The results from the simulation are shown in figure 6.5. The horizontal axis represents the measurement result obtained and the vertical axis shows the probability of obtaining each result. The circuit was executed 10 times in order to obtain a compact image. As it can be seen, every state in figure 6.5 is obtained with probability 0.1%. Each measurement result contains the results obtained from the 7 S^a-rotated Bell measurements and the measurement values of q_0, q_1 and q_2. The first 14 bits in each result represent the results of the 7 S^a-rotated Bell measurements. If we start reading from the left, the first two bits refer to the last measurement and the last two bits refer to the first measurement. This is because the first S^a-rotated Bell measurement is performed on qubits q_4 and q_3. The next S^a-rotated Bell measurement is performed on qubits q_6 and q_5. This continues until the last S^a-rotated Bell measurement which is performed on qubits q_16 and q_15. Then the last 3 remaining bits represent q_2, q_1 and q_0. The probability of obtaining each state of figure 6.5 is not the main focus here. Instead what we are interested in is obtaining the correct result after decryption given a measurement result of q_0, q_1 and q_2. In order to show the results obtained for the encrypted qubits more clearly, we also present figure 6.6 which contains the measurement results of just q_2, q_1 and q_0. The encrypted state 001 was obtained once, the states 010, 100 and 111 were obtained two times and finally the state 011 was obtained three times. These are the measurement results that have to be decrypted using the final keys.

The decryption process is better understood by means of an example. If we take the first result from figure 6.5 which is 00101000010010001, the last 3 bits 001 represent the encrypted value of Grover's algorithm result. However, the state 001 is not one of the solutions of the Grover circuit used. On the other hand the first fourteen bits, 00101000010010, are the results of the seven Bell measurements. Using these values and the Python script for key updating based on the rules from chapter 2, the final key dk can be obtained. In this case the final key is $(a_{\text{final}}(0), b_{\text{final}}(0)) = (0, 1)$ for q_0, $(a_{\text{final}}(1), b_{\text{final}}(1)) = (1, 1)$ for q_1 and $(a_{\text{final}}(2), b_{\text{final}}(2)) = (0, 1)$ for q_2 so $dk = (a_{\text{final}}, b_{\text{final}}) = (a_{\text{final}}(0)a_{\text{final}}(1)a_{\text{final}}(2), b_{\text{final}}(0))b_{\text{final}}(1)b_{\text{final}}(2)) = (010, 111)$. At this point the result of the algorithm can be decrypted. Since the measurement of q_0, q_1 and q_2 was performed before decrypting the qubits using the final key, to decrypt a classical result the operation is simply $a_{\text{final}} \oplus r_c$, the bitwise modulo 2 sum where r_c is the classical bit string obtained from the quantum measurement. Therefore for this example where the encrypted result of Grover's algorithm is $r_c = 001$ we have $010 \oplus 001 = 011$. Recall that for the particular Grover circuit that was evaluated the correct results are either 011 or 101, so the decrypted result that was obtained is indeed correct.

For the remaining 9 results obtained from figure 6.5, the same process was performed to obtain the decrypted result. Table 6.1 shows the final decrypted results along the final key of every qubit for each result obtained from the Qiskit simulation. As it can be seen every simulation result is correctly decrypted into 011 or 101, the desired elements from the Grover search problem. The bit string 011 was obtained 7 times and 101 was

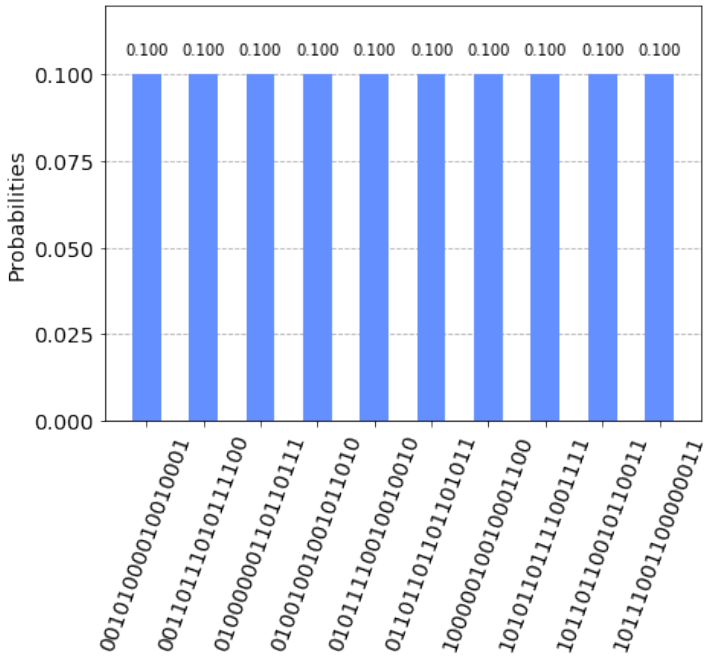


Figure 6.5: Results of the homomorphic Grover circuit simulation. This histogram shows 10 different measurement results. Each bar contains the results of the 7 S^a-rotated Bell measurements and the final encrypted states of qubits q_2, q_1 and q_0.

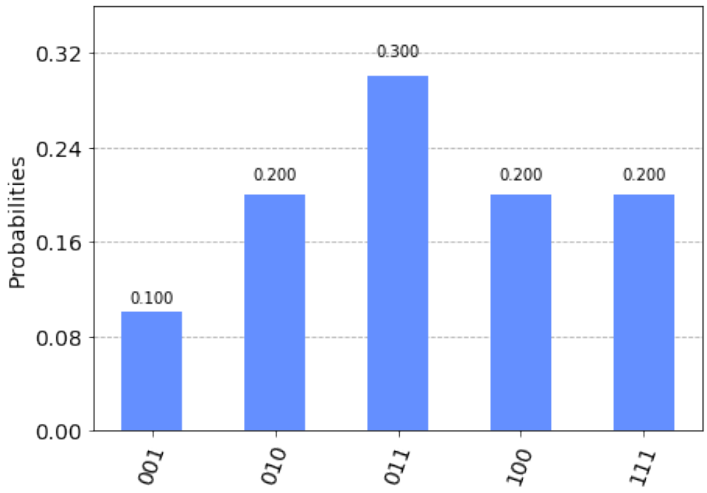


Figure 6.6: Results of the homomorphic Grover circuit simulation. This histogram shows 10 different measurement results. Only the final encrypted states of qubits q_2, q_1 and q_0 are included.

obtained the remaining 3. If the simulation is performed more times, the results get closer to the expected 50% chance of obtaining each state. Therefore the simulation obtains the correct results of the algorithm after decryption for all possible cases, demonstrating the correctness of the QHE scheme.

Simulation result	S^a-rotated Bell measurements	Encrypted result	Final q_2 key	Final q_1 key	Final q_0 key	Decrypted result
00101000010010001	00101000010010	001	(0,1)	(1,1)	(0,1)	011
00110111010111100	00110111010111	100	(0,1)	(0,0)	(1,0)	101
01000000110110111	01000000110110	111	(0,0)	(1,0)	(0,1)	101
01001001001011010	01001001001011	010	(0,0)	(0,0)	(1,0)	011
01011110010010010	01011110010010	010	(0,0)	(0,1)	(1,1)	011
01101101101101011	01101101101101	011	(0,1)	(0,1)	(0,0)	011
10000010010001100	10000010010001	100	(1,0)	(1,0)	(1,1)	011
10101101111001111	10101101111001	111	(1,1)	(0,0)	(0,0)	011
10110110010110011	10110110010110	011	(0,1)	(0,1)	(0,0)	011
10111001100000011	10111001100000	011	(1,1)	(1,0)	(0,1)	101

Table 6.1: Results from figure 6.5. For each result obtained from the simulation the following information is given: its 7 Bell measurements results, the encrypted Grover search result, the corresponding final key for each qubit q_2, q_1 and q_0 and the final Grover search result.

We want to remark that the physical implementation of the IBMQ gates does not significantly impact the efficiency of the QHE scheme. The physical gates that the IBM quantum computers can implement are limited to some single-qubit gates such as $\sqrt{X}$ and one two-qubit gate. On the other hand, the QHE scheme applies the set of gates $\mathcal{G} = \{X, Z, H, S, CNOT, T, T^\dagger\}$ that can perform universal quantum computation and the scheme assumes that all these gates can be performed in the particular quantum platform that the server is using. The IBMQ platform was used as an example to demonstrate the QHE scheme but the scheme is not restricted to the IBMQ particular platform hardware. In any case even though the IBM quantum computers can only implement a limited number of physical gates, these physical gates implement all the logical gates contained in the evaluated circuit. Then the number of logical $T/T^\dagger$ gates remains the same and so since the amount of S^a-rotated Bell measurements would be the same, the efficiency of the QHE protocol would remain the same.

To obtain the number of physical $T/T^\dagger$ gates contained in the whole algorithm, we only need to calculate how many physical gates are needed to implement one T gate and multiply that number by the number of T gates needed for the whole algorithm, so we have $T_{\text{physical}} \cdot M$, where T_{physical} is the number of physical IBMQ gates required to construct one T gate and M is the number of T gates contained in the algorithm. In the next section, we calculate this value of M for the whole Grover algorithm. This calculation serves to determine the efficiency of the homomorphic evaluation of the algorithm.

6.3 Grover search

In this section the $T/T^\dagger$-gate complexity of Grover's algorithm is discussed, along with a proposal to evaluate any Grover circuit homomorphically. For this purpose, the oracle and the diffusion operator have to be decomposed into gates from the set $\mathcal{G} = \{X, Z, H, S, CNOT, T, T^\dagger\}$. We will consider the case where there is only one marked element in the list and then generalize for the case with m marked elements.

Starting with the oracle U_w, a quantum state that is the solution of the problem is marked. The oracle can be constructed using a multi-controlled Z-gate with $n-1$ control qubits that adds a negative phase only to the desired quantum state. The control qubits should only activate for the desired state. This means that for each qubit in which $|w\rangle$ contains a 0 the corresponding control qubit has to be surrounded by two X gates, one before the control qubit. This is also applies to the target qubit. As we have seen, a controlled Z gate can easily be decomposed into a $CNOT$ gate by surrounding the target qubit with two H gates due to the identity $Z = HXH$, so the oracle can be implemented with a multi-controlled $CNOT$ gate with $n-1$ control qubits.

On the other hand, the diffusion operator is independent of the searched element and is the same for every Grover circuit. Making use of $U_s = 2|s\rangle\langle s| - I$ and $|s\rangle = H^{\otimes n}|0^{\otimes n}\rangle$ we have:

$$U_s = 2|s\rangle\langle s| - I = H^{\otimes n}\left(2\left|0^{\otimes n}\right\rangle\left\langle 0^{\otimes n}\right| - I\right)H^{\otimes n}. \tag{6.1}$$

This means that the operator $2|0^{\otimes n}\rangle\langle 0^{\otimes n}| - I$ has to be surrounded by $2n$ H-gates. If a global (-1) phase is applied to the previous operator, we have $I - 2|0^{\otimes n}\rangle\langle 0^{\otimes n}|$. This operator is basically another oracle in which the desired state is the $|0^{\otimes n}\rangle$ state since $[I - 2|0^{\otimes n}\rangle\langle 0^{\otimes n}|]|0^{\otimes n}\rangle = -|0^{\otimes n}\rangle$ and $[I - 2|0^{\otimes n}\rangle\langle 0^{\otimes n}|]|x\rangle = |x\rangle$ for any $|x\rangle \neq |0^{\otimes}\rangle$. Then it can be implemented using a multi-controlled Z gate with $n-1$ control qubits, where each qubit is surrounded by two X-gates. This multi-controlled Z gate can be decomposed like the one used in U_w. Therefore the diffusion operator can be implemented using a multi-control $CNOT$ gate with $n-1$ control qubits and a target qubit surrounded by 2 H gates, where all the n qubits are surrounded by $2n$ X gates and $2n$ H gates.

As an example, the whole algorithm is shown in figure 6.7 for the state $|1001\rangle$, in order to make clear how the oracle and diffusion operator are implemented using gates from the set $\mathcal{G} = \{X, Z, H, S, CNOT, T, T^\dagger\}$. The only gates that are not present in this set are the multi-controlled $CNOT$ gates with 3 control qubits.

The number of $T/T^\dagger$ gates is the relevant parameter that determines if a certain algorithm can be efficiently encrypted using Liang's QHE scheme. Then the only gates that have to be taken into account from the oracle and the diffusion operator are the multi-controlled $CNOT$ gates, since these gates are the only ones that contain $T/T^\dagger$ gates. Recall that the Grover algorithm needs n qubits to search $N = 2^n$ elements in a list. The oracle and the diffusion operator are applied once for each iteration of the Grover algorithm. Then if there is only one marked state, one iteration of Grover's

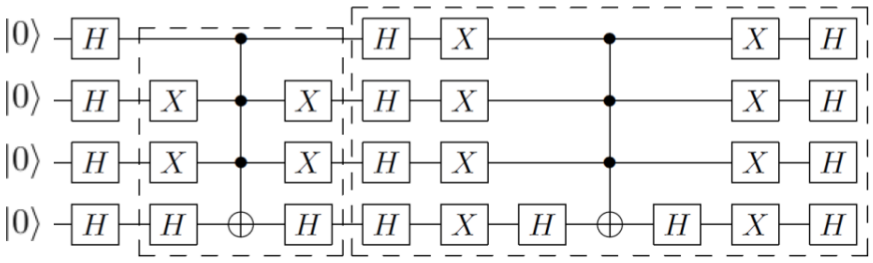


Figure 6.7: Grover oracle and diffusion operator for $w = 1001$ using gates from the set $\mathcal{G}$ and two multi-controlled $CNOT$ gates with three control qubits. Both operators are surrounded by a dotted box to clearly delimit the operations they contain. The operators are used iteratively $O(\sqrt{N})$ times and then each qubit is measured.

algorithm requires two multi-controlled $CNOT$ gates (with $n-1$ control qubits), one for the oracle and another for the diffusion operator. The next step is decomposing these gates in terms of $T/T^\dagger$ gates.

As it was shown in chapter 4, a multi-controlled $CNOT$ gate with $n-1$ control qubits can be decomposed into $2(n-2)$ Toffoli gates using $(n-2)$ extra ancilla qubits. An example of this decomposition [139] is shown again in figure 6.8 for any single qubit gate U. If $U = X$ this corresponds to the decomposition of the multi-controlled $CNOT$ gate.

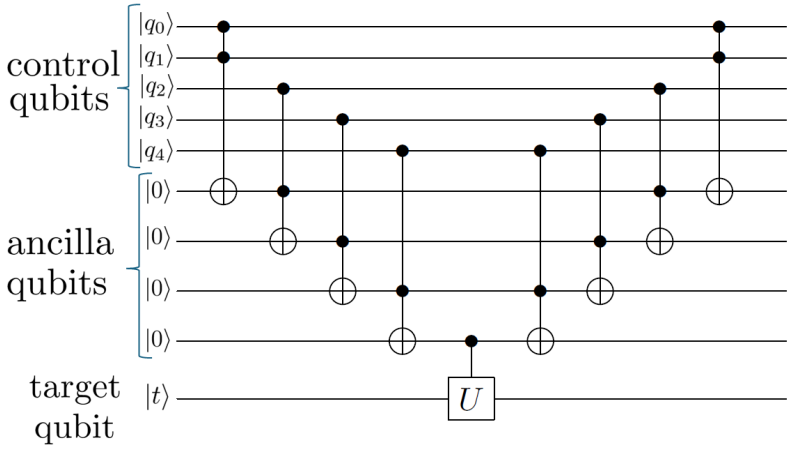


Figure 6.8: A $n = 6$ multi-controlled single qubit gate with 5 control qubits and 4 ancilla qubits, decomposed into 8 Toffoli gates.

If two multi-controlled $CNOT$ gates are used to run one query of the Grover algorithm for a list with just one marked item, then the number of Toffoli gates is given by:

$$2 \quad \text{MCNOT} = 2(2(n-2)) \quad \text{Toffolis} = 4(n-2) \quad \text{Toffolis} = 4(\log_2(N)-2) \quad \text{Toffolis} \tag{6.2}$$

where $n = \log_2(N)$ was used at the last step. Since the oracle and the diffusion operator are repeated $O(\sqrt{N})$ times, the number of Toffoli gates needed to run the whole algorithm is given by:

$$4[\log_2(N)-2)] \cdot \sqrt{N} \quad \text{Toffolis}. \tag{6.3}$$

Since a Toffoli gate can be decomposed in seven $T/T^\dagger$-gates (figure 6.2) the number of $T/T^\dagger$ gates, M, is finally obtained:

$$M = 7 \cdot 4[\log_2(N)-2)] \cdot \sqrt{N} = 28[\log_2(N)-2)] \cdot \sqrt{N} \quad \rightarrow O(\log_2(N)\sqrt{N}). \tag{6.4}$$

From eq. (6.4) we see that M grows slower than a linear function. Since the QHE scheme is only suitable for circuits with a polynomial number of $T/T^\dagger$ gates, the fact that $M = O(\log_2(N)\sqrt{N})$ means that the Grover algorithm can be implemented efficiently using the QHE scheme discussed.

If there are more desired elements in the N items list instead of just one, this result is still true. The reason for this is that for each new marked element, the diffusion operator remains the same and only the oracle changes. In this case, the oracle marks m elements in the list rather than one. Thus, instead of just using one multi-controlled $CNOT$ gate, the oracle would be implemented using m multi-controlled $CNOT$ gates. If $m+1$ multi-controlled $CNOT$ gates are used to run one query of the Grover algorithm, the number of Toffoli gates is given by:

$$m+1 \quad \text{MCNOT} = (m+1)(2(n-2)) \quad \text{Toffolis} = 2(m+1)(\log_2(N)-2) \quad \text{Toffolis}. \tag{6.5}$$

Following the same reasoning as in the previous case of just one marked element, the number of $T/T^\dagger$ gates of the whole algorithm is still $O(\log_2(N)\sqrt{N})$ and so the decryption process of the algorithm remains efficient. Therefore the Grover algorithm can be homomorphically implemented using n qubits and $n-2$ extra ancilla qubits in an efficient manner. For a $N = 2^n$ list, the client should then prepare $2n-2$ qubits in total, encrypt them, send them to the server and then perform the corresponding M measurements to correctly decrypt the results.

We want to mention that more optimizations can be used, such as decomposing the multi-controlled $CNOT$ gate using the relative phase Toffoli gate from [166] instead of the decomposition based on Toffoli gates that was used in figure 6.8. This leads to a reduction in the number of $T/T^\dagger$ gates. In particular, instead of the $14(n-2)$ $T/T^\dagger$ gates that were needed using our decomposition for the multi-controlled $CNOT$ gate with n qubits, only $8n-17$ $T/T^\dagger$ gates are used. Also, the number of ancillas decreases from $n-2$ in our case to $\lceil \frac{n-3}{2} \rceil$. Using this relative phase Toffoli gate decomposition, we get

$$M = 2[8\log_2(N) - 17] \cdot \sqrt{N} = [16\log_2(N) - 34] \cdot \sqrt{N}, \tag{6.6}$$

and so M is still bounded by $O(\log_2(N)\sqrt{N})$. Another example of further optimizations that can be used is replacing the diffusion operator with all single qubit gates, as proposed in [167]. Inspired by quantum partial search algorithms such as [168] and [169], a version of Grover's algorithm with improved depth was proposed [170]. Also, quantum search algorithms with reduced depth compared to the standard Grover algorithm have been implemented on the IBM quantum computers [171]. These algorithms with reduced depth constitute examples of other type of further optimizations that can be applied. However, we want to clarify that our publication [163] was not concerned about finding the most optimal homomorphic implementation of Grover's algorithm. Instead, its purpose was to study whether the homomorphic encryption of Grover's algorithm can be implemented in an efficiently enough manner. Using our gate decompositions, we have found an upper bound that grows slower than any linear function. Therefore, we have shown that our homomorphic implementation of the algorithm can be performed in an efficiently enough manner. More efficient implementations can be applied , such as the relative phase Toffoli gates, but studying them in detail in order to find the most optimal implementation is left as future work.

We want to mention some results regarding the special case in which there is only one desired element in the database. Grover [172] proposed an algorithm that finds this unique element using only $O(\sqrt{N}\log_2\log_2 N)$ elementary gates without increasing the number of queries needed in a significant manner. This algorithm is no longer made of $O(\sqrt{N})$ identical iterations and it is more complicated than the standard Grover algorithm. Then in 2017 Arunachalam and de Wolf [164] proposed a more efficient Grover search algorithm regarding its gate complexity. For a sufficiently large database of size N that contains just one marked element and for any constant r, this algorithm finds the only desired element using $O(\sqrt{N})$ queries and $O(\sqrt{N}\log_2^{(r)} N)$ elementary gates. The iterated binary logarithm is defined as $\log_2^{(s+1)} = \log_2 \circ \log_2^{(s)}$ where $\log_2^{(0)}$ is the identity function. The elementary gates used are the Toffoli gate and any unitary single qubit gate like the H and X gates. If we have a very large N items list that is also a power of 2, r can be chosen to be $r = \log_2^\star N$ so the algorithm finds the searched element using only $O(\sqrt{N}\log_2(\log_2^\star N))$ elementary gates in the optimal $\frac{\pi}{4}\sqrt{N}$ queries. The function $\log_2^\star N$ represents the number of times the binary logarithm must be iteratively applied to N to obtain a number that is at most 1: $\log_2^\star N = \min\{r \geq 0 : \log_2^{(r)} N \leq 1\}$. Later in 2021, Briański et al. [165] proposed an algorithm that is even more efficient regarding the gate complexity. Fix any $\varepsilon \in (0,1)$ and any $N \in \mathbb{N}$ of the form $N = 2^n$. Suppose a quantum oracle O is given that operates on n qubits marking exactly one element. Then there exists a quantum circuit, denoted by $\mathcal{A}$, that uses the oracle O at most $(1+\varepsilon)\frac{\pi}{4}\sqrt{N}$ times and uses at most $O(\log_2(1/\varepsilon)\sqrt{N})$ nonoracle basic gates, which finds the element marked by O with certainty. Since these algorithms have a reduced number of $T/T^\dagger$ gates compared to the standard Grover algorithm, they can also be homomorphically evaluated using the QHE scheme presented in chapter 2. Furthermore, their homomorphic evaluation is even more efficient than the standard Grover algorithm due to their reduced $T/T^\dagger$-gate complexity.

Arunachalam and de Wolf [164] mentioned that most applications of Grover's algorithm study databases with an unknown number of desired elements and only focus on the number of queries. They ended their article asking whether there are any applications where the reduction in the number of quantum gates for the special case of just one marked element is both applicable and significant. We want to give a positive answer for this question. Quantum homomorphic encryption is a perfect application for this algorithm, since the advantage it offers regarding the gate complexity is significant in the context of improving the efficiency of the decryption process.

6.4 Homomorphic quantum key attack

We want to comment on the complexity of the oracle of Grover's algorithm. The oracle that has been presented is the simplest one that can be chosen. For more practical quantum search problems, such as quantum key search for block ciphers like the Advanced Encryption Standard (AES), more complex oracles are needed. These oracles require more resources, so it is interesting to study if the homomorphic evaluation of these oracles is still efficient.

In [173], quantum circuits that perform quantum key search using the Grover algorithm for the Advanced Encryption Standard (AES) are proposed. Given a block cipher C with block length n and key length k, for a key $K \in \{0,1\}^k$, we can denote by $C_K(m) \in \{0,1\}^n$ the encryption of message $m \in \{0,1\}^n$ under the key K. Given r plaintext-ciphertext pairs (m_i, c_i) where $c_i = C_K(m_i)$, Grover's algorithm can be applied in order to find the secret key K. Since AES operates on plaintexts-ciphertexts of length 128, we have that $c_i \in \{0,1\}^{128}$. The Boolean function f for the Grover oracle takes a key $K \in \{0,1\}^k$ (where $k \in \{128, 192, 256\}$ for AES) as input and can be defined as:

$$f(K) = \begin{cases} 1 & \text{if } c_i = C_K(m_i) \text{ for all } 1 \leq i \leq r \\ 0 & \text{otherwise.} \end{cases} \tag{6.7}$$

There are other keys than K that can possibly encrypt the known plaintexts to the same ciphertexts. These keys are called spurious keys.

The circuits proposed in [173] are studied for each standardized key size of AES: 128, 192 and 256 bits. Each variant is denoted by AES-128, AES-192 and AES-256. Specifically, the number of T gates and Clifford gates for the oracle and the whole Grover key search are calculated for these circuits. As always, we are interested in the number of T gates of these circuits, since it determines the complexity of the decryption process of the QHE scheme. It is assumed that $r = 3$ for AES-128, $r = 4$ for AES-192, and $r = 5$ for AES-256 suitable plaintexts-ciphertexts pairs are needed to characterize the secret key uniquely. Using a circuit for the block cipher, the Grover oracle encrypts r plaintext blocks under the same candidate key. Next, a comparison operation with the classical ciphertexts flips the result qubit conditionally and the r encryptions are

un-computed. This comparison function is implemented using a multi-controlled NOT gate with $128r$ control qubits (for $r = 3, 4, 5$) and a single target qubit. For a key size of $k = 128$ bits, the oracle is composed of a 384-fold controlled NOT gate and six AES boxes that can be decomposed into 12204 and $6 \cdot 1060864 = 6365184$ T gates respectively. Its circuit is shown in figure 6.9. Thus, the oracle for AES-128 requires 6377388 T gates.

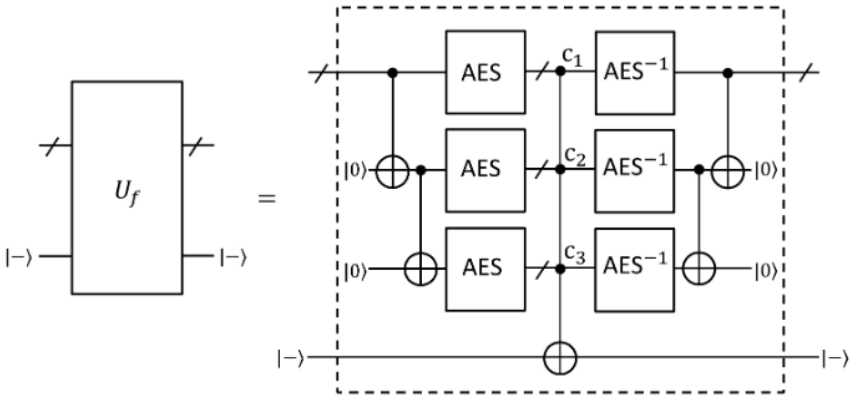


Figure 6.9: Oracle implementation for the key size $k = 128$. It requires $r = 3$ AES invocations that have to be uncomputed later, which means that six AES boxes are required in total to find a unique key. Once the AES boxes have been computed, the result is compared with the given ciphertexts $c_1, \ldots, c_r$ using a multiply controlled NOT gate in which the controls are determined by the values of the bits of each c_i. This is denoted by the superscript c_i on top of the controls in the figure. To determine the number of T gates, we have to add the contribution from the 384-fold controlled NOT gate and each AES box. Image obtained from [173].

As a crude time estimate, assuming that one Bell measurement and the subsequent key-updating needed to obtain the key of the next Bell measurement can be performed in 10^{-3} seconds at best and 1 second at worst, between 1.771 and 1771 hours would be needed to decrypt the whole oracle. Taking into account this estimation and the number of T gates mentioned, we believe implementing this oracle homomorphically is quite complex but still feasible. To implement the whole Grover algorithm, a 128-fold controlled NOT implements the diffusion operator and $\lfloor \frac{\pi}{4} 2^{k/2} \rfloor$ iterations are needed. The estimated number of T gates required to implement the whole Grover algorithm on AES-128 is:

$$(6377388 + 1000) \cdot \lfloor \frac{\pi}{4} 2^{64} \rfloor = 9.24 \cdot 10^{25} = 1.19 \cdot 2^{86}. \tag{6.8}$$

Considering this large number of T gates, we believe that the homomorphic implementation of the whole Grover algorithm on AES-128 can not be implemented efficiently since the decryption procedure would take too much time to be completed.

For AES-192, the oracle is similar to the one presented in figure 6.9, only that it now requires eight AES boxes and a a 512-fold controlled NOT gate, so we have that their number of T gates is $8 \cdot 1204224 = 9633792$ and 16300 respectively. The T-count of the whole oracle is then 9650092. In case of AES-256, the oracle requires ten AES boxes and a 640-fold controlled NOT gate, so their number of T gates is $10 \cdot 1505280 = 15052800$ and 20396 respectively. The T-count of the whole oracle is then 15073196. For AES-192, a 192-fold controlled NOT implements the diffusion operator, so the estimated number of T gates required to implement the whole Grover algorithm on AES-192 is:

$$(9650092 + 1512) \cdot \lfloor \frac{\pi}{4} 2^{96} \rfloor = 1.81 \cdot 2^{118}. \tag{6.9}$$

In case of AES-256, a 256-fold controlled NOT implements the diffusion operator, so the estimated number of T gates required to implement the whole Grover algorithm on AES-256 is:

$$(15073196 + 2024) \cdot \lfloor \frac{\pi}{4} 2^{128} \rfloor = 1.41 \cdot 2^{151}. \tag{6.10}$$

From these number of T gates contained in AES-192 and AES-256, we can see that more gates are needed compared to AES-128 and so the homomorphic implementation of AES-192 and AES-256 is even more inefficient.

We have studied more articles that cover practical Grover search problems. In [174], the cost analysis of implementing Grover's key search algorithm on the family of KATAN block ciphers is studied. KATAN is a family of hardware-oriented, lightweight block ciphers specially designed for sensor networks, radio-frequency identification tags, and Internet of Things. The different variants studied are KATAN32, KATAN48 and KATAN64, depending on the block length. For the oracle, 64630, 85680 and 128464 T gates are needed for KATAN32, KATAN48 and KATAN64 respectively using Toffoli gates in the decomposition. The authors also studied the resources needed using a gate called the AND gate. It is shown in figure 6.10. If the oracle is constructed using the AND gate presented in the paper, 37248, 49104 and 73600 T gates are needed for KATAN32, KATAN48 and KATAN64 respectively. These T counts are lower than those obtained for AES and so these oracles can be implemented more efficiently. Using the same crude time estimation as in AES, the worst KATAN64 oracle that contains 128464 T gates would be implemented in 35 hours at worst and 0.035 hours at best. To implement the whole algorithm $\lfloor \frac{\pi}{4} 2^{k/2} \rfloor$ iterations are needed with no parallelization. Thus, the T gate count is $2^{55.63}$, $2^{56.04}$ and $2^{56.62}$ for KATAN32, KATAN48 and KATAN64 if the Toffoli gate is used. If the AND gate is used for the whole algorithm, then the T gate count is $2^{54.84}$, $2^{55.23}$ and $2^{55.82}$ for KATAN32, KATAN48 and KATAN64. These numbers are lower compared to AES. However they are still too large for efficient homomorphic implementation, since taking the lowest value of $2^{54.84}$ and using the previous time estimation, at best $8.95 \cdot 10^9$ hours would be needed to complete decryption process of the whole algorithm.

In [175], Grover's search algorithm is applied on all the variants of a lightweight cipher known as SIMON. The different variants of SIMON are denoted by SIMON $2n/mn$,

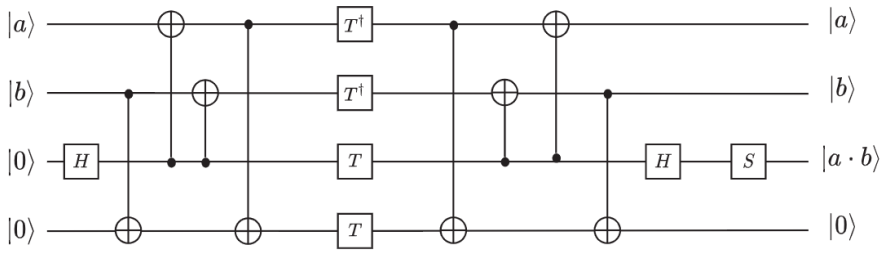


Figure 6.10: AND gate used to study the cost analysis of implementing Grover's key search algorithm on the family of KATAN block ciphers. Image obtained from [174].

where $2n$ denotes the block size of the variant and mn is the size of the secret key. Here, n can take values from 16, 24, 32, 48 or 64 and m from 2, 3 or 4. The quantum resources needed for this attack are also estimated. The variants studied were: SIMON 32/64, 48/72, 48/96, 64/96, 64/128, 96/96, 96/144, 128/128, 128/192 and 128/256. For the oracle, the lowest number of T gates found is 24492 for SIMON 32/64 and the highest is 205740 for SIMON 128/256. Once again these oracles present a lower T gate count compared to those found for AES. For the whole algorithm, the lowest number of T gates is $1.27 \cdot 2^{46}$ for SIMON 32/64 and the highest is $1.11 \cdot 2^{145.2}$ for SIMON 128/256.

In [176], Grover's algorithm is applied to an optimized implementation of Simplified AES and the quantum resources for the attack are also obtained. Simplified AES uses a 16-bit block plaintext and a 16-bit key. In order to construct the oracle, 192 Toffoli gates and 32 multi control $CNOT$ gates with three control qubits are needed. Using the decompositions of these gates that we have presented (figures 6.2 and 6.8) for simplicity, the upper bound for the number of T gates is $192 \cdot 7 + 32 \cdot 4 \cdot 7 = 2240$. For the whole algorithm, the number of T gates is $2240 \cdot \lfloor \frac{\pi}{4} 2^{16/2} \rfloor = 450240$. This number of T gates is significantly lower compared to the previous cases shown. Thus, the efficient homomorphic encryption of Grover's algorithm applied to simplified AES is feasible, since using the same crude time estimation as in AES we find that the decryption process would take between 125 hours at worst and 0.125 hours at best. However, simplified AES is a teaching tool designed to help students understand the basic structures of AES and is not a block cipher that is actually used in practical cryptography.

Regarding AES, the results presented in [173] have been improved in different articles. In [178], an explicit quantum design of AES-128 is given. The number of T gates for one invocation of AES-128 is 1053696, which is better than the 1060864 T gates needed in [173]. Besides this improvement, the number of qubits used is also reduced. In [177], rather than focusing on minimizing the number of qubits, the authors study AES-128, AES-192 and AES-256 focusing on minimizing the oracle's depth. They also managed to reduce the number of known plaintext-ciphertext pairs required to obtain a unique key compared to [173]. This way, using $r = 2$ for both AES-128 and AES-192 and $r = 3$

Reference	Variant	Number of T gates: oracle	Number of T gates: Grover key search
Ref [173]	AES-128 with $r = 3$	6377388	$1.19 \cdot 2^{86}$
	AES-192 with $r = 4$	9650092	$1.81 \cdot 2^{118}$
	AES-256 with $r = 5$	15073196	$1.41 \cdot 2^{151}$
Ref [174]	KATAN32-Toffoli gate	64630	$2^{55.63}$
	KATAN48-Toffoli gate	85680	$2^{56.04}$
	KATAN64-Toffoli gate	128464	$2^{56.62}$
	KATAN32-AND gate	37248	$2^{54.84}$
	KATAN48-AND gate	49104	$2^{55.23}$
	KATAN64-AND gate	73600	$2^{55.82}$
Ref [175]	SIMON 32/64	24492	$1.27 \cdot 2^{46}$
	SIMON 128/256	205740	$1.11 \cdot 2^{145.2}$
Ref [176]	Simplified AES	2240	450240
Ref [177]	AES-128 with $r = 1$	54908	$1.32 \cdot 2^{79}$
	AES-128 with $r = 2$	109820	$1.32 \cdot 2^{20}$
	AES-192 with $r = 2$	122876	$1.47 \cdot 2^{112}$
	AES-256 with $r = 2$	151164	$1.81 \cdot 2^{144}$
	AES-256 with $r = 3$	226748	$1.36 \cdot 2^{145}$

Table 6.2: Number of T gates required to implement the oracle and the whole Grover key search for different schemes. Using the mentioned crude time estimation, each number also represents the time it would take to complete the decryption process of the oracle and the whole algorithm in the worst case expressed in seconds. The time needed in the best case is obtained by simply multiplying each worst case value by 10^{-3}. r refers to the number of known plaintext-ciphertext pairs that are required for a successful key-recovery attack.

for AES-256 guarantees a unique key with overwhelming probability. In particular, the probabilities of obtaining no spurious keys are $1 - \varepsilon$, where $\varepsilon < 2^{-64}$, $\varepsilon < 2^{-128}$ and $\varepsilon < 2^{-128}$ respectively. Besides this, they also mention that if a success probability lower than 1 is acceptable, then it suffices to use $r = \lceil k/n \rceil$ plaintext-ciphertext pairs. As an example, if $r = 1$ is used for AES-128, the probability of not having spurious keys is $1/e \approxeq 0.368$. This could be a high enough probability for a successful attack in certain situations, like when there exists a tight limit on the width of the attack circuit. Regarding the oracle, for AES-128 they report 54908 T gates for $r = 1$ and 109820 T gates for $r = 2$, for AES-192 they report 122876 T gates for $r = 2$ and for AES-256 they report 151164 T gates for $r = 2$ and 226748 T gates for $r = 3$. These oracles are a considerable improvement compared to the previous AES oracles mentioned and their homomorphic implementation seems feasible. For the whole algorithm, this work reports $1.32 \cdot 2^{79}$ and $1.32 \cdot 2^{80}$ T gates for AES-128 with $r = 1$ and $r = 2$ respectively. Even though the number of T gates is significantly reduced, we believe these numbers

are still too large for efficient homomorphic encryption, since using our usual estimation the decryption process would take too much time to be completed.

Table 6.2 shows the number of T gates needed to implement the corresponding oracle and the whole Grover key search for the different schemes that have been presented in this section. Using the usual crude time estimation, each number also represents the time it would take to complete the decryption process of the oracle and the whole algorithm in the worst case expressed in seconds. To obtain the corresponding time needed in the best case, we simply have to multiply each worst case value by 10^{-3}.

In conclusion we have studied the encryption resources of different practical Grover's algorithms. We have found that the number of T gates of quantum key attacks for symmetric encryption schemes is too large to be implemented homomorphically in an efficient manner except in toy models like simplified AES. Therefore, applying homomorphic encryption to these algorithms seems too inefficient. Still, other practical problems where Grover's algorithm can be applied, such as the Boolean satisfiability problem, may have simpler oracles that can lead to an efficient homomorphic implementation of the whole algorithm. The research of more practical versions of Grover's algorithm that have an efficient homomorphic implementation is left as future work. Nevertheless, we believe that the work presented here can serve as a good basis in this endeavour.

6.5 Conclusions

- We have combined Liang's [127] quasi-compact QHE scheme and Grover's algorithm to perform homomorphically encrypted quantum search. This QHE scheme has perfect security, $\mathcal{F}$-homomorphism, no interaction between client and server and quasi-compactness. The scheme does not contradict the no-go result obtained in [121] because it is not compact. The decryption procedure is independent of the size of the evaluated circuit and depends only on the number of $T/T^\dagger$ gates contained in the circuit.

- Using this QHE scheme, we have simulated a quantum circuit that implements the Grover algorithm homomorphically for a particular case which contains three qubits using Qiskit [163]. We calculated the values of the keys at each circuit's step and showed how to apply classically controlled-S gates in the simulation to account for every possible result. After obtaining the final key for each qubit involved, the results obtained from this simulation can always be correctly decrypted. This can be seen in table 6.1.

- Regarding the decryption procedure of the QHE scheme, it is only efficient for circuits with a polynomial number of $T/T^\dagger$-gates. As we have seen, the number of $T/T^\dagger$ gates of any Grover circuit is $O(\log_2(N)\sqrt{N})$ which means that it grows slower than any linear function. Therefore, Grover's algorithm is a perfect

illustration of a quantum algorithm that can be implemented homomorphically with perfect security and non interaction in an efficient manner. In the particular case of a database with just one desired element, more efficient algorithms have been proposed by Arunachalam [164] and Briański [165]. Their gate complexity is $O(\sqrt{N}\log_2(\log_2^\star N))$ and $O(\log_2(1/\varepsilon)\sqrt{N})$ respectively, so the decryption procedure of their homomorphic evaluation is more efficient than the standard Grover algorithm.

7 The Szegedy Quantum Algorithm and Quantum Walks

7.1 Background

Quantum walks are one of the most promising family of algorithms in the current landscape. These algorithms are generated from the quantization of classical random walks. The discrete time version of these algorithms was the first one to be proposed [179]. A continuous time version was proposed later [180]. These walks have produced a large variety of algorithms for problems such us element distinctness [181], quantum search [182] and triangle finding [183]. Another reason why they are remarkable is that they can be used to simulate different physical systems [184].

One of the most important quantum walks in discrete time was introduced by Szegedy [185] as a generalization of the Grover algorithm [156]. Unlike other approaches, such as the coined quantum walk [186] that act on undirected graphs and present difficulties quantizing arbitrary Markov chains, Szegedy quantum walk can quantize a general Markov chain. For this reason, this walk can be applied to any arbitrary weighted graph. The original formulation proposed by Szegedy was a coinless quantum walk on bipartite graphs that quantized a general Markov chain on weighted graphs by means of a duplication process. Later, an alternative formulation of the walk was developed [187] that avoided the requirement of duplicating the graph. This formulation allows to establish an equivalence between the coined quantum walk model and Szegedy quantum walk [188, 189].

Szegedy quantum walk has shown its usefulness for many different types of problems, such as classification [190–192], optimization [193–197], quantum search [184, 198–200] and machine learning [201]. Also a graph-phased Szegedy quantum walk has been developed recently, which includes complex-phase extensions [202]. We want to mention the quantum PageRank algorithm [190] as another important application that relies on the Szegedy quantum walk. The classical PageRank algorithm represented a revolution in the field of search engines for surfing the Internet [203]. This algorithm can rank pages objectively using the structure of the network they form. The quantum version has shown that it can outperform the classical one.

Recently, another type of walk algorithm that combines classical and quantum traits

was developed [204]. This algorithm is known as the semiclassical walk. They have been analysed in the context of searching and ranking nodes in graphs, demonstrating advantages with respect both classical and quantum algorithms [204, 205]. In practice, these walk algorithms are based on repeated measurements of the quantum walker's position at fixed time intervals. This trait could be important in algorithms that need to know the position of the walker not only at the end of the quantum walk. After each measurement, part of the system has to be reset. This means that the information of the qubits must be erased and the system must be reinitialized using either quantum or classically controlled gates that can read the previous measurement result.

7.2 Review of quantum walks algorithms

7.2.1 Classical walks

We first review the classical walks in an brief manner. The classical random walk happens in the nodes of a graph. Taking a stochastic perspective, the walker is located in one particular node of the graph at each time step. From this node, it can jump to any other node, including the current one, with a certain probability [204]. A classical walk performed in a graph with three nodes is shown in figure 7.1. At time $t = 0$ the walker

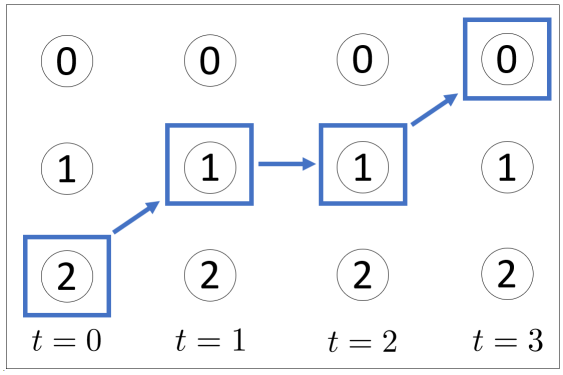


Figure 7.1: Example of a trajectory followed by a particle in a classical walk performed in a graph with 3 nodes. The walker is in a certain node at each time step and it jumps to another node with a certain probability. Image obtained from [204].

is located in node 2. At $t = 1$, it jumps to node 1 stochastically. At $t = 2$ it remains in the same node and finally at $t = 3$ it jumps to node 0. The trajectory that the walker follows is different each time the walk is performed, since this process is stochastic. A probability distribution for the walker being at each node at each time step can be obtained if we average over the different trajectories that the walker may follow.

The probability distribution of the walker being at each node can be simulated deterministically if the probabilities that the walker jumps from one node to any other are known. We define the classical transition matrix of a graph G, whose elements G_{ji} represent the probabilities that the walker jumps from node i to node j. The probabilities of each column add to one, so we have:

$$\sum_{j=0}^{N-1} G_{ji} = 1, \tag{7.1}$$

where N is the number of nodes contained in the graph. The matrix G is then a $N \times N$ matrix.

We denote by $p(t)$ a column vector whose elements are the probabilities that the walker is at each node at a certain time t. Then, the classical walk can be simulated deterministically, given an initial probability distribution called $p(0)$, using the transition matrix as

$$p(t) = G^t p(0). \tag{7.2}$$

Both points of view have applications depending of what the desired result is: a specific position of the walker or a probability distribution. The stochastic simulation is used in optimization algorithms, such as simulated annealing with Metropolis-Hastings, where obtaining a single node as an optimal solution is what is desired [206–208]. Calculating the entire transition matrix can be unfeasible if the search space is too large. Then, by calculating only the probabilities required at each time step, the computational cost of the algorithm is reduced.

7.2.2 Szegedy quantum walk

The classical Markov chain can be quantized through Szegedy's quantum walk [185,187]. In this quantum walk the Hilbert space is the span of all the vectors representing the $N \times N$ directed edges of the graph, so we have

$$\mathcal{H} = \text{span}\{|i\rangle_1 |j\rangle_2 ,\ i,j = 0,1,...,N-1\} = \mathbb{C}^N \otimes \mathbb{C}^N, \tag{7.3}$$

where the states with indexes 1 and 2 refer to the nodes on two copies of the original graph. Then, the states are defined over two quantum registers, which can be reinterpreted as position and coin registers respectively [202, 204]. By convention the nodes of the network, and thus the matrix indexes, are counted from 0 to $N-1$. We define the vectors

$$|\psi_i\rangle := |i\rangle_1 \otimes \sum_{k=0}^{N-1} \sqrt{G_{ki}}\, |k\rangle_2 , \tag{7.4}$$

which are a superposition of the vectors that represent the edges outgoing from the i^{th} vertex, whose coefficients are given by the square root of the i^{th} column of the transition matrix G.

Next, we define a projector operator onto the subspace generated by these vectors:

$$\Pi := \sum_{i=0}^{N-1} |\psi_i\rangle \langle\psi_i| . \tag{7.5}$$

The quantum walk operator U_w is defined as

$$U_w := S_w R, \tag{7.6}$$

where R is a reflection about the subspace generated by the states $|\psi_i\rangle$,

$$R := 2\Pi - \mathbb{1}, \tag{7.7}$$

and S_w represents the swap operator between the two quantum registers,

$$S_w := \sum_{i,j=0}^{N-1} |i,j\rangle \langle j,i| . \tag{7.8}$$

The initial state of the system $|\psi(0)\rangle$ is typically a linear combination of the $|\psi_i\rangle$ states that represent the initial probability distribution of the walker. Once the unitary evolution U_w has been applied a number t of time steps, the position of the walker can be usually obtained by measuring the first register. Then, the probability distribution, denoted by $p_q(t)$, is provided by the projection onto the computational basis of the first register,

$$[p_q(t)]_i = \left|\left|\, {}_1\langle i| \, U_w^t \, |\psi(0)\rangle \right|\right|^2 . \tag{7.9}$$

There are some algorithms where the second register is measured instead of the first [190–192, 205]. Nevertheless without loss of generality, in our publication [209] we only considered measurements in the first register.

The formulation of Szegedy quantum walk that we have just described corresponds to the alternative formulation that we mentioned earlier [187]. This way, it can be understood as a coined quantum walk in which the first register encodes the position of the walker in the graph, and the second register encodes the coin state. Szegedy's original formulation was based on a coinless quantum walk that performed two reflections, each around one of the two previous subspaces [185]. Using this original formulation, the unitary operator for one step of Szegedy's quantum walk can be expressed as [184]:

$$W = R_B R_A, \tag{7.10}$$

where R_A corresponds to a reflection over the subspace spanned by the $|\psi_i\rangle$ states and R_B corresponds to another reflection over the subspace generated by the swapped version of these states. If we identify $R_A = R$ and $R_B = S_w R S_w$, we can see that the original unitary operator W corresponds to the application of the coined version of the operator twice, which means that $W = U_w^2$. We presented this original formulation for completeness purposes, and we will use the alternative formulation [187] from this point onwards.

7.2.3 Semiclassical walk formulation

Semiclassical walks are a new type of algorithms that combine both classical and quantum features [204]. They have showcased superior performance compared to both classical and quantum algorithms in ranking and searching problems [205]. Considering a functional point of view in a quantum computer, these walks consist of repeated measurements of the walker's position at regular time intervals. Two parameters are used to describe a semiclassical walk:

- Quantum time t_q, which is the number of times the walk unitary evolution operator U_w is applied between measurements.
- Classical time t_c, which represents the number of times that the whole scheme of quantum evolution and measurement is repeated.

In case of the semiclassical Szegedy walk, we can distinguish two classes of algorithms, class I and class II, depending on which register is being measured (1 or 2) to obtain the information about the position of the walker. It is more common to have walks of class I, but there are applications where the measured register is the second one, like the quantum PageRank algorithm [199, 203]. Nevertheless, we will deal just with the semiclassical walks of class I from this point onwards.

Each classical step of the semiclassical walk consists of a quantum evolution which is determined by the quantum time t_q, a measurement of the position and then a reset of the system. The information contained in the second register is erased to reset the system and the result obtained from measuring the position of the walker is used to prepare its corresponding proxy state. These proxy states represent the walker's classical position in the Hilbert space where the quantum walk happens. These proxy states can be defined arbitrarily but in the case of Szegedy's quantum walk, there is a natural choice of states, which is the set of states determined by eq. (7.4). An example of this type of semiclassical walks of class I is shown in figure 7.2. We denote by x_{t_c} the classical position of the walker at a certain classical time t_c. The walker starts at node x_0. Then we prepare the state $|\psi_{x_0}\rangle$ and perform the quantum evolution parametrized by the quantum time t_q, applying the operator U_w as many times as t_q indicates. Next, the first register is measured and the resulting state of the system is a particular node x_1 in the first register. The state of the second register is not relevant, since the system is reset using the information about the node obtained from the measurement. Next, we prepare the new state for the node that was measured: $|\psi_{x_1}\rangle$. This procedure can be repeated as many classical steps as desired.

Up until this point, we have described the semiclassical walk in an abstract manner. We now proceed to show the quantum circuits needed to perform this process. As we mentioned, at each classical step t_c, the classical position of the walker x_{t_c} is represented in the quantum system by its corresponding state $\left|\psi_{x_{t_c}}\right\rangle$ in eq. (7.4). In order to prepare this state, we introduce the update operator V, which prepares the state $|\psi_i\rangle$ from the

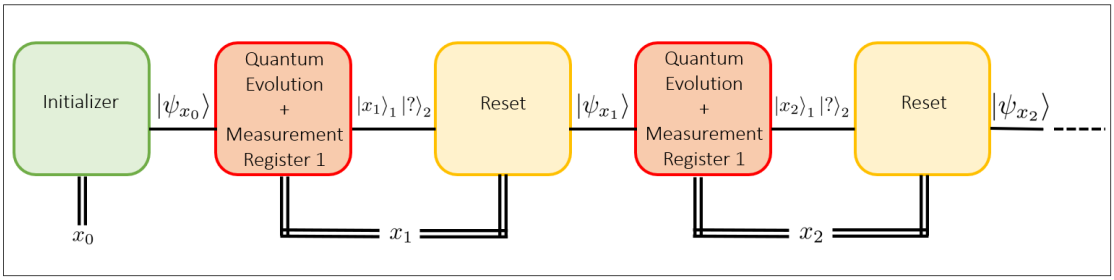


Figure 7.2: Representation of the semiclassical Szegedy walk of class I. The position of the walker at each classical time step is given by x_{t_c}. The classical information is used to prepare the corresponding state $|\psi_{x_{t_c}}\rangle$, then the quantum evolution is performed, and finally a new classical position is measured from the first register. Image obtained from [204].

computational basis of the first register, provided the second one is in the state $|0\rangle_2$ [210]. We then have:

$$V |i\rangle_1 |0\rangle_2 = |\psi_i\rangle . \tag{7.11}$$

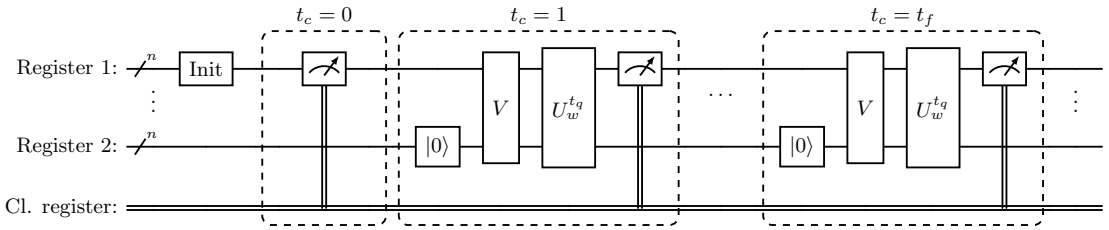


Figure 7.3: Quantum circuit for the semiclassical Szegedy walk. The walk is performed t_f classical time steps, and at each classical step the quantum evolution operator U_w is applied t_q times. After each classical step measurement performed in the first register and before the next quantum walk evolution, the second register is reset to the state $|0\rangle_2$. Then the update operator V is applied to create the new initial state $|\psi_x\rangle$ depending on the measurement result x.

The implementation of a semiclassical walk using quantum circuits is shown in Figure 7.3. First, we generate an initial state in the first register. This state represents the initial classical distribution of the walker at time step $t_c = 0$. It can be implemented as a quantum superposition of the computational basis where all coefficients are positive or as a mixed state. After the measurement at $t_c = 0$, the first register results in the state $|x_0\rangle_1$. From this point onwards, each classical step $t_c > 0$ is composed of four operations. The second register is reset to the state $|0\rangle_2$, so that its previous information is erased and the system transforms into the state $|x_{t_c-1}\rangle_1 |0\rangle_2$. Next, the update operator creates the initial state of the step $|\psi_{x_{t_c-1}}\rangle$. The quantum unitary evolution operator U_w of the Szegedy quantum walk is applied a number of times determined by t_q, and finally the first register is measured once again to obtain the new classical position x_{t_c}. This procedure

is repeated until the last classical step t_f is completed.

From a rigorous point of view, semiclassical walks can be understood as classical walks in which the transition matrices encode the quantum evolution. These are known as semiclassical transition matrices and can be defined as:

$$G_{ji}^{(t_q)} := \left|\left| {}_1\langle j| U_w^{t_q} |\psi_i\rangle \right|\right|^2 . \tag{7.12}$$

Notice that we actually have an entire family of semiclassical walks, parametrized by the quantum time t_q, and that the time steps of the walker are actually determined by the classical time t_c. The semiclassical matrix could be used to simulate the semiclassical walk deterministically in a classical computer, provided there is an efficient classical simulator of the quantum walk to obtain all the matrix elements [211].

8 Homomorphic Encryption of Quantum Walks Algorithms

8.1 Motivation

Quantum walks algorithms constitute one of the most successful family of quantum algorithms, due to their many different variants and wide range of applications. Following our quest to study the homomorphic encryption of different quantum algorithms and taking into account the success found analysing the Grover algorithm, it is natural to turn our attention to quantum walks next. After the success found simulating the homomorphic implementation of Grover's algorithm, we are also interested in the simulation of QHE schemes, and we want to improve them as much as possible.

Regarding the simulation of QHE schemes, an open-source Python implementation of the EPR scheme [123] has recently been constructed [212]. There was also a previous work that simulated circuits consisting only of Clifford gates using Qiskit and the EPR schemes [213]. Liang's schemes have also been simulated in different works. A simulation of Grover's algorithm was performed using a circuit that only contained two qubits and Clifford gates [128]. Besides this, they also performed an experimental implementation of the decryption scheme for a single T gate using an IBM quantum computer. In our second publication [163], we simulated a more complex Grover search circuit using Liang's schemes in Qiskit. This simulation contained three qubits and seven $T/T^\dagger$ gates. We managed to correct the errors generated by the $T/T^\dagger$ gates using classically controlled S gates. However, the value of each key had to be calculated beforehand to determine the correct control value of the classically controlled S gates. This requires the calculation of a number of key values that grows exponentially with the number of $T/T^\dagger$ gates, and an exponential amount of classically controlled S gates.

In our third publication [209], we reformulated Liang's schemes using classical-quantum circuits for the decryption procedure. This reformulation of the scheme allowed us to construct a classical simulator using Qiskit. Our simulator constitutes an improvement compared to the previous simulations of Liang's schemes, since it does not require any previous calculation of the keys. Instead, the keys are calculated as the simulation runs. This results in a simulation that contains a linear number of classically controlled S gates instead of an exponential one. Thus, the evaluated circuits can contain any arbitrary number of $T/T^\dagger$ gates. Based on our simulation algorithm, we also developed a

Python library called Classical-Quantum Circuits for Quantum Homomorphic Encryption (CQC-QHE) to construct and classically simulate the classical-quantum circuits that are required for the quantum homomorphic encryption simulation.

We applied this QHE scheme to quantum walks, specifically the Szegedy quantum walk and its semiclassical counterpart. Before our work, a previous study of quantum walks with QHE was done [117]. However, it was focused on the context of Boson sampling using a multi walker, and is radically different from the usual quantum walks studied in the literature [184]. Furthermore, the QHE scheme applied was very limited since it was not universal, contrary to our scheme.

In order to apply the QHE scheme to semiclassical walks, the rules for the homomorphic evaluation of the reset and intermediate measurement operations must also be considered. We also provided the rules for these operations in our third publication [209]. We then examined the $T/T^\dagger$-gate complexity of the circuits required to implement Szegedy quantum walk on different graphs structures: cyclic graphs, complete graphs and bipartite graphs. We proved that all the circuits considered can be implemented homomorphically in an efficient manner. Besides this, we simulated a homomorphic Szegedy quantum walk on the bipartite graph for six qubits using Qiskit. The correct results are always obtained after the decryption procedure, independently of the initial key used in encryption. In order to check that the results are indeed the correct ones, we used the python library SQUWALS [211] to simulate the walk algorithm deterministically. We simulated a homomorphic semiclassical walk on a cycle graph that contains six qubits too. Once again, we used SQUWALS to simulate the walk deterministically in order to verify that the decrypted results of the QHE simulation are correct. The positive results obtained from both simulations show the correct functioning of the QHE scheme. Finally as we mentioned, we introduced the library CQC-QHE to construct and classically simulate the classical-quantum circuits needed for quantum homomorphic encryption simulation scenarios. This library can be accessed by anyone, so we expect our work will be of interest to those looking to apply homomorphic encryption to specific quantum circuits.

8.2 Quantum homomorphic encryption

In this section we describe a QHE scheme based on gate teleportation similar to Liang's scheme. However, we do it in a more practical manner. For a rigorous mathematical description we refer to the original work [127]. Moreover, we will show how this scheme can be easily implemented in terms of classical-quantum circuits.

8.2.1 Scheme description

Let us consider a quantum algorithm as the process of applying a quantum circuit represented by an unitary operator U to an initial state $|\alpha\rangle$, to obtain the quantum state $|\beta\rangle = U|\alpha\rangle$. The client is able to prepare $|\alpha\rangle$, and the task of the server is applying the quantum circuit of the algorithm. The client wants to prevent the server from obtaining information about $|\alpha\rangle$, so the quantum algorithm is performed using quantum homomorphic encryption. The QHE scheme used in our publication [209] consists of three main procedures, two performed by the client and one by the server. The quantum circuit performing U must be decomposed in Clifford+T gates from the set $\mathcal{G} = \{X, Z, H, S, S^\dagger, CNOT, T, T^\dagger\}$.

Step 1: the client initializes a n-qubit system in the desired initial state $|\alpha\rangle$ of the corresponding quantum algorithm. The qubits are denoted as q_i for $i = 1, .., n$. The client encrypts the system using QOTP encryption [144] before sending the qubits to the server, who will run the algorithm. In order to accomplish this, the client randomly generates a secret key composed of two bit strings of length n. These two bit strings are stored in classical-bitstring variables. Let us denote as x and z these classical-bitstring variables, so we have:

$$x = x_1 x_2 ... x_n, \tag{8.1}$$

$$z = z_1 z_2 ... z_n. \tag{8.2}$$

The client applies Pauli gates X and Z to each qubit q_i in order to encrypt them, depending on the classical bits x_i and z_i which contain the initial secret key at this point. Then, the client prepares the encrypted state

$$|\alpha^{enc}\rangle = \left[\bigotimes_{i=1}^{n} X_i^{x_i} Z_i^{z_i}\right] |\alpha\rangle\,, \tag{8.3}$$

and sends the qubits to the server.

Step 2: The quantum circuit that the server runs can be read sequentially gate by gate. For each gate g in the circuit that has to be evaluated, the server applies its corresponding homomorphic evaluation scheme. Whereas this process consists of simply applying gate g for Clifford gates, it is more complex for $T/T^\dagger$ gates. Each gate evaluation produces a change in the encrypting key of the system, which means that the values stored in classical bits x_i and z_i have to be updated accordingly. For each gate g ($CNOT$) applied to qubit q_i (qubits q_i and q_j), there is a key-updating function, denoted as $f_{g,i}$ ($f_{C,ij}$), that determines how the associated classical bits x_i and z_i must be updated. Then, the server generates a sequence with the key updating functions according to the sequence of gates contained in the quantum circuit. Once the server finishes the evaluation of all the gates, it sends the qubits with the state $|\beta^{enc}\rangle$ back to the client, along with the sequence of key-updating functions.

Step 3: For each function $f_{g,i}$ or $f_{C,ij}$ in the sequence of key-updating functions, the client performs the corresponding key updating of classical bits:

$$x_i, z_i \leftarrow f_{g,i}(x_i, z_i), \tag{8.4}$$

$$(x_i, z_i), (x_j, z_j) \leftarrow f_{C,ij}((x_i, z_i), (x_j, z_j)). \tag{8.5}$$

After completing this sequence, these classical bits contain the final encryption key. Therefore, the client can obtain the decrypted final state applying QOTP decryption with these bits:

$$|\beta\rangle = \left[\bigotimes_{i=1}^{n} X_i^{x_i} Z_i^{z_i}\right] |\beta^{enc}\rangle \,. \tag{8.6}$$

In principle, obtaining the final decrypted state $|\beta\rangle$ is what the client needs in case he wants to utilize it as the initial state for a different algorithm, or if he wants to perform measurements in different basis than the computational one. On the other hand, in most cases the client just wants to measure the final state $|\beta\rangle$ in the computational basis. In such case, the client can measure directly $|\beta^{enc}\rangle$ and obtain the classical bitstring encrypted with the key stored in classical bits x_i. Finally, the client can easily decrypt the classical result using this key and classical XOR operations instead of applying X and Z quantum gates.

8.2.2 Evaluation schemes and key-updating functions for Clifford+T gates

We have only described our QHE scheme in an abstract sense so far. We proceed to show the explicit form of the evaluation schemes and key-updating functions for each quantum gate. In the following, we use a and b to denote constant values of bits, whereas to denote classical-bit variables whose value can change we use x and z.

Let $|\phi\rangle$ be an arbitrary quantum state, like any intermediate state of the quantum algorithm. In case of single qubit gates, for the sake of simplicity we suppose that they act on the first qubit of $|\phi\rangle$, which is encrypted with $x_1 = a$ and $z_1 = b$ in the first qubit q_1, so we have $X_1^a Z_1^b |\phi\rangle$. We already know how the encrypting key changes after applying each of the quantum gates, since the rules for each gate were explained in chapter 2 but it is still useful to revisit this. Let us start with Clifford gates, and in particular the X gate. We have, up to a global phase:

$$X_1 X_1^a Z_1^b |\phi\rangle = X_1^a Z_1^b X_1 |\phi\rangle \,, \tag{8.7}$$

so the classical bits do not change after applying the quantum gate. If we applied a Z gate instead, we would have:

$$Z_1 X_1^a Z_1^b |\phi\rangle = X_1^a Z_1^b Z_1 |\phi\rangle \,, \tag{8.8}$$

so the encrypting key remains unchanged once more. Thus, the key-updating function for both gates is simply the identity:

$$f_{X,i}(x_i, z_i) = f_{Z,i}(x_i, z_i) = (x_i, z_i). \tag{8.9}$$

Next we have the Hadamard H gate, so

$$H_1 X_1^a Z_1^b |\phi\rangle = X_1^b Z_1^a H_1 |\phi\rangle . \tag{8.10}$$

Since in this case the encrypting key changes, the classical bits must be updated to $x_1 = b$ and $z_1 = a$. Then, the key-updating function is

$$f_{H,i}(x_i, z_i) = (z_i, x_i). \tag{8.11}$$

Finally, for the S and $S^\dagger$ gates we have

$$S_1 X_1^a Z_1^b |\phi\rangle = X_1^a Z_1^{a\oplus b} S_1 |\phi\rangle , \tag{8.12}$$

$$S_1^\dagger X_1^a Z_1^b |\phi\rangle = X_1^a Z_1^{a\oplus b} S_1^\dagger |\phi\rangle , \tag{8.13}$$

so in both cases the bits are updated to $x_1 = a$ and $z_1 = a \oplus b$. This means that both gates share the same key-updating function:

$$f_{S,i}(x_i, z_i) = f_{S^\dagger,i}(x_i, z_i) = (x_i, x_i \oplus z_i). \tag{8.14}$$

For the $CNOT$ gate, assuming that the quantum state is encrypted with $x_1 = a_1$, $z_1 = b_1$, $x_2 = a_2$, $z_2 = b_2$ and that the $CNOT$ gate takes qubit q_1 as the control qubit and q_2 as the target qubit, we have

$$CNOT_{12} X_1^{a_1} Z_1^{b_1} X_2^{a_2} Z_2^{b_2} |\phi\rangle = X_1^{a_1} Z_1^{b_1\oplus b_2} X_2^{a_1\oplus a_2} Z_2^{b_2} CNOT |\phi\rangle . \tag{8.15}$$

In this case the classical bits must be updated to $x_1 = a_1$, $z_1 = b_1 \oplus b_2$, $x_2 = a_1 \oplus a_2$, $z_2 = b_2$, so the key-updating function is

$$f_{C,ij}((x_i, z_i), (x_j, z_j)) = ((x_i, z_i \oplus z_j), (x_i \oplus x_j, z_j)). \tag{8.16}$$

With this, the key-updating functions of Clifford gates have been obtained. We can see that the homomorphic evaluation scheme for Clifford gates corresponds trivially to applying these gates. However, for non-Clifford gates the situation is different. Recall that if a T gate is applied to an encrypted state (assuming it is in the first qubit), we have:

$$T_1 X_1^a Z_1^b |\phi\rangle = (S_1^\dagger)^a X_1^a Z_1^{a\oplus b} T_1 |\phi\rangle , \tag{8.17}$$

so there is an additional phase error given by $(S^\dagger)^a$. The bit value a is given by the encrypting-key bit x_1 before the T gate is applied. The server is unable to correct this error since a depends on the initial encrypting key, which the server can not access. Since our QHE scheme is non-interactive, the server can not send the qubit to the client to correct the phase error until the algorithm is finished. We make use of the gate

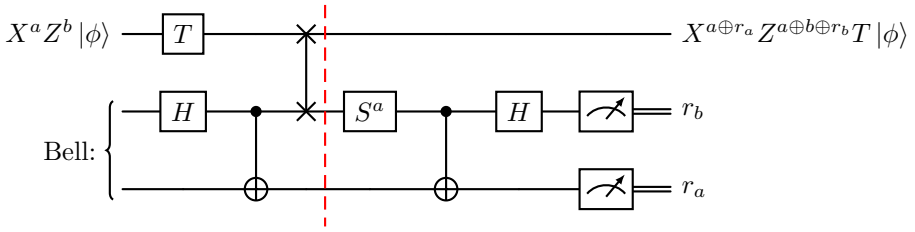


Figure 8.1: Homomorphic evaluation scheme for a T gate using gate teleportation. Besides the T gate, a swap gate, Clifford gates and measurements are performed. The part on the left of the red line is performed by the server, and the part on the right by the client once the server has finished running the quantum algorithm.

teleportation procedure that was introduced in Liang's schemes [127] to teleport a S^a gate to the qubit. We recall this procedure in Figure 8.1. The server creates a Bell state in two ancilla qubits (denoted as Bell) using a H and a $CNOT$ gate. Then, the server applies a swap gate between the problem qubit and the first qubit of the Bell register.

Due to the principle of deferred measurements [139], the client can postpone the S^a-rotated Bell measurement until the server has finished running the quantum circuit. Thus, the server completes the algorithm and then sends the ancilla qubits along with the main qubits of the system. Taking all this into account, the evaluation scheme for a T gate corresponds to the part to the left of the red line in Figure 8.1, and the key-updating function is given by:

$$f_{T,i;l}(x_i, z_i) = (x_i \oplus r_{a_l}, x_i \oplus z_i \oplus r_{b_l}). \tag{8.18}$$

Let L be the number of $T/T^\dagger$ contained in the quantum circuit of the algorithm. For each of these gates, two different ancilla qubits are needed, there are L Bell registers. The new subscript $l = 1, ..., L$ in the key-updating function refers to the Bell register that the client must measure to obtain the classical bits r_{a_l} and r_{b_l}. Then, when the function $f_{T,i;l}$ is read in the sequence of key-updating functions in Step 3, the client has to perform the quantum operations at the right of the red line in Figure 8.1, besides the classical XOR operations. Thus, it can be thought as a kind of classical-quantum key-updating function.

In last place, for a $T^\dagger$ gate we have

$$T_1^\dagger X_1^a Z_1^b |\phi\rangle = S_1^a X_1^a Z_1^{a\oplus b} T_1^\dagger |\phi\rangle . \tag{8.19}$$

Using the same procedure as with the T gate, in this case the teleported S^a combined with the S^a error produces Z^a. This cancels the a superscript in the Z gate, so the key updating function is:

$$f_{T^\dagger,i;l}(x_i, z_i) = (x_i \oplus r_{a_l}, z_i \oplus r_{b_l}). \tag{8.20}$$

8.2.3 Circuit representation

The whole procedure of Step 2 can be represented by a quantum circuit. We also want a circuit representation for the decryption procedure that the client performs in Step 3. The key-updating functions of Clifford gates are based on applying XOR or swap operations, which can be achieved using classical $CNOT$ and swap gates on the classical bits x_i and z_i. Thus, the key-updating process can be represented with classical circuits. In the case of $T/T^\dagger$ gates, the key-updating function can be treated as a classical-quantum procedure, so it can represented using a circuit that contains both classical and quantum gates. Then, the whole procedure performed on Step 3 can be represented with a classical-quantum circuit. In Table 8.1 we summarize the building blocks for the circuits that the server and the client have to run.

As an example of the application of the QHE scheme, we show a quantum algorithm that we want to homomorphically evaluate in Figure 8.2. The client would create the initial state. Then the server would run the evaluation schemes associated to each gate and generate the corresponding sequence of key-updating functions $[f_{H,1}, f_{S,2}, f_{C,12}, f_{T,1;1}, f_{S^\dagger,1}, f_{T^\dagger,1;2}]$. This sequence is then used by the client to update the encrypting keys.

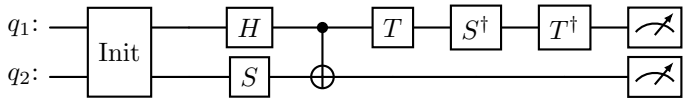


Figure 8.2: Example of a quantum algorithm with 2 $T/T^\dagger$ gates and Clifford gates applied to two qubits. The first block denoted by "Init" is an algorithm used by the client to create the initial state sent to the server. Its homomorphic implementation does not concern us since it is not performed by the server. The remaining Clifford+T gates constitute the server's algorithm. The final step is measuring each qubit.

From this simple circuit, the circuits that must be run in the three steps of the QHE scheme can be showcased. In Figure 8.3(a) we present the circuit that the client must run to initialize the system in Step 1. Notice that the Init gate does not need to be compiled in the Clifford+T gates of the universal set of gates $\mathcal{G}$, since it is performed before encryption. After generating the initial state, the client initializes the classical bits that store the encrypting key at random and applies classically-controlled quantum gates to encrypt the system. Since we have $n = 2$ qubits, there are $2n = 4$ classical bits. In Figure 8.3(b) we show the homomorphic evaluation of the quantum gates of the circuit performed by the server. Since there are two $T/T^\dagger$ gates, two quantum Bell registers are required, which means that four ancilla qubits are used. Finally, in Figure 8.3(c) we can see the classical-quantum circuit that the client has to perform in order to obtain the final encrypting key. Notice that in this example we measure the main qubits,

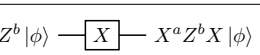
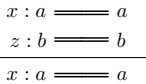
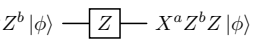
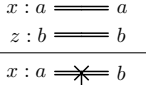
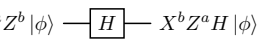
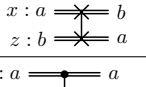
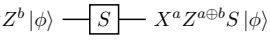
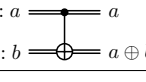
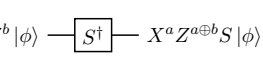
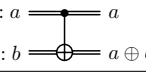
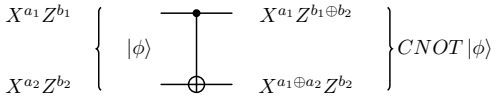
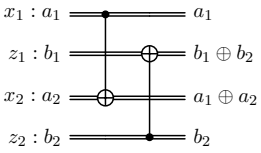
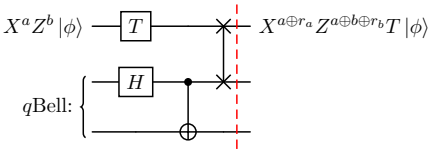
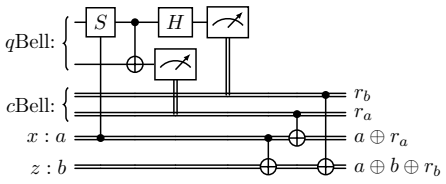
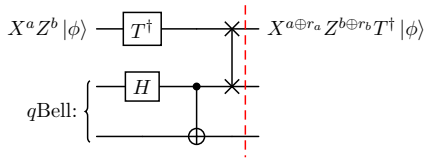
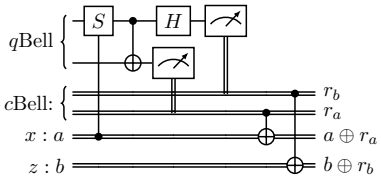
Gate	Server	Client
X		
Z		
H		
S		
$S^\dagger$		
$CNOT$		
T		
$T^\dagger$		

Table 8.1: Classical-quantum circuits for the evaluation schemes performed by the server and the corresponding key-updating function performed by the client, associated to each Clifford+T gate. Quantum bits are represented by single lines, whereas classical bits are represented using double lines. Notice that for $T/T^\dagger$ gates, the client applies a S gate controlled by classical bit x. Since this bit stores the value a, the client ends up applying S^a. For each quantum Bell register, denoted as qBell, there is an associated classical register, denoted by cBell, with two bits. The qubits of the registers qBell start in state $|0\rangle$. We want to remark that there is a fundamental difference regarding the evaluation of $T/T^\dagger$ gates and Clifford gates, since the former makes use of both qubits and classical bits whereas the latter just requires classical bits.

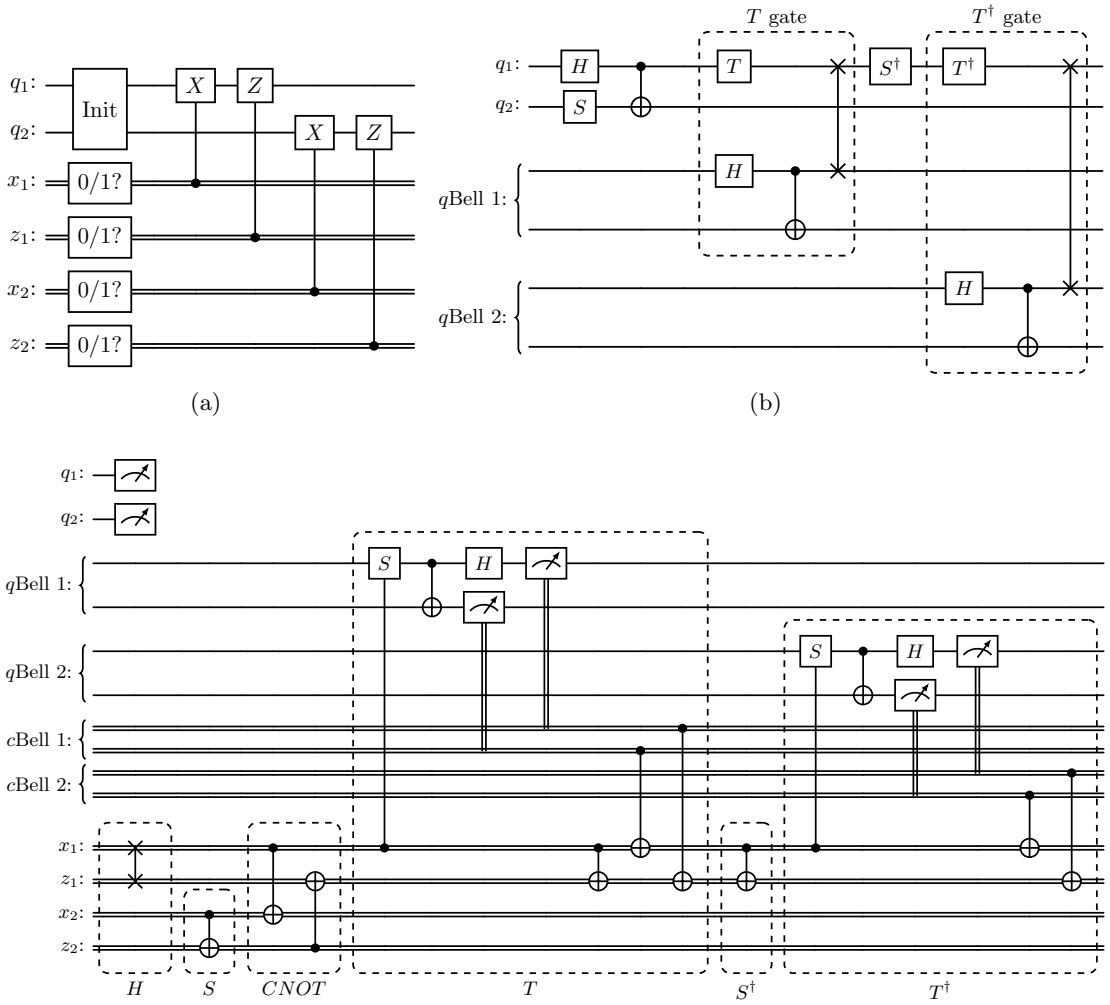


Figure 8.3: Circuits for the three steps of the QHE scheme applied to the example circuit in Figure 8.2. a) Classical-quantum circuit for the initialization performed by the client in Step 1. Qubits q_1 and q_2 begin in the state $|0\rangle$. b) Quantum circuit for the quantum algorithm performed by the server in Step 2. The qubits of the registers qBell start in the state $|0\rangle$. c) Classical-quantum circuit for key-updating and measurement performed by the client in Step 3. Quantum bits are represented using single lines, whereas classical bits are illustrated using double lines.

q_1 and q_2, before the decryption process. Then the classical results must be decrypted with classical bits x_1 and x_2. This procedure will be used in the following simulations in order to see the results before and after decrypting. Nevertheless, we could instead apply X and Z gates controlled by their respective classical bits to obtain the final decrypted quantum state.

8.2.4 Security and resources analysis of the QHE scheme

The QHE scheme that we have reviewed in this section is non-interactive and $\mathcal{F}$-homomorphic since it can be applied to any quantum circuit. Like in Liang's schemes, the encryption is performed using the QOTP protocol. Recall that it has perfect security, because if the encryption bitstrings are chosen at random and used just once, applying this protocol to any arbitrary quantum state ρ produces the totally mixed state [144]:

$$\frac{1}{2^{2n}} \sum_{a,b\in\{0,1\}^n} \left[\bigotimes_{i=1}^{n} X_i^{a_i} Z_i^{b_i}\right] \rho \left[\bigotimes_{i=1}^{n} X_i^{a_i} Z_i^{b_i}\right]^{\dagger} = \frac{\mathbb{1}_{2^n}}{2^n}. \tag{8.21}$$

Combining this with the fact that there is no interaction in the evaluation and decryption processes, then this QHE has also perfect security.

Our QHE scheme only differs from Liang's scheme [127] in regards to the way the key-updating functions are created. As it has been presented our scheme does not have L-quasi-compactness, since the number of key-updating functions scales linearly with the number of gates contained in the circuit. This happens because we have an independent key-updating function per quantum gate, independently of the gate being a Clifford or $T/T^{\dagger}$ gate. However, the scheme can be slightly modified in order to achieve the L-quasi-compactness and make it more similar to Liang's scheme.

Since all key-updating functions related to Clifford gates are deterministic and do not require quantum measurements, a sequence of Clifford key-updating functions can be composed into a single one denoted by $\widetilde{f}_l$. Then, before the first $T/T^{\dagger}$ gate, between each of them and after the last one there is just a single key-updating function $\widetilde{f}_l$, where $l = 1, ..., L+1$. This function can be calculated only by means of XOR operations. Each new bit value is obtained as the XOR sum of the $2n$ classical bits. Since the XOR is self-inverse, in this sum each bit appears at most once. Thus, the number of classical operations that are required for each of the $2n$ new bit values is at most $2n-1$, not scaling with the number of quantum gates that regulate the composed function $\widetilde{f}_l$. In case of the key-updating functions associated with the $T/T^{\dagger}$ gates, at most 3 XOR operations are required. Since there are L functions of this class, the maximum number of classical operations required for decrypting, denoted as $|O_{cl}|$, is

$$|O_{cl}| = (L+1)|\widetilde{f}_l| + 3L = (L+1)2n(2n-1) + 3L, \tag{8.22}$$

where $|\widetilde{f}_l|$ denotes the maximum number of classical operations of $\widetilde{f}_l$. Therefore, the complexity scales as $O(L)$ and the QHE scheme has L-quasi-compactness.

The sequence of key-updating functions for the example circuit of Figure 8.2 would be $[\widetilde{f}_1, f_{T,1;1}, \widetilde{f}_2, f_{T^\dagger,1;2}, \widetilde{f}_3]$, where the composed functions are:

$$\widetilde{f}_1((x_1,z_1),(x_2,z_2)) = ((z_1, x_1 \oplus x_2 \oplus z_2), (x_2 \oplus z_1, x_2 \oplus z_2)), \tag{8.23}$$

$$\widetilde{f}_2((x_1,z_1),(x_2,z_2)) = ((x_1, x_1 \oplus z_1), (x_2, z_2)), \tag{8.24}$$

$$\widetilde{f}_3((x_1,z_1),(x_2,z_2)) = ((x_1, z_1), (x_2, z_2)). \tag{8.25}$$

The first function is an ordered combination of the key-updating functions of the H, S and $CNOT$ gates. The second one corresponds directly to the function of the $S^\dagger$ gate applied to the first qubit, and the third one is simply the identity because there are no more gates after the last $T/T^\dagger$ gate.

Since two ancilla qubits are required for each $T/T^\dagger$ gate and the client has to perform $O(L)$ classical-quantum decrypting operations, the efficiency of the scheme depends on the number of $T/T^\dagger$ gates. Therefore, any quantum algorithm can be implemented efficiently with this scheme provided that L grows as a polynomial with the size of the problem [127].

Even though we have ultimately described a L-quasi-compact scheme making use of the composite Clifford key-updating functions, these functions can not be implemented in Qiskit due to its current limitations. Thus for simulation purposes, we will apply the unmodified scheme with a function per gate in Section 8.4. Still, the composition of Clifford gates for key updating is theoretically feasible, and incorporating them could serve as future work.

Finally, regarding the privacy of the server's circuit, notice that using an independent key-updating function per gate would allow the client to obtain information about the sequence of quantum gates used that are neither X nor Z gates, as the server would not append any key-updating function for these two gates because they are trivial. Even if the composite Clifford functions were used, each Clifford subcircuit in the entire circuit (up to Pauli operators on some qubits) still needs to be known by the client, although not in the original form necessarily. Thus, the security of the server's circuit is compromised, as in the case of Liang's scheme. Nevertheless, the client is typically the one that proposes the circuit to the server, like in the quantum walks we study in this chapter, so that it is not necessary for the server to keep the circuit secret.

8.2.5 QHE scheme with semiclassical framework

In order to perform the homomorphic implementation of a semiclassical walk, the client would create the initial state that represents the classical distribution, and the server

would perform the rest of the circuit shown in Figure 7.3. Notice that besides the usual quantum operators, which can be expressed in Clifford+T gates, there also are intermediate measurements and qubit reset operations. Then, these new operations need their corresponding key-updating functions. Although there are rules for the final measurements and new qubits initialization in the literature [123], they are not treated as intermediate operations in a quantum circuit and the measured state of the system is not considered. Thus, we have to further analyse this case to verify that the rules hold.

We start with the rules of the measurement operation. Suppose there is a qubit that after being measured results in the state $|0\rangle$ with probability p_0 and in the state $|1\rangle$ with probability p_1. If a X gate is applied these probabilities are exchanged. Then, if the resulting state after measuring a qubit is $|m\rangle$, the encrypted state with X^a results in the state $X^a|m\rangle$, where $m \in \{0,1\}$. Since applying a Z gate does not affect the measurement probabilities, this gate does not play any role. Moreover, as the resulting state of a measurement is always an eigenstate of the Z gate, we can consider that the resulting state is not encrypted by any Z gate without loss of generality. Taking all these information into account, we see that the key-updating function associated to the measurement operator is

$$f_{M,i}(x_i, z_i) = (x_i, 0). \tag{8.26}$$

In case of the reset operation, the final state is always $|0\rangle$ regardless of the encrypting key of the qubit, so this final state is not encrypted by any gate. Then, the associated key-updating function is

$$f_{R,i}(x_i, z_i) = (0, 0). \tag{8.27}$$

In Table 8.2 the circuits that the sever and the client must apply for these two operations are shown. We want to remark that these operations behave as Clifford gates, which means that their key-updating functions are deterministic and can be composed with Clifford key-updating functions in order to reach the L-quasi-compactness.

Finally, we proceed to analyse the security of the algorithm. In this case, as the initial state represents a classical distribution, this state is a linear combination of the computational basis states with no relative phases or a mixed state. Thus, all the information can be obtained if the measurements are only performed in the computational basis, whose results are invariant under the application of Z gates. Then, this state can be encrypted using just X gates. Even though the length of the encrypting key is half of the standard one as it only contains a single bitstring instead of two, the initial state also contains less information than an usual quantum state. Thus, for states representing classical distributions, applying QOTP encryption with X gates provides perfect security too. For a classical distribution state, it does not matter if it is a mixed state or a

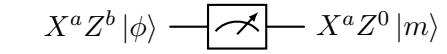
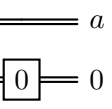
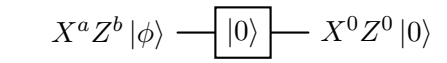
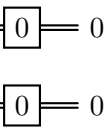
Gate	Server	Client
		
$\|0\rangle$		

Table 8.2: Classical-quantum circuits for the evaluation schemes performed by the server and the key-updating function performed by the client, corresponding to measurement and reset operations. Quantum bits are represented by single lines and classical bits are represented by double lines.

quantum superposition. Let us consider the mixed state

$$\rho_c = \sum_{i=0}^{N-1} p_i \left|i\right\rangle \left\langle i\right|, \tag{8.28}$$

where p_i are the different probabilities of the classical distribution represented by this state. It is trivial that $\rho_c = Z_i \rho_c Z_i^\dagger$ for any Z gate acting on any qubit. Therefore, if we apply all the random keys with X gates we recover the totally mixed state again:

$$\frac{1}{2^n} \sum_{a\in\{0,1\}^n} \left[\bigotimes_{i=1}^{n} X_i^{a_i}\right] \rho_c \left[\bigotimes_{i=1}^{n} X_i^{a_i}\right]^\dagger = \frac{1}{2^{2n}} \sum_{a,b\in\{0,1\}^n} \left[\bigotimes_{i=1}^{n} X_i^{a_i} Z_i^{b_i}\right] \rho_c \left[\bigotimes_{i=1}^{n} X_i^{a_i} Z_i^{b_i}\right]^\dagger = \frac{\mathbb{1}_{2^n}}{2^n}, \tag{8.29}$$

where in the last step equation (8.21) was used.

8.3 $T/T^\dagger$ gate complexity of Szegedy quantum circuits

As we already know, a quantum algorithm can be implemented efficiently using a QHE scheme if the number of $T/T^\dagger$ gates contained in the circuit scales polynomially with the size of the problem. In the case of Szegedy quantum walks, the size of the problem is expressed in terms of the number of nodes of the graph N. On the other hand, the number of time steps of the quantum walk depends on the particular algorithm chosen. As long as the algorithm is efficient, the number of time steps scales polynomially with N. As an example, this parameter scales with the square root of N [184, 200] for search problems in graphs. Therefore, the efficiency of the implementation ultimately depends on the quantum circuit constructed for the unitary evolution operator U_w in (7.6).

In order to obtain a quantum circuit for the Szegedy quantum unitary U_w, two registers of qubits are needed. Each register represents N possible states for a graph with N nodes. From this point onwards, we consider that each register has n qubits, being $N = 2^n$, so the total number of qubits contained in the circuit is $2n$. Szegedy quantum walk unitary evolution U_w (7.6) is composed of two operators: the swap S_w and the reflection R. For a general circuit, we have to diagonalize the reflection operator (7.7) using the update operator V [210, 214] whose action was defined in (7.11), so we have:

$$V^\dagger R V = 2\sum_{i=0}^{N-1} V^\dagger |\psi_i\rangle \langle\psi_i| V - \mathbb{1} = 2\sum_{i=0}^{N-1} |i\rangle_1\langle i| \otimes |0\rangle_2\langle 0| - \mathbb{1} = \mathbb{1} \otimes D, \tag{8.30}$$

where

$$D = 2|0\rangle_2\langle 0| - \mathbb{1} \tag{8.31}$$

is a diagonal operator that acts on the second register. Thus, the reflection operator R can be decomposed as $R = V(\mathbb{1} \otimes D)V^\dagger$. Using this fact, we can implement a general circuit for the unitary operator U_w of the Szegedy quantum walk. This circuit is shown in Figure 8.4, where there are four operators. The swap operator S_w is constructed using swap gates that act between the qubits of the first and second registers. Since the swap gates can be decomposed using three $CNOT$ gates [139], they do not contribute to the number of $T/T^\dagger$ gates.

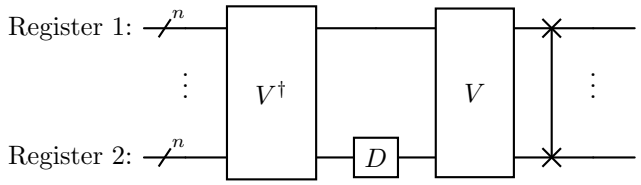


Figure 8.4: General quantum circuit for the unitary evolution operator U_w of Szegedy quantum walk in terms of the update operator V, a diagonal operator D and the swap operator S.

The diagonal operator D is a reflection around the state $|0\rangle_2$ in the second register. This operator only depends on the number n of qubits in each register and is independent of the type of graph. The circuit for this operator is shown in Figure 8.5. Before continuing, let us fix some notation about controlled gates for the rest of the chapter. We denote by single-controlled-U gate a general U gate that is controlled by $n_c = 1$ control qubits, whereas we denote by multi-controlled-U gate a general U gate that is controlled by $n_c > 1$ control qubits. The $CNOT$ gate is a single-controlled-X gate, whereas a Toffoli gate corresponds to a multi-controlled-X gate with $n_c = 2$ control qubits. Since the usual controlled gates that activate when the control qubits are in the state $|1\rangle$ can be transformed into controlled gates that activate when the control qubits are in the state $|0\rangle$, surrounding them using Clifford X gates, there is no need

to consider the particular controlling state for obtaining the number of $T/T^\dagger$ gates. In the case of the operator D, we have a multi-controlled-X gate with $n_c = n - 1$ control qubits. This type of gate can be decomposed into Toffoli gates using ancilla qubits as shown in Figure 8.6(a). For n_c control qubits, the decomposition yields $2n_c - 3$ Toffoli gates, so for $n_c = n - 1$ we have $2n - 5$ Toffoli gates. Furthermore, a Toffoli gate can be decomposed into Clifford+T gates as shown in Figure 8.7, so that one of these gates adds 7 $T/T^\dagger$ gates. Thus, the operator D contains $7(2n - 5) = 14n - 35$ $T/T^\dagger$ gates. Since $n = \log_2(N)$, this operator grows logarithmically and it can be homomorphically implemented in an efficient manner for any graph. Note that this formula only applies to $n \geq 3$, since for $n \leq 2$ the number of $T/T^\dagger$ gates in the circuit is zero.

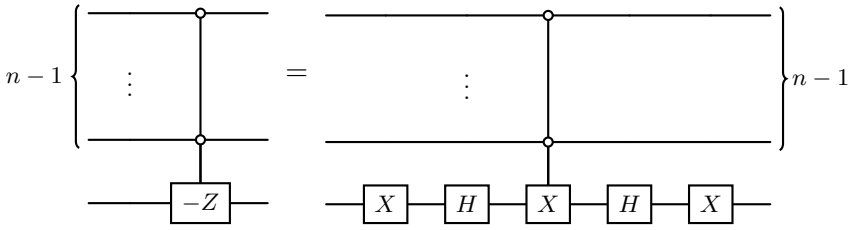


Figure 8.5: Quantum circuit for the diagonal operator D in Figure 8.4. It can be implemented using a multi-controlled-$(-Z)$ gate. Since $-Z = XZX$ and $Z = HXH$, this gate can be implemented with a multi-controlled-X gate.

The update operator $V^\dagger$ is the inverse of the operator V. Its circuit is constructed inverting the order of the gates present in V and substituting each of these gates by their inverse. If a circuit is compiled in Clifford+T gates, the inverse gates are also Clifford+T gates, which means that the number of $T/T^\dagger$ gates is conserved. Thus, both operators contain the same number of $T/T^\dagger$ gates. Let L_V be the number of $T/T^\dagger$ gates contained in the operator V. Then the total number of $T/T^\dagger$ gates contained in the circuit for U_w, denoted by L_U, can be expressed as:

$$L_U = 14n - 35 + 2L_V. \tag{8.32}$$

Thus, the efficiency of the implementation depends only on the update operator V. This is also the case for a semiclassical walk, since the quantum circuit is composed of the V and U_w operators, and measurements and reset operations behave as Clifford gates regarding the $T/T^\dagger$ count.

The number of $T/T^\dagger$ gates of the update operator V depends on the particular graph chosen. In the following sections we present the decomposition of the update operator V for different types of graphs whose symmetry allows the construction of efficient circuits. These graphs are represented in Figure 8.8. The main feature of these graphs is the quantum circuits they lead to, which are simple toy models for testing Szegedy quantum walk simulations. They can also provide insights for more complex graphs. Furthermore, some of these graphs have applications in search algorithms [202]. Quantum circuits

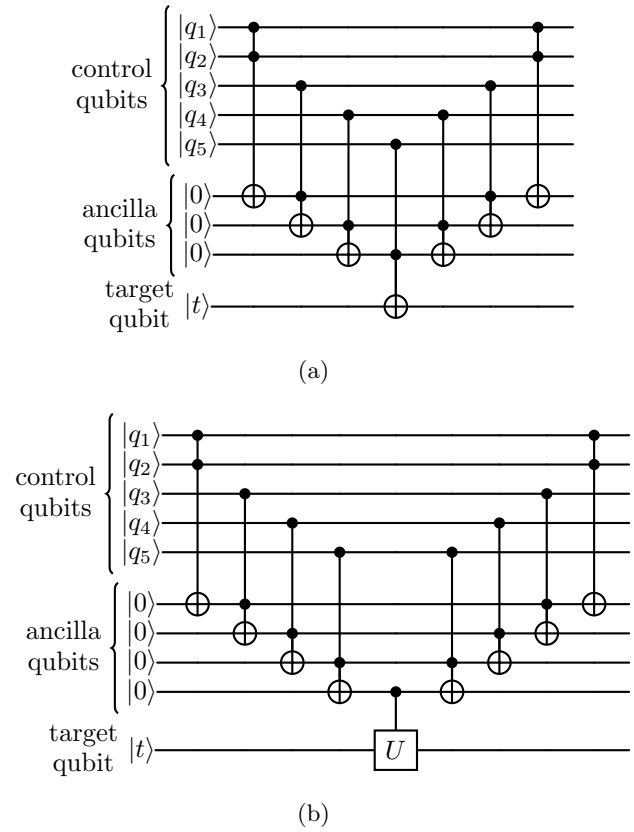


Figure 8.6: a) Decomposition of a multi-controlled-X gate that contains $n_c = 5$ control qubits into $2n_c - 3 = 7$ Toffoli gates using $n_c - 2 = 3$ ancilla qubits. b) Decomposition of a multi-controlled-U gate that contains $n_c = 5$ control qubits into $2n_c - 2 = 8$ Toffoli gates and a single-controlled-U gate using $n_c - 1 = 4$ ancilla qubits. These ancilla qubits must start in state $|0\rangle$ and they also end up in the state $|0\rangle$. Thus, they can be reused for all multi-controlled gates in the circuit. The total number of ancilla qubits only depends on the largest multi-controlled gate, providing at most $n - 1$ ancilla qubits for a circuit with n qubits. Moreover, if they are provided by the client, they can be encrypted, as long as the initial state before encryption has these qubits in state $|0\rangle$.

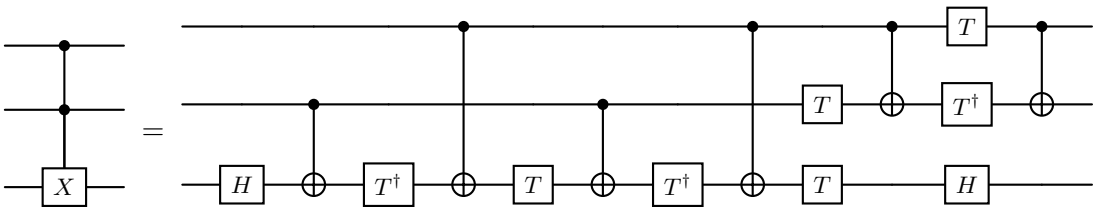


Figure 8.7: Decomposition of a Toffoli gate into Clifford+T gates using 7 $T/T^\dagger$ gates.

derived from these graphs that scale polylogarithmically in the number of gates have been proposed [214]. Nevertheless, these circuits are given in terms of multi-controlled gates and also general rotation gates, whose decomposition in Clifford+T gates is more complex.

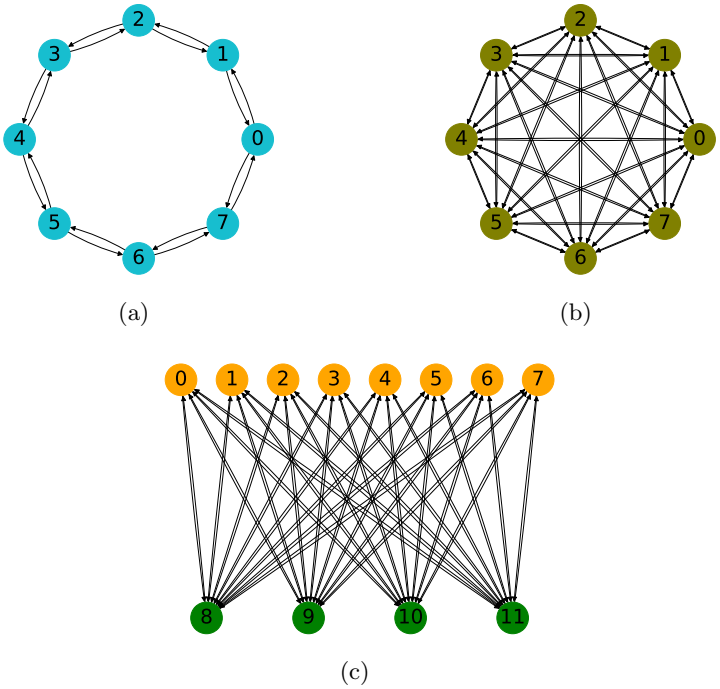


Figure 8.8: Representation of different types of graphs with symmetry. a) Cycle graph with $N = 8$ nodes. b) Complete graph with $N = 8$ nodes. c) Complete bipartite graph with a set of $N_1 = 8$ nodes shown in orange and another set of $N_2 = 4$ nodes shown in green. The graphs have been plotted using the python library NetworkX [215].

8.3.1 Cyclic graph

A cycle graph is shown in Figure 8.8(a). It is composed of a closed line of N nodes, where each node is only connected to its front node and its back node. The transition matrix is

$$G_{ji} = \frac{1}{2}\delta_{i,j-1} + \frac{1}{2}\delta_{i,j+1}. \tag{8.33}$$

The quantum circuit for the update operator V [214] is presented in Figure 8.9. In our publication [209] we used big-endian ordering convention for the order of qubits. This way, the left-most qubit in the tensor product of a multi-qubit state corresponds to the upper-most qubit of the circuit. The contribution to the number of $T/T^\dagger$ gates comes

from the permutation operators, denoted by $\mathcal{P}^+$. The action of these operators on m qubits adds 1 to each computational basis state:

$$\mathcal{P}^+ |x\rangle = |x + 1 \bmod m\rangle . \tag{8.34}$$

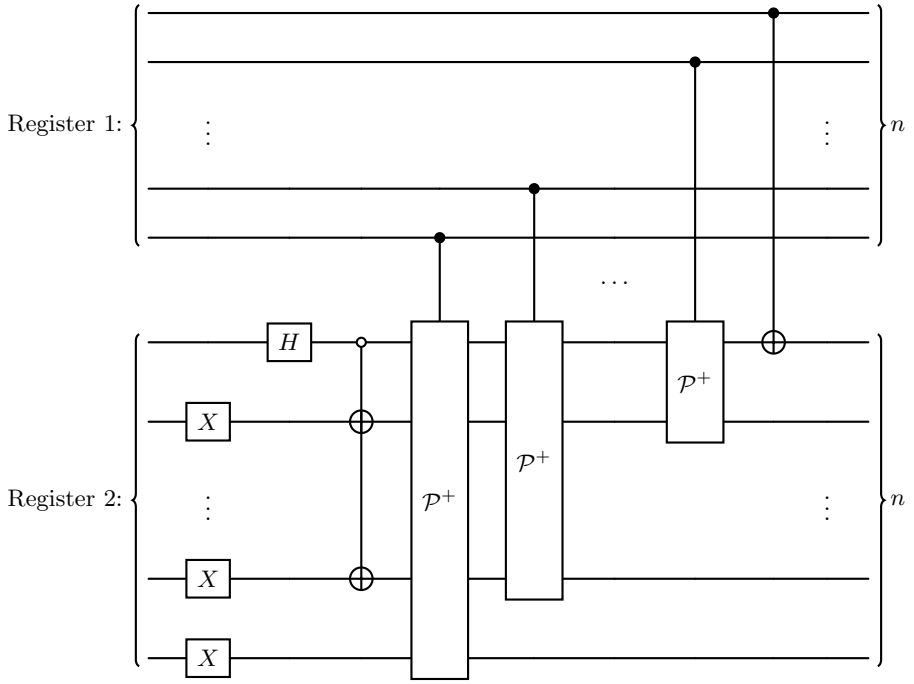


Figure 8.9: Quantum circuit for the update operator V for the Szegedy quantum walk on a cycle graph.

The quantum circuit for a controlled-$\mathcal{P}^+$ operator that acts on m qubits is shown in Figure 8.10. There are multi-controlled-X gates with different amounts of control qubits, starting from $n_c = 2$ until $n_c = m$ control qubits. Once again, each of these gates can be decomposed into $2n_c - 3$ Toffoli gates using the decomposition of Figure 8.6(a). Then, since each Toffoli gate can be decomposed into 7 $T/T^\dagger$ gates, we have that the number of $T/T^\dagger$ gates for a controlled-$\mathcal{P}^+$ operator is:

$$L_{\mathcal{P}^+} = 7 \sum_{n_c=2}^{m} (2n_c - 3) = 7(m^2 - 2m + 1). \tag{8.35}$$

In the circuit for the update operator V we have controlled-$\mathcal{P}^+$ operators that act on different numbers of qubits, starting from $m = 2$ to $m = n$ target qubits. Thus, adding the number of $T/T^\dagger$ gates of each controlled-$\mathcal{P}^+$ operator we obtain the number of $T/T^\dagger$ gates for the update operator V:

$$L_V = 7 \sum_{m=2}^{n} (m^2 - 2m + 1) = \frac{7}{6}(2n^3 - 3n^2 + n). \tag{8.36}$$

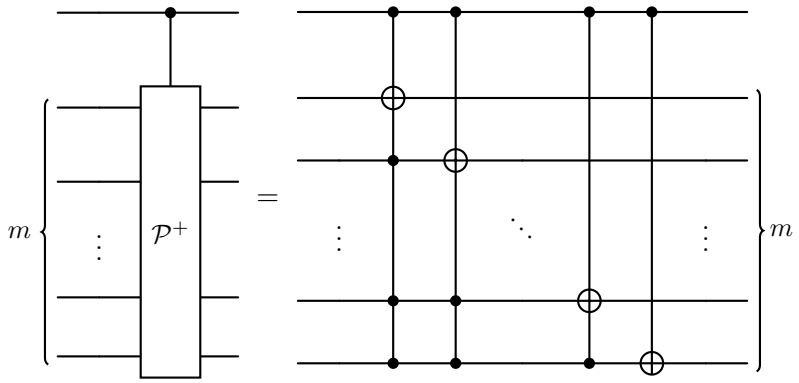


Figure 8.10: Quantum circuit for a controlled-$\mathcal{P}^+$ operator applied to m qubits.

Substituting in equation (8.32) we obtain the number of $T/T^\dagger$ gates required for the unitary evolution operator U_w in the case of the cycle graph:

$$L_U = \frac{14}{3}n^3 - 7n^2 + \frac{49}{3}n - 35. \tag{8.37}$$

Finally, using the fact that $n = \log_2(N)$ we can express L_U in terms of the number of nodes:

$$L_U = \frac{14}{3}\log_2^3(N) - 7\log_2^2(N) + \frac{49}{3}\log_2(N) - 35. \tag{8.38}$$

As it can be observed, in the asymptotic limit the number of $T/T^\dagger$ gates scales as $O(\log_2^3(N))$, so it scales polylogarithmically with the size of the problem. Therefore this quantum walk can be implemented efficiently using a QHE scheme.

8.3.2 Complete graph

A complete graph is shown in Figure 8.8(b). It is formed by N nodes, where each node connects to all the remaining nodes and without self-loops. The transition matrix is

$$G_{ji} = \frac{1}{N-1}(1 - \delta_{ij}). \tag{8.39}$$

The quantum circuit for the update operator V [214] is shown in Figure 8.11. The contribution to the number of $T/T^\dagger$ gates in this circuit originates from the same $\mathcal{P}^+$ operators as in the cycle graph and a new contribution due to the operator $\widetilde{V}$, whose quantum circuit is presented in Figure 8.12. Then, the number of $T/T^\dagger$ gates for the unitary evolution operator U_w in the case of the complete graph is

$$L_U = \frac{14}{3}n^3 - 7n^2 + \frac{49}{3}n - 35 + 2L_{\widetilde{V}}. \tag{8.40}$$

In Figure 8.12 we have identified five different types of sources that contribute to the number of $T/T^\dagger$ gates. Then we can express this number for the $\widetilde{V}$ operator in terms of these sources as

$$L_{\widetilde{V}} = L_{CX} + L_{CH} + L_{CA} + L_{CR} + L_R. \tag{8.41}$$

Since we are interested in the asymptotic scaling, we assume that each register contains $n \geq 3$ qubits, so that all these sources are present in the circuit.

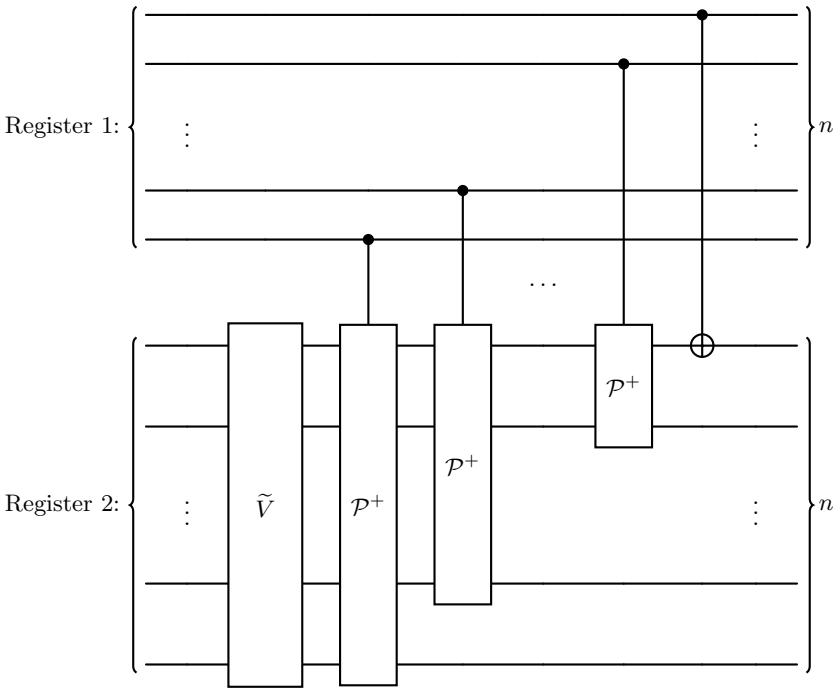


Figure 8.11: Quantum circuit for the update operator V for Szegedy quantum walk on a complete graph.

The term L_{CX} corresponds to the number of $T/T^\dagger$ gates due to the decomposition of the last multi-controlled-X gate, which has $n_c = n-1$ control qubits. Making use of the decomposition in Toffoli gates of Figure 8.6(a) and taking into account that each Toffoli gate contains 7 $T/T^\dagger$ gates, we obtain that

$$L_{CX} = 7(2(n-1)-3) = 14n - 35. \tag{8.42}$$

Next we have the term L_{CH}, which depends on the decomposition of the last multi-controlled-H gate. Since the Hadamard gate can be expressed using $H = AXA^\dagger$, with $A = SHTHS^\dagger H$, the multi-controlled-H gate can be decomposed into a multi-controlled-X gate in which the target qubit is surrounded by A and $A^\dagger$. Thus, this unitary transformation costs two $T/T^\dagger$ gates, which added to the contribution of the

multi-controlled-X gate results in:

$$L_{CH} = 7(2(n-1)-3)+2 = 14n-33. \tag{8.43}$$

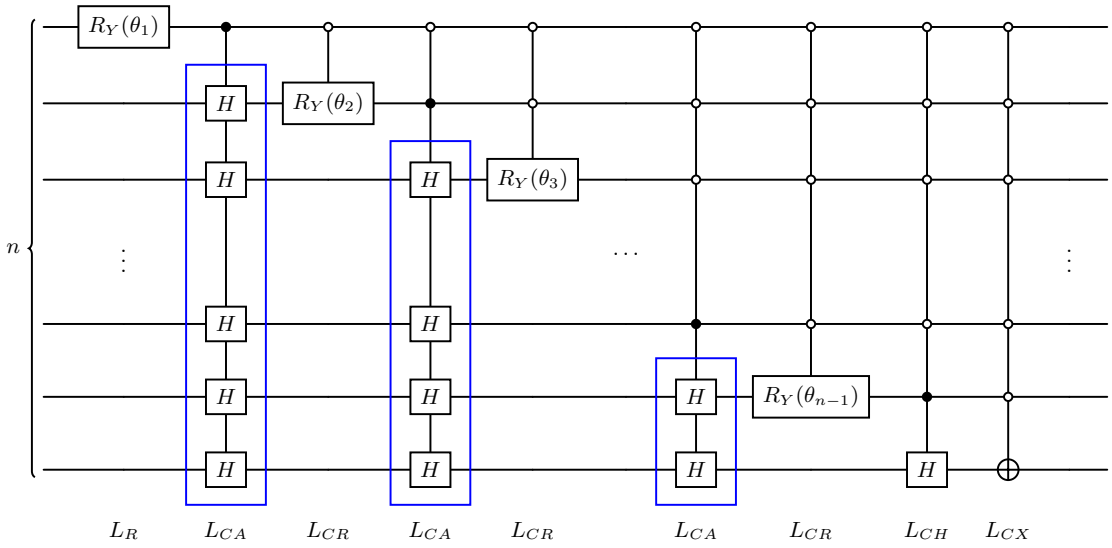


Figure 8.12: Quantum circuit of the operator $\widetilde{V}$ that appears in Figure 8.11. The angles are given by $\theta_i = 2\arccos\left(\sqrt{(2^{n-i}-1)/(2^{n-i+1}-1)}\right)$. The formula of the $R_Y(\theta)$ gate is given in equation (8.45).

The remaining controlled-H gates appear in the form of controlled arrays of H gates, shown as blue boxes in Figure 8.12. These controlled arrays contribute to the term L_{CA}. In this case we treat each array as a general unitary gate U, so we can apply a decomposition into $2n_c - 2$ Toffoli gates and an array of H gates that are controlled by a single qubit, as shown in Figure 8.6(b) (taking $U = H$). On one hand, for the Toffoli gates we have to take into account that there are different multi-controlled gates regarding their number of control qubits, starting from $n_c = 1$ to $n_c = n-2$ control qubits, and that each Toffoli gate contains 7 $T/T^\dagger$ gates. On the other hand, each single-controlled-array with h Hadamard gates corresponds to h single-controlled-H gates. As we did previously, these gates can be converted into Clifford $CNOT$ gates at the cost of two $T/T^\dagger$ gates. We have to consider the different arrays of H gates present, which start from $h = 2$ to $h = n-1$ Hadamard gates. Thus, the total contribution of all these gates can be expressed as

$$L_{CA} = 7\sum_{n_c=1}^{n-2}(2n_c-2) + 2\sum_{h=2}^{n-1} h = 8n^2 - 36n + 40. \tag{8.44}$$

The term L_{CR} corresponds to the number of $T/T^\dagger$ gates added by the controlled-R_Y gates. These gates can be decomposed into $2n_c - 2$ Toffoli gates and a single-controlled-R_Y gate applying the decomposition of Figure 8.6(b) once more. The contribution due

to the Toffoli gates is the same as it was for the term L_{CA}, since we have multi-controlled gates from $n_c = 1$ to $n_c = n - 2$ control qubits. Given the matrix form of a R_Y gate as

$$R_Y(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}, \tag{8.45}$$

we have that $R_Y(\theta) = XR_Y(-\theta/2)XR_Y(\theta/2)$. Then a single-controlled-R_Y gate can be decomposed into two R_Y gates using the decomposition shown in Figure 8.13. Thus, the term L_{CR} can be expressed as

$$L_{CR} = 7\sum_{n_c=1}^{n-2}(2n_c - 2) + (n-2)2L_R = 7n^2 - 35n + 42 + (2n-4)L_R, \tag{8.46}$$

where L_R represents the number of $T/T^\dagger$ gates required to compile a R_Y gate, which so far we consider that it does not depend on the angle θ.

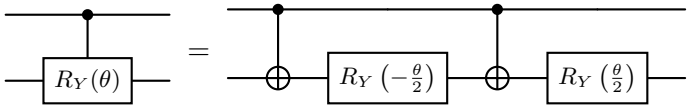


Figure 8.13: Decomposition of a single-controlled-R_Y gate into two $CNOT$ gates and two R_Y gates.

Taking into account all the contributions in eq (8.41) we obtain the number of $T/T^\dagger$ gates for the operator $\widetilde{V}$:

$$L_{\widetilde{V}} = 15n^2 - 43n + 14 + (2n-3)L_R. \tag{8.47}$$

Substituting in eq (8.40) we finally obtain the number of $T/T^\dagger$ gates of the unitary evolution operator U_w:

$$L_U = \frac{14}{3}n^3 + 23n^2 - \frac{209}{3}n - 7 + (4n-6)L_R. \tag{8.48}$$

We can express it in terms of the number of nodes of the graph, leading to

$$L_U = \frac{14}{3}\log_2^3(N) + 23\log_2^2(N) - \frac{209}{3}\log_2(N) - 7 + (4\log_2(N) - 6)L_R. \tag{8.49}$$

If we assume that a R_Y gate can be implemented efficiently, with a scaling that grows slower than the cube of the logarithm of the number of nodes, then the number of $T/T^\dagger$ gates scales as $O(\log_2^3(N))$ and the unitary evolution U_w can be implemented efficiently using a QHE scheme.

Solovay-Kitaev theorem states that any single qubit gate U can be implemented using Clifford+T gates with $O(\log^c(1/\varepsilon))$ gates, where c is a small constant approximately

equal to 2 and ε represents the error of the approximation using a finite number of gates [139]:

$$\varepsilon \geq \|\bar{U} - U\|, \tag{8.50}$$

where $\bar{U}$ is the approximated gate. Then, R_Y gates can be implemented efficiently in principle for any angle θ. Furthermore, the number of $T/T^\dagger$ gates required to implement a R_Y gate is the same as for a R_Z gate due to $R_Y(\theta) = SHR_Z(\theta)HS^\dagger$. There are more efficient algorithms for decomposing a R_Z in $T/T^\dagger$ gates, such as the algorithm [216] which claims that any R_Z gate requires at worst a number of $T/T^\dagger$ gates

$$L_R = 4\log_2(1/\varepsilon) + O(\log(\log(1/\varepsilon)). \tag{8.51}$$

The error ε is due to the approximation of a single R_Z gate. The total error of the algorithm denoted by ϵ scales linearly with the number of gates that are decomposed approximately. Since the unitary evolution U_w is decomposed into $4\log_2(N) - 6$ R_Y gates, in the case of a quantum algorithm that uses t time steps of the walk we have that the error per R_Y gate is

$$\varepsilon = \frac{\epsilon}{(4\log_2(N) - 6)t}. \tag{8.52}$$

Thus, we can express the cost of a single R_Y gate as

$$L_R = 4\log_2((4\log_2(N) - 6)t/\epsilon) + O(\log(\log((4\log_2(N) - 6)t/\epsilon)). \tag{8.53}$$

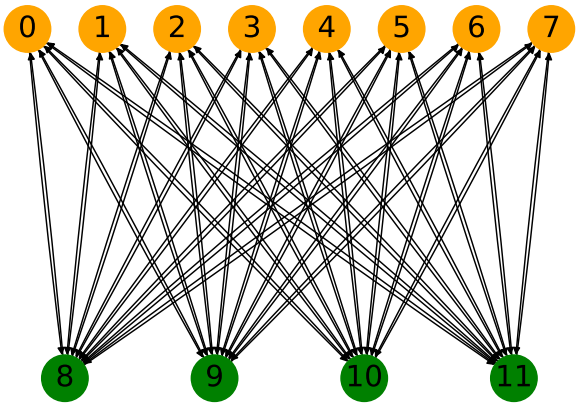
If the algorithm is efficient then the number t of time steps scales as a polynomial in N. An example of such case is the quantum walk search algorithm with sinks, which requires $O(\sqrt{N/M})$ time steps for finding one of the M marked nodes [184]. Therefore, the assumption that L_R scales slower than $O(\log_2^3(N))$ is fulfilled and this quantum walk can be efficiently implemented using a QHE scheme.

8.3.3 Bipartite graph

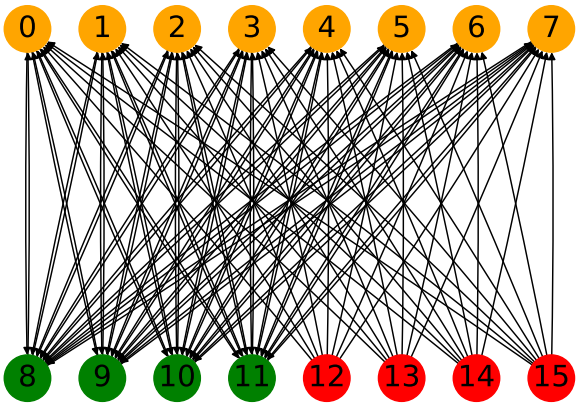
A complete bipartite graph is shown in Figure 8.8(c). It is composed of two sets of nodes denoted by N_1 and N_2 respectively, so that it has $N = N_1 + N_2$ nodes in total. Each node links to all the nodes of the other set, but there are no edges that connect two nodes inside the same set. Let us denote the set of indexes for the first N_1 nodes as V_1 and for the remaining N_2 nodes as V_2. The corresponding transition matrix in this case is:

$$G_{ji} = \begin{cases} \dfrac{1}{N_2} & \text{if } i \in V_1 \text{ and } j \in V_2, \\ \dfrac{1}{N_1} & \text{if } i \in V_2 \text{ and } j \in V_1, \\ 0 & \text{otherwise.} \end{cases}$$

Let n_1 and n_2 be two integer numbers so that $N_1 = 2^{n_1}$ and $N_2 = 2^{n_2}$ with $n_2 \leq n_1$. In order to construct a quantum circuit for this graph we require $n = n_1 + 1$ qubits, so that the circuit simulates the Szegedy quantum walk in an augmented graph with $2^n = 2N_1 \geq N_1 + N_2$ nodes, while still recovering the original Hilbert space of the bipartite graph with $N_1 + N_2$ nodes as an invariant subspace.



(a)



(b)

Figure 8.14: a) Representation of a complete bipartite graph with a set of $N_1 = 8$ nodes shown in orange and another set of $N_2 = 4$ nodes shown in green. b) Representation of an augmented complete bipartite graph with a third set of $N_1 - N_2 = 4$ nodes shown in red.

We have slightly modified the circuit found in the literature [214] in the case of the bipartite graph in order to obtain a single update operator V like it was shown in Figure 8.4. The quantum circuit that we have constructed for the update operator is shown in Figure 8.15. Below we provide a proof of correctness for our proposed quantum circuit.

We start by showing the complete bipartite graph again in 8.14(a) and presenting its corresponding augmented graph with $2^n = 2N_1 \geq N_1 + N_2$ nodes, shown in figure 8.14(b), along it. This augmented graph has a third set with $N_1 - N_2$ additional nodes marked in red, which link to the N_1 nodes of the first set. Nevertheless, no node in the graph links to them.

The Hilbert space of the original bipartite graph with $N_1 + N_2$ nodes is

$$\mathcal{H} = \text{span}\{|i\rangle_1 |j\rangle_2 , \ i, j = 0, 1, ..., N_1 + N_2 - 1\}. \tag{8.54}$$

We proceed to prove that it can be recovered as an invariant subspace of the Hilbert space of the augmented graph, which is

$$\widetilde{\mathcal{H}} = \text{span}\{|i\rangle_1 |j\rangle_2 , \ i, j = 0, 1, ..., 2N_1 - 1\}. \tag{8.55}$$

We denote the set of indexes for the first set of N_1 nodes as $\mathcal{V}_1$, for the second set of N_2 nodes as $\mathcal{V}_2$, and for the new third set of $N_1 - N_2$ nodes as $\mathcal{V}_3$. Then the transition matrix for the augmented graph is

$$\widetilde{G}_{ji} = \begin{cases} \dfrac{1}{N_2} & \text{if } i \in \mathcal{V}_1 \text{ and } j \in \mathcal{V}_2, \\ \dfrac{1}{N_1} & \text{if } i \in \mathcal{V}_2 \text{ and } j \in \mathcal{V}_1, \\ \dfrac{1}{N_1} & \text{if } i \in \mathcal{V}_3 \text{ and } j \in \mathcal{V}_1, \\ 0 & \text{otherwise.} \end{cases} \tag{8.56}$$

The unitary evolution operator for the Szegedy quantum walk in this graph, denoted by $\widetilde{U}_w$, is

$$\widetilde{U}_w = S_w \left[2 \sum_{i=0}^{2N_1-1} |\psi_i\rangle \langle\psi_i| - \mathbb{1} \right] = S_w \left[2 \sum_{i \in (\mathcal{V}_1 \cup \mathcal{V}_2)} |\psi_i\rangle \langle\psi_i| + 2 \sum_{i \in \mathcal{V}_3} |\psi_i\rangle \langle\psi_i| - \mathbb{1} \right], \tag{8.57}$$

where we have separated the sum in two terms, the first one corresponding to the $|\psi_i\rangle$ states of the original nodes, and the other term with the states $|\psi_i\rangle$ of the added nodes. From equation (7.4) and equation (8.56) we have that

$$|\psi_i\rangle = \begin{cases} |i\rangle_1 \otimes \displaystyle\sum_{k \in \mathcal{V}_2} \frac{1}{\sqrt{N_2}} |k\rangle_2 & \text{if } i \in \mathcal{V}_1, \\ |i\rangle_1 \otimes \displaystyle\sum_{k \in \mathcal{V}_1} \frac{1}{\sqrt{N_1}} |k\rangle_2 & \text{if } i \in \mathcal{V}_2, \\ |i\rangle_1 \otimes \displaystyle\sum_{k \in \mathcal{V}_1} \frac{1}{\sqrt{N_1}} |k\rangle_2 & \text{if } i \in \mathcal{V}_3. \end{cases} \tag{8.58}$$

If we select an arbitrary state in the original Hilbert space $\mathcal{H}$, we can see that it is a linear combination of the computational basis whose first register has an index in $\mathcal{V}_1 \cup \mathcal{V}_2$. Thus, it is perpendicular to the states $|\psi_i\rangle$ for $i \in \mathcal{V}_3$. Therefore, the second term in the sum of equation (8.57) does not affect these arbitrary states and the action of the operator is the same as that of the original graph evolution operator:

$$U_w = S_w \left[2 \sum_{i=0}^{N_1+N_2-1} |\psi_i\rangle \langle\psi_i| - \mathbb{1} \right]. \tag{8.59}$$

Therefore, the original Hilbert space $\mathcal{H}$ is invariant under the action of $\widetilde{U}_w$, and the action of the operator remains the same as the original quantum walk. The reason for this is that the extra $N_1 - N_2$ nodes are not linked by any node. Nevertheless, it does not matter what nodes they link to. It is arranged in this manner for convenience in order to construct a quantum circuit for the update operator.

As we know, the action of the update operator V is defined as

$$V |i\rangle_1 |0\rangle_2 = |\psi_i\rangle . \tag{8.60}$$

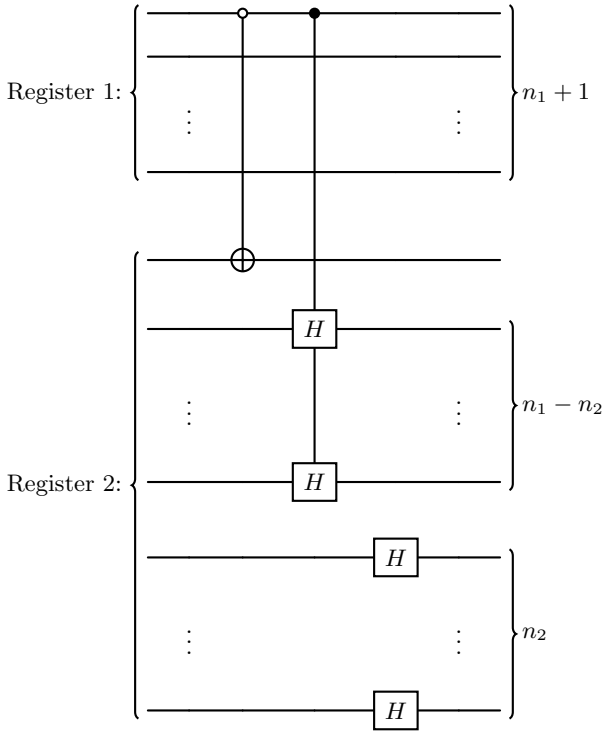


Figure 8.15: Quantum circuit for the update operator V for Szegedy quantum walk on a complete bipartite graph.

The quantum circuit that we propose for implementing it is shown in Figure 8.15. For an initial state $|i\rangle_1 |0\rangle_2$ with $i \in \mathcal{V}_1$, the first qubit of the first register is in the state $|0\rangle$. Then, a X gate is applied to the first qubit of the second register, transforming it to the state $|1\rangle$. Any computational basis state with the first qubit in the state $|1\rangle$ corresponds to the second half of the nodes. After applying the Hadamard gates to the last n_2 qubits, the second register ends up in a linear combination of the first $2^{n_2} = N_2$ states $|k\rangle_2$ of the second half of the nodes. Thus, the resulting state is a linear combination of all the nodes in $\mathcal{V}_2$ and the state $|\psi_i\rangle$ for $i \in \mathcal{V}_1$ is obtained. If instead the initial state is $|i\rangle_1 |0\rangle_2$ with $i \in \mathcal{V}_2$ or $i \in \mathcal{V}_3$, the first qubit of the first register is in the state $|1\rangle$. In this case, Hadamard gates are applied to the last n_1 qubits of the second register, which ends up in a linear combination of $2^{n_1} = N_1$ states whose first qubit is in the state $|0\rangle$. Thus, it corresponds to the linear combination of all the states associated to the nodes in $\mathcal{V}_1$, and the corresponding states $|\psi_i\rangle$ are obtained once again. Therefore this quantum circuit correctly implements the action of the update operator V.

Focusing our attention to the number of $T/T^\dagger$ gates again, we see that in this case the contribution to the count of $T/T^\dagger$ gates comes from the controlled array of H gates. Since there is only a single control qubit, this controlled array corresponds to $n_1 - n_2$ single-controlled-H gates. Each of these gates contains 2 $T/T^\dagger$ gates after rotating the H gates into X gates as we did in the case of the complete graph. Therefore, the total number of $T/T^\dagger$ gates for the update operator V is:

$$L_V = 2(n_1 - n_2). \tag{8.61}$$

Taking into account that the operator D requires $n = n_1 + 1$ qubits, we can substitute in equation (8.32) to obtain:

$$L_U = 14(n_1 + 1) - 35 + 4(n_1 - n_2) = 18n_1 - 4n_2 - 21. \tag{8.62}$$

Finally, we express this equation in terms of the number of nodes:

$$L_U = 18\log_2(N_1) - 4\log_2(N_2) - 21, \tag{8.63}$$

so it scales linearly with the logarithm of the number of nodes. Therefore the homomorphic evaluation of the bipartite graph can be implemented efficiently.

8.4 Homomorphic quantum walk simulation

In this section, we present simulations of a quantum and a semiclassical Szegedy walk with our QHE scheme using the IBM Qiskit simulator. Due to measurements of the Bell registers, the key-updating process is not deterministic. Preceding QHE implementations of Liang's scheme in Qiskit used only Clifford gates in the circuit [128] or required the previous calculation of all the possible mappings between the initial and the final

key [163]. For each $T/T^\dagger$ gate there are 2 resulting bits to consider, which meant that the number of possibilities that had to be precalculated increased exponentially, making the simulation of an arbitrarily large circuit impossible. Furthermore, an exponentially growing number of classically-controlled S gates was required in the simulation. In our case, due to the classical gates applied for the key-updating process presented in Section 8.2.3, the encrypting key is updated during the running time in the Qiskit simulator. This eliminates the necessity to precalculate anything and leads to the running time of the simulation increasing linearly. We first explain how the classical gates were constructed in Qiskit and then we proceed to show the simulations performed.

8.4.1 Classical gates in Qiskit

Qiskit allows the implementation of quantum gates that are controlled by classical bits. However, the application of classical gates on classical bits is not currently allowed by Qiskit. In order to overcome this issue, we use ancilla qubits to emulate these classical gates. These ancilla qubits can be loaded with the value of the classical bits so we can apply the equivalent quantum gate to the desired classical gate, and finally measure them to copy the results back to the bits.

We will start with the classical bit-flip or NOT gate. The classical-quantum circuit that emulates this gate with an ancilla qubit is shown in Figure 8.16. We first apply a reset operation to the ancilla qubit so that it is in the state $|0\rangle$, and then we apply a X gate controlled by the classical bit. Thus, if the bit has the value $a \in \{0, 1\}$, the qubit is left in the state $|a\rangle$. The equivalent quantum gate to the classical NOT gate is the X gate. We then apply this gate to the qubit, and finally measure its value so it is stored in the classical bit. Notice that since the qubit was previously reset to the state $|0\rangle$, it is always either on $|0\rangle$ or $|1\rangle$ after the X gates are applied. Therefore, the measurement is deterministic and the emulation of the classical NOT gate is correct.

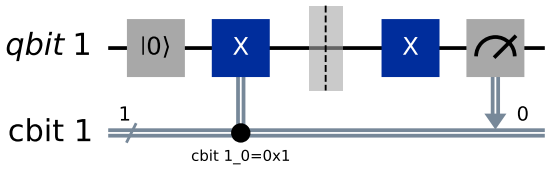
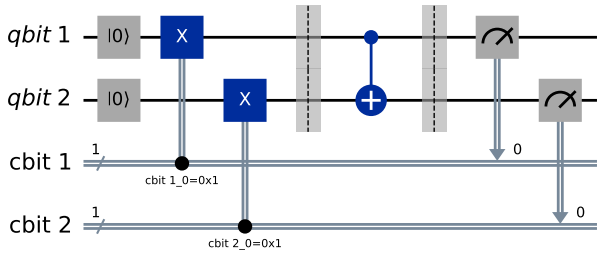


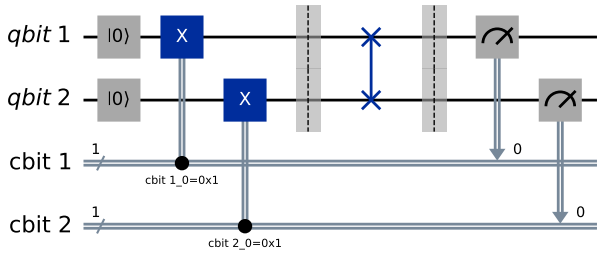
Figure 8.16: Quantum circuit for the emulation of a classical NOT gate using one ancilla qubit.

For classical gates that act on two bits, in this case the classical $CNOT$ gates or the classical swap gates, we require two ancilla qubits. The classical-quantum circuits that emulate these gates are shown in Figure 8.17. Once more, we first reset the ancilla qubits

and then copy the values of the classical bits applying classically-controlled-X gates. Then, we apply the equivalent quantum gate, a $CNOT$ or a swap gate respectively, to the ancilla qubits. Finally we measure the qubits and store the results in the classical bits.



(a)



(b)

Figure 8.17: a) Quantum circuit for the emulation of a classical $CNOT$ gate using two ancilla qubits. b) Quantum circuit for the emulation of a classical swap gate using two ancilla qubits.

Another classical gate we are interested in is the reset gate. It erases the value of a classical bit and turns it back to a 0. In this case, we first apply a reset operation to an ancilla qubit so the state $|0\rangle$ is obtained and then we measure it, storing the result in the classical bit. The classical-quantum emulation circuit is shown in Figure 8.18.

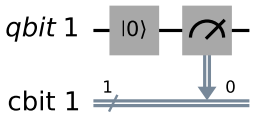


Figure 8.18: Quantum circuit for the emulation of a classical reset using an ancilla qubit.

Lastly, we also want to be able to initialize the classical bits at 0 or 1 randomly in order to generate an initial secret key at running time. In this case, we have that the quantum analogue gate is the H gate. Even though this gate is purely quantum and the qubit ends up in a coherent superposition after applying it, if we measure the qubit the classical bit obtained is either 0 or 1 at random with a probability of 50% for each value. The classical-quantum emulation circuit is shown in Figure 8.19.

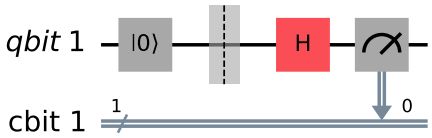


Figure 8.19: Quantum circuit for the emulation of a classical random initialization using an ancilla qubit. The measurement result serves as one bit of the secret key.

Since the ancilla qubits have to be reset before each classical gate emulation and these ancilla qubits do not play any role after the measurement, they can be reused for all the classical gates in the circuit. This means that we need only two ancilla qubits in total. Moreover, the addition of these qubits in the simulation can even be avoided. In our QHE scheme the client measures the quantum state in the computational basis and then proceeds to decrypt the classical result. This measurement can be performed before all the classical gates required for the key-updating process. Since the main qubits of the circuit do not play any role after measurement, we can reset them and make use of two of them as the ancilla qubits for the classical gates. Thus, we save memory and time resources by a factor of four, since there is no need to add two new qubits to the system. Naturally, this simplification can only be performed if the main circuit contains more than a single qubit, which is the most common situation.

8.4.2 Simulation results in Qiskit

The first simulation we have performed is a time step of the Szegedy quantum walk on the bipartite graph with $N_1 = N_2 = 4$, so that the graph contains $N = 8$ nodes in total. We select this case because for $N_1 = N_2$ the quantum circuit for the update operator V does not contain any $T/T^\dagger$ gate, as it can be seen from equation (8.61). Since the circuit for this graph requires $n = 3$ qubits per register, the only gate that contains $T/T^\dagger$ gates is the Toffoli gate of the diagonal operator D, so that the total number of $T/T^\dagger$ gates is $L = 7$. This number is the minimum non-null number of $T/T^\dagger$ gates that we can have in this kind of circuits, which means that this circuit is the easiest one that can be simulated. The decomposition of the algorithm in terms of

Clifford+T gates can be seen in Figure 8.20. We have chosen the initial state of the walk as $|\psi(0)\rangle = \sqrt{0.75}\,|\psi_0\rangle + \sqrt{0.25}\,|\psi_4\rangle$. This state is created first applying a R_Y gate that leaves the first register in the state $\sqrt{0.75}\,|0\rangle_1 + \sqrt{0.25}\,|4\rangle_1$, and then a posterior application of the update operator V. Notice that since the client does not perform the QHE evaluation, the R_Y gate does not have to be decomposed.

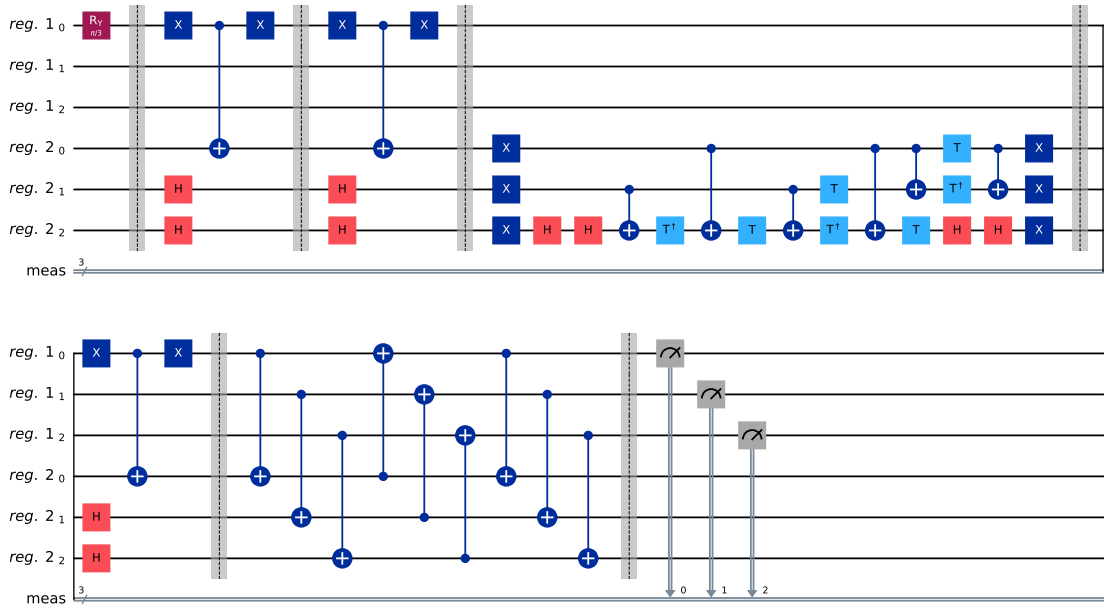


Figure 8.20: Quantum circuit for Szegedy quantum walk over a bipartite graph with $N_1 = N_2 = 4$ nodes decomposed in Clifford+T gates. Notice that although Qiskit uses little-endian ordering, we construct the circuits using big-endian ordering, which means that the resulting bitstring of the simulation must be reversed. The two first blocks of the circuit correspond to the R_Y gate and the V operators of the client. The rest of the circuit is the U_w operator of the server. The last block of U_w is the compilation of the three swap gates into $CNOT$ gates.

For the homomorphic implementation, since there are 7 $T/T^\dagger$ we need 14 ancilla qubits in this case. Thus, the total number of qubits required for the QHE implementation is 20. Using the classical-quantum circuit blocks of Figure 8.1 we have constructed the circuit for the three steps of the QHE scheme described in Section 8.2, in a similar manner as the circuit presented in Figure 8.3. We have concatenated them in a single circuit so that it can be run in the Qiskit simulator. As we mentioned in section 8.4.1, Qiskit can implement classically-controlled gates but it is not currently able to implement classical gates applied to classical bits in a straightforward manner. In order to overcome this issue, we have used two additional ancilla qubits. For each classical gate, making use of classically-controlled-X gates we copy the value of the classical bits into the qubits

which start in the state $|0\rangle$. We then apply the equivalent quantum gate between the qubits, and we measure them to copy the values back to the classical bits. If we reset these qubits after each classical gate, they can be reused for the whole circuit. We also want to remark that even though Qiskit uses little-endian ordering [54], we construct the circuits using big-endian ordering, which means that the resulting bitstring of the simulation must be reversed.

In Table 8.3 we present the results of seven runs of the simulation. As it can be observed, each initial key is updated into the final key once the algorithm is finished. Moreover, the last two rows are highlighted in blue to show how even if the initial keys are the same, we obtain two different final keys that depend on the measurement results of the Bell registers. Thus, the final key is not obtained deterministically. In this particular circuit, the position of the walker is obtained by measuring only the first register, so the classical results are bitstrings with only 3 bits. Once the final key is calculated, the decrypted result can be obtained applying the element-wise XOR to the first three bits of the final X key. As an example, the first encrypted result obtained, 101, is decrypted using the corresponding final key 010, so we have $101 \oplus 010 = 111$ and the final decrypted result is obtained.

Initial key		S^a-rotated Bell measurements	Final key		Result	
X	Z		X	Z	Encrypted	Decrypted
000|010	011|100	10|11|01|10|11|00|00	010|000	011|111	101	111
001|000	011|110	00|01|10|11|11|00|01	010|001	010|111	110	100
011|001	011|000	00|00|11|10|01|10|00	100|011	101|111	000	100
000|001	001|101	01|00|00|00|11|00|01	001|000	010|101	101	100
011|010	010|101	00|11|00|11|00|01|10	000|011	010|110	101	101
011|010	111|011	11|01|00|10|10|10|01	111|011	001|111	011	100
011|010	111|011	00|10|10|01|11|00|00	010|011	010|111	110	100

Table 8.3: Results for seven repetitions of the simulation of the QHE scheme applied to the quantum walk on the bipartite graph. We show the initial keys that were generated, the results obtained from measuring the Bell registers, the updated final keys, and the results of measuring the qubits of the first register. These last results are decrypted according to the first three bits of the final key associated to X gates. The last two rows marked in blue show that the final key is obtained stochastically even if the initial key is the same.

If we simulate the algorithm for a larger amount of repetitions, we can sample the probability distribution of the walker. For this purpose, we have run the circuit for 20000 repetitions. The results are shown in Figure 8.21. Before decrypting, an homogeneous distribution is obtained. The reason for this is due not only to the randomness present in the initial key generation, but also the randomness of the S^a-rotated Bell measurements. In order to evaluate the correctness of the results obtained after decrypting, we have simulated the walk algorithm deterministically making use of the python library SQUWALS. It provides an efficient classical simulator for Szegedy quantum walks [211].

As it can be seen from Figure 8.21, the decrypted distribution and the simulated distribution are similar, and actually their differences are caused due to the stochasticity of the Qiskit simulator for a finite number of repetitions. Therefore, the homomorphic simulation of the quantum walk on the bipartite graph is correct.

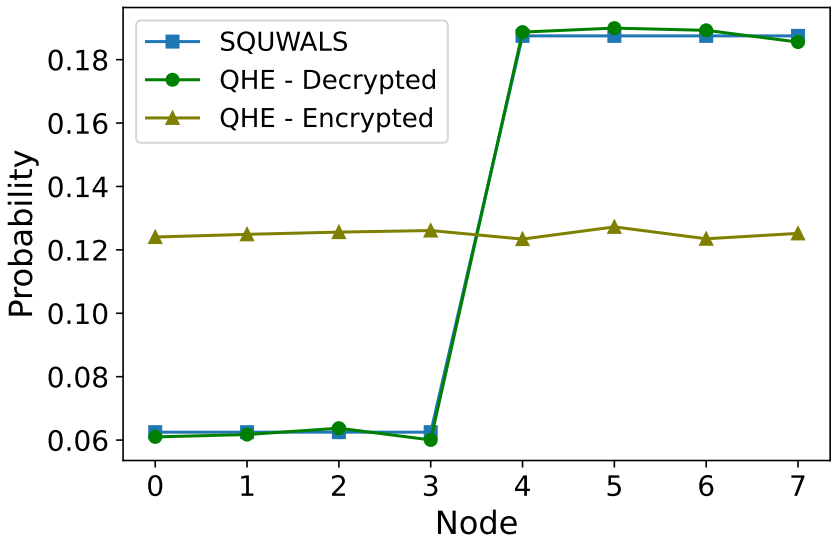
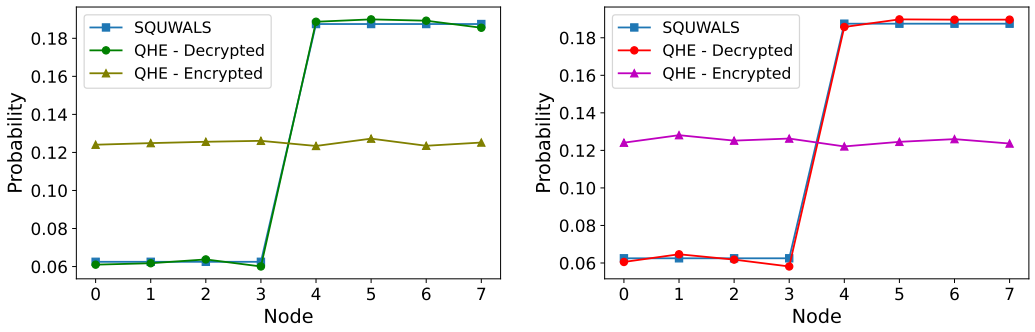


Figure 8.21: Probability distributions of the walker for the quantum walk on the bipartite graph using the QHE scheme and 20000 samples, before and after decrypting. The results are compared with the deterministic probability distribution obtained making use of the SQUWALS library [211].

Besides this simulation, we have also simulated a semiclassical walk on a cycle graph with $N = 8$ nodes. In this case the quantum circuit for the unitary evolution operator U_w has $n = 3$ qubits per register. Then, substituting in equation (8.37) we see that it has $L_U = 77$ $T/T^\dagger$ gates. This means that 154 ancilla qubits would be required for the QHE scheme implementation of a just single quantum step. For a classical simulator like Qiskit, whose memory requirements increase exponentially with the number of qubits, a simulation of this magnitude is impossible to achieve. In order to overcome this issue, we use a simplification of the simulation which requires only 2 ancilla qubits for all the $T/T^\dagger$ gates contained in a circuit, based on the simulation implemented previously in our Grover search publication [163]. Due to the principle of deferred measurement, it was established that the client can wait to measure the Bell registers after the server has finished its operations. Nevertheless, for the sake of simulation, the client could perform key-updating in parallel with the circuit of the server, and measure the Bell register just after the server evaluates a $T/T^\dagger$ gate. To measure in the correct Bell basis, key-updating is performed to know the value of the classical bit a corresponding to that $T/T^\dagger$ gate, stored in the classical bit x associated with the qubit at that instant of the simulation. Then, for each gate contained in the quantum algorithm, we have

to implement its quantum evaluation scheme circuit in parallel with the corresponding classical-quantum circuit required for key-updating.

After measuring a quantum Bell register, its qubits no longer play a role in the simulation. Then, we can reset the Bell qubits once a measurement is completed, and reuse them for all the remaining $T/T^\dagger$ gates of the circuit. Thus, we can represent all the Bell registers that would be required in a real implementation of the QHE scheme using just two ancilla qubits. Therefore, the memory requirements are independent of the number of $T/T^\dagger$ gates, and the scheme can be simulated for any circuit provided that the number of qubits of the original algorithm allows it. Notice that for this simulation procedure the main qubits are measured at the end of the simulation. This means that we cannot make use of two of them as ancilla qubits for the classical gates emulation. Nevertheless, as the quantum Bell register is reset after each measurement, its two qubits can also be used for all the classical gates. Thus, adding two additional qubits to the system can also be avoided in the case of the simplified simulation.



(a) (b)

Figure 8.22: Probability distributions of the walker for the quantum walk on the bipartite graph using the QHE scheme and 20000 samples, making use of a) the realistic simulation and b) the simplified simulation. The results are compared with the deterministic probability distribution obtained using SQUWALS.

In order to verify that the simplified simulation provides the same results than the simulation with all the ancilla qubits, we compare the results obtained using both simulation methods for the previous bipartite graph in Figure 8.22. We can see that the results are similar in both cases, and their differences can be attributed once again to the fact that we are sampling the probability distributions with a finite number of repetitions.

We also verified that the simplification works for a circuit that contains measurement and reset operations. For this purpose we created an artificial circuit, shown in Figure 8.23.

The results obtained from simulating this example circuit are shown in Figure 8.24. The homogeneous distribution is obtained from both QHE simulations before decrypting,

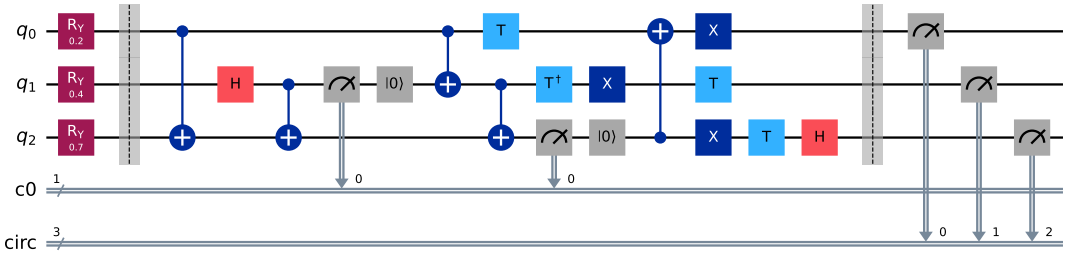


Figure 8.23: Example circuit with measurement and reset operations. The part of the client corresponds to the R_Y gates located before the first grey barrier.

and once the results are decrypted they are the same as the original unencrypted circuit.

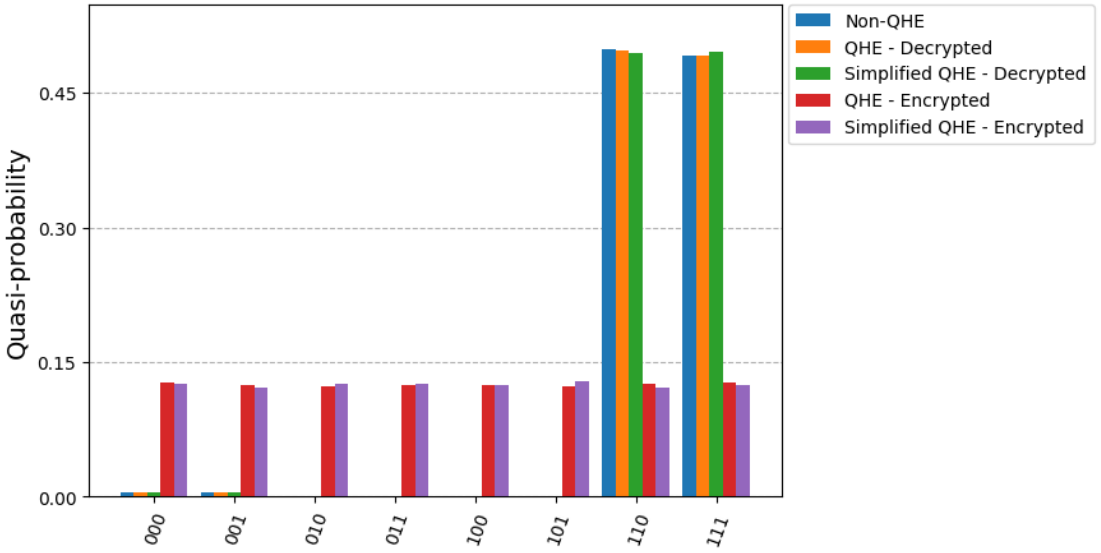


Figure 8.24: Results of the QHE scheme simulation of the quantum circuit in Figure 8.23 using the realistic and the simplified simulation. The results are also compared with the simulation of the circuit without applying the QHE scheme.

Regarding the semiclassical walk on the cycle graph, we selected a quantum time $t_q = 2$. Then, the semiclassical walk is equivalent to a classical walk over two independent square graphs [204]. Its representation is shown in Figure 8.25(a). For the initial classical probability vector we chose $(0.75, 0.25, 0, 0, 0, 0, 0, 0)^T$, whose corresponding quantum state is created by the client with a R_Y gate. We performed the simulation of the semiclassical walk for $t_c = 10$ classical steps. Like we did for the simulation of the bipartite graph, we have also performed a deterministic simulation using the SQUWALS library [211]. The results of this simulation are shown in Figure 8.25(b), where we show

the probability that the walker has of being at each node for each classical step. The walker alternates between odd and even nodes at each time step, since both squares of the graph have an even number of nodes. Furthermore, as both squares of the graph are not connected, the total probability inside them is conserved. The results of the QHE scheme implementation using Qiskit for 20000 repetitions are presented in Figure 8.25(c) before decrypting, and in Figure 8.25(d) after decrypting. As expected, an homogeneous distribution is obtained before decrypting, whereas after the decryption process is completed we obtain similar results to the deterministic simulation. This means that the QHE scheme simulation is successful for the semiclassical walks too, which contain reset and measurement operations. The fact that the QHE scheme functions for both types of walks demonstrates the correctness of the scheme.

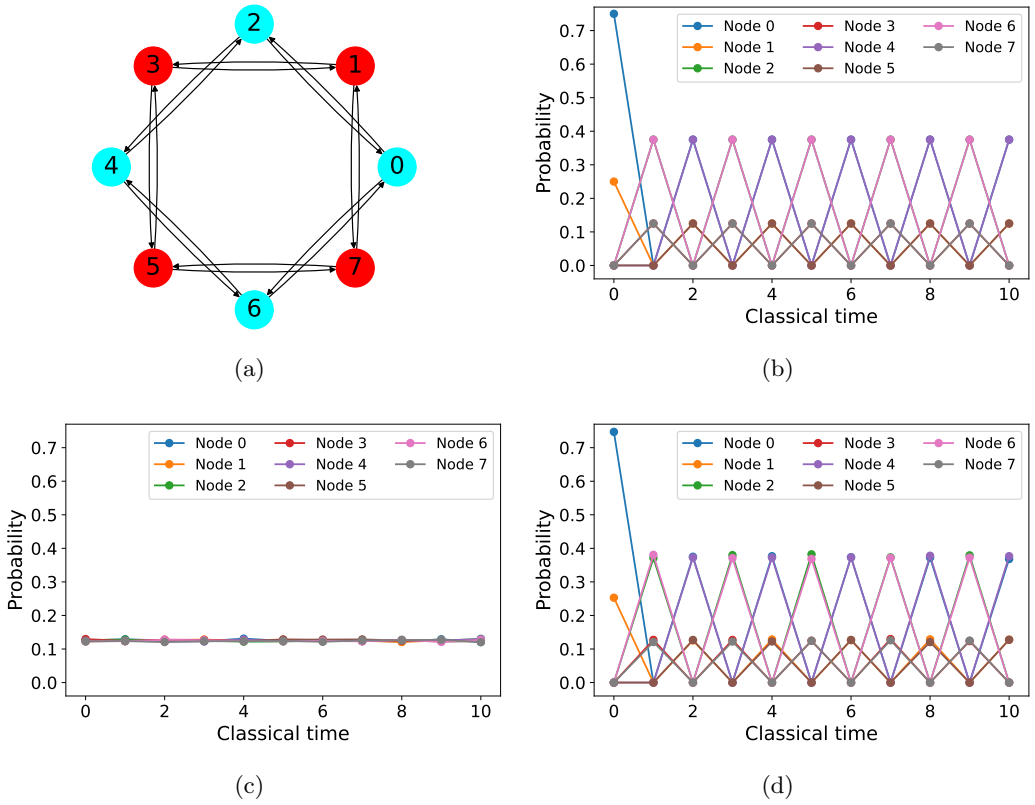


Figure 8.25: Semiclassical graph for the semiclassical walk on the cycle with $N = 8$ nodes for $t_q = 2$. The graph has been plotted using the python library NetworkX [215]. b) Probability of the walker being at each of the eight nodes at each classical time step of the semiclassical walk in a) obtained deterministically using the SQUWALS library [211]. c)-d) Probability of the walker being at each of the eight nodes at each classical time step of the semiclassical walk in a) sampled with 20000 repetitions of the QHE scheme simulation, before decrypting and after decrypting respectively.

We want to remark that these simulations constitute the first simulations of Liang's scheme in which the keys are updated as the simulation is being run and also where the number of classically controlled S gates grows linearly. Moreover, the Python library CQC-QHE has been built from these simulations. It is available on GitHub [217]. There is a guide about how the different functions work, and tutorials implementing the quantum and semiclassical walks shown in this chapter. At the present time, the user must provide the circuit that he desires to evaluate decomposed into Clifford+T gates, or with multi-controlled-X gates that activate for the state $|1\rangle$. A possible path to expand the library could be including the automatic decomposition of more quantum gates, like R_Y gates.

This library represents an important contribution to the field of QHE simulation, since it can be applied to a plethora of quantum circuits in a direct manner. Therefore using this library, QHE simulations of many existing quantum algorithms can be tested easily and this can represent an important leap in the field.

8.5 Conclusions

- In this chapter, Liang's quasi-compact QHE scheme [127] was reinterpreted in terms of classical-quantum circuits. The resulting scheme is still perfectly secure, non-interactive, $\mathcal{F}$-homomorphic and quasi-compact. The complexity of the decryption procedure still depends on the number of $T/T^\dagger$ gates, L. Moreover, we have proved that it can also be applied to semiclassical circuits, making use of the key-updating rules for the homomorphic evaluations of the intermediate measurement and reset operations. These rules were presented in table 8.2.

- We have studied the efficiency of this QHE scheme applied to both quantum and semiclassical walks based on Szegedy's quantization of certain types of networks: cyclic graphs, complete graphs and bipartite graphs. The value of L for the different graphs analysed grows slower than any linear function. In particular, L scales as $O(\log_2^3(N))$ for the cyclic graphs and the complete graphs, and it scales as $O(\log_2(N))$ for the bipartite graphs. The scheme is efficient only for circuits with a polynomial number of $T/T^\dagger$ gates. Therefore, the Szegedy quantum walks analysed are perfect illustrations of algorithms that can be evaluated homomorphically with non-interaction and perfect security in an efficient manner.

- Making use of this QHE scheme in Qiskit, first we have simulated a quantum circuit that implements the walk on a bipartite graph homomorphically to verify the correct functioning of the scheme. We showed how to apply classical-quantum gates in the Qiskit simulation in order to correct the phase errors and update the keys as the gates were applied. The python library SQUWALS was used to simulate the walk algorithm deterministically in order to verify that the obtained results are correct. The correct results can always be successfully decrypted, independently

of the initial key used in encryption. This fact is demonstrated by table 8.3 and figure 8.21. We have also simulated a quite more complex circuit for a semiclassical walk on a cycle graph, which contains intermediate measurement and reset operations. Once more, the walk was simulated deterministically using the SQUWALS library. This deterministic simulation was employed to verify the correctness of the decrypted results of the QHE simulation. These positive outcomes, which can be seen in figure 8.25, demonstrate the correct functioning of the QHE protocol, even if there are non-unitary operations in the circuit.

- These results represent an improvement compared to previous simulations of the QHE scheme, since those were restricted to just Clifford gates or had an exponential number of classically controlled S gates, requiring the calculation of the final keys beforehand. Our simulations use classical gates to simulate the key-updating functions at running time. This is performed for all gates in the circuit, so the decryption procedure scales linearly with the total number of gates instead of L. The reason for this is that Qiskit does not allow the application of arbitrary functions to classical bits. Still, we believe this method of QHE simulation is successful as a proof of concept. On the other hand, the compilation of the Clifford composite functions in XOR gates could serve as future work.

- Using our results we have created a Python library based on Qiskit, named CQC-QHE [217]. It is able to construct and classically simulate the classical-quantum circuits required for quantum homomorphic encryption. Currently, the user must provide the circuit that he wants to evaluate compiled into Clifford+T gates, or with multi-controlled-X gates that activate for the state $|1\rangle$. The automatic decomposition of circuits that contain other types of gates, such as R_Y gates or controlled-H, is a research line left as future work. Since this is an open access library and its application for plenty of quantum circuits is direct, we expect that it will be applied by different researchers interested in QHE simulation, further expanding this field of research.

- More optimizations can be applied regarding the decomposition of controlled gates, such as decomposing the multi-controlled-X gate applying the relative phase Toffoli gate [166] or the recent multi-controlled quantum gates implementations proposed in [218]. These implementations would decrease the number of $T/T^\dagger$ gates and ancilla qubits required. In any case, since the implementations we have constructed are already efficient for the Szegedy graphs studied, the study of further optimizations is left as future work.

Bibliography

[1] R. P. Feynman, "Simulating physics with computers," *International Journal of Theoretical Physics*, vol. 21, pp. 467–488, 1982.

[2] D. Deutsch, "Quantum theory, the Church–Turing principle and the universal quantum computer," *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, vol. 400, pp. 97 – 117, 1985.

[3] A. M. Turing, "On Computable Numbers, with an Application to the Entscheidungsproblem," *Proceedings of the London Mathematical Society*, vol. s2-42, no. 1, pp. 230–265, 1937.

[4] E. Bernstein and U. Vazirani, "Quantum complexity theory," in *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, pp. 11–20, 1993.

[5] D. R. Simon, "On the Power of Quantum Computation," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1474–1483, 1997.

[6] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pp. 124–134, 1994.

[7] W. G. Unruh, "Maintaining coherence in quantum computers," *Phys. Rev. A*, vol. 51, pp. 992–997, 1995.

[8] S. Haroche and J. M. Raimond, "Quantum Computing: Dream or Nightmare?," *Physics Today*, vol. 49, no. 8, pp. 51–52, 1996.

[9] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Phys. Rev. A*, vol. 52, pp. R2493–R2496, 1995.

[10] A. M. Steane, "Error Correcting Codes in Quantum Theory," *Phys. Rev. Lett.*, vol. 77, pp. 793–797, 1996.

[11] P. Shor, "Fault-tolerant quantum computation," in *Proceedings of 37th Conference on Foundations of Computer Science*, pp. 56–65, 1996.

[12] D. Aharonov and M. Ben-Or, "Fault-tolerant quantum computation with constant error," in *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of*

Computing, STOC '97, pp. 176–188, 1997.

[13] A. Y. Kitaev, "Quantum computations: algorithms and error correction," *Russian Mathematical Surveys*, vol. 52, pp. 1191–1249, 1997.

[14] E. Knill, R. Laflamme, and W. H. Zurek, "Resilient Quantum Computation," *Science*, vol. 279, no. 5349, pp. 342–345, 1998.

[15] J. Preskill, "Reliable quantum computers," *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 454, no. 1969, pp. 385–410, 1998.

[16] J. I. Cirac and P. Zoller, "Quantum Computations with Cold Trapped Ions," *Phys. Rev. Lett.*, vol. 74, pp. 4091–4094, 1995.

[17] R. Campos, *Hybrid Quantum-Classical Algorithms*. PhD thesis, Universidad Complutense de Madrid, 2024. `arXiv:2406.12371`.

[18] J. Preskill, "Quantum computing 40 years later," *arXiv preprint arXiv:2106.10522*, 2021.

[19] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell, *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[20] J. Preskill, "Quantum computing and the entanglement frontier," *arXiv preprint arXiv:1203.5813*, 2012.

[21] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, p. 79, 2018.

[22] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, "A variational eigenvalue solver on a photonic quantum processor," *Nature Communications*, vol. 5, p. 4213, 2014.

[23] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature*, vol. 549, pp. 242–246, 2017.

[24] S. Endo, Z. Cai, S. C. Benjamin, and X. Yuan, "Hybrid quantum-classical algorithms and quantum error mitigation," *Journal of the Physical Society of Japan*, vol. 90, no. 3, p. 032001, 2021.

[25] D. Jaksch, C. Bruder, J. I. Cirac, C. W. Gardiner, and P. Zoller, "Cold Bosonic Atoms in Optical Lattices," *Phys. Rev. Lett.*, vol. 81, pp. 3108–3111, Oct 1998.

[26] J. Zhang, G. Pagano, P. W. Hess, A. Kyprianidis, P. Becker, H. Kaplan, A. V. Gorshkov, Z.-X. Gong, and C. Monroe, "Observation of a many-body dynamical phase transition with a 53-qubit quantum simulator," *Nature*, vol. 551, pp. 601–604, 2017.

[27] H. Bernien, S. Schwartz, A. Keesling, H. Levine, A. Omran, H. Pichler, S. Choi, A. S. Zibrov, M. Endres, M. Greiner, V. Vuletić, and M. D. Lukin, "Probing many-body dynamics on a 51-atom quantum simulator," *Nature*, vol. 551, pp. 579–584, 2017.

[28] G. Semeghini, H. Levine, A. Keesling, S. Ebadi, T. T. Wang, D. Bluvstein, R. Verresen, H. Pichler, M. Kalinowski, R. Samajdar, A. Omran, S. Sachdev, A. Vishwanath, M. Greiner, V. Vuletić, and M. D. Lukin, "Probing topological spin liquids on a programmable quantum simulator," *Science*, vol. 374, no. 6572, pp. 1242–1247, 2021.

[29] S. S. Gill, A. Kumar, H. Singh, M. Singh, K. Kaur, M. Usman, and R. Buyya, "Quantum computing: A taxonomy, systematic review and future directions," *Software: Practice and Experience*, vol. 52, no. 1, pp. 66–114, 2022.

[30] J. L. Park, "The concept of transition in quantum mechanics," *Foundations of Physics*, vol. 1, pp. 23–33, 1970.

[31] Qureca, "Overview on Quantum Initiatives Worldwide-Update Mid-2021." `https://www.qureca.com/overview-on-quantum-initiatives-worldwideupdate-mid-2021/`, 2021. [Online; accessed March 2025].

[32] J. Chow, O. Dial, and J. Gambetta, "IBM Quantum Breaks the 100-Qubit Processor Barrier." `https://research.ibm.com/blog/127-qubit-quantumprocessor-eagle`, 2021. [Online; accessed March 2025].

[33] Z. Yang, M. Zolanvari, and R. Jain, "A survey of important issues in quantum computing and communications," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 1059–1094, 2023.

[34] M. H. Devoret, A. Wallraff, and J. M. Martinis, "Superconducting Qubits: A Short Review," *arXiv preprint arXiv:cond-mat/0411174*, 2004.

[35] D. DiVincenzo, "The Physical Implementation of Quantum Computation," *Fortschritte der Physik*, vol. 48, pp. 771–783, 2000.

[36] M. H. Devoret and R. J. Schoelkopf, "Superconducting Circuits for Quantum Information: An Outlook," *Science*, vol. 339, no. 6124, pp. 1169–1174, 2013.

[37] D. Castelvecchi, "Quantum computers ready to leap out of the lab in 2017," *Nature*, vol. 541, pp. 9–10, 01 2017.

[38] G. Wendin, "Quantum information processing with superconducting circuits: a review," *Reports on Progress in Physics*, vol. 80, no. 10, p. 106001, 2017.

[39] J. M. Gambetta, J. M. Chow, and M. Steffen, "Building logical qubits in a superconducting quantum computing system," *npj Quantum Information*, vol. 3, p. 2, 2017.

[40] J. J. García-Ripoll, E. Solano, and M. A. Martin-Delgado, "Quantum simulation of Anderson and Kondo lattices with superconducting qubits," *Phys. Rev. B*, vol. 77, p. 024522, 2008.

[41] F. Jazaeri, A. Beckers, A. Tajalli, and J.-M. Sallese, "A Review on Quantum Computing: From Qubits to Front-end Electronics and Cryogenic MOSFET Physics," in *2019 MIXDES - 26th International Conference "Mixed Design of Integrated Circuits and Systems"*, pp. 15–25, 2019.

[42] O. Viyuela, A. Rivas, S. Gasparinetti, A. Wallraff, S. Filipp, and M. A. Martin-Delgado, "Observation of topological Uhlmann phases with superconducting qubits," *npj Quantum Information*, vol. 4, p. 10, 2018.

[43] G. García-Pérez, M. A. C. Rossi, and S. Maniscalco, "IBM Q Experience as a versatile experimental testbed for simulating open quantum systems," *npj Quantum Information*, vol. 6, p. 1, 2020.

[44] M. Amico, Z. H. Saleem, and M. Kumph, "Experimental study of Shor's factoring algorithm using the IBM Q Experience," *Phys. Rev. A*, vol. 100, p. 012305, 2019.

[45] K. Das and A. Sadhu, "Experimental study on the quantum search algorithm over structured datasets using IBMQ experience," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 8, Part B, pp. 6441–6452, 2022.

[46] S. J. Devitt, W. J. Munro, and K. Nemoto, "Quantum error correction for beginners," *Reports on Progress in Physics*, vol. 76, no. 7, p. 076001, 2013.

[47] A. Kitaev, "Fault-tolerant quantum computation by anyons," *Annals of Physics*, vol. 303, no. 1, pp. 2–30, 2003.

[48] H. Bombin and M. A. Martin-Delgado, "Topological Quantum Distillation," *Phys. Rev. Lett.*, vol. 97, p. 180501, 2006.

[49] M. Kleinmann, H. Kampermann, T. Meyer, and D. Bruß, "Physical purification of quantum states," *Phys. Rev. A*, vol. 73, p. 062309, 2006.

[50] L. Vandersypen and A. van Leeuwenhoek, "1.4 Quantum computing - the next challenge in circuit and system design," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 24–29, 2017.

[51] M. Y. Lanzerotti, G. Fiorenza, and R. A. Rand, "Microminiature packaging and integrated circuitry: The work of E. F. Rent, with an application to on-chip interconnection requirements," *IBM Journal of Research and Development*, vol. 49, no. 4.5, pp. 777–803, 2005.

[52] J. Levy, "1 million qubit quantum computers: moving beyond the current 'brute force' Strategy.." `https://seeqc.com/blog/1-million-qubit-quantum-computers-moving-beyond-the-current-brute-force-strategy`, 2022. [Online; accessed April 2025].

[53] A. Singh, K. Dev, H. Siljak, H. D. Joshi, and M. Magarini, "Quantum Internet—Applications, Functionalities, Enabling Technologies, Challenges, and Research Directions," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2218–2247, 2021.

[54] A. Javadi-Abhari et al, "Quantum computing with Qiskit," *arXiv:2405.08810*, 2024.

[55] Google, "Cirq—an open source framework for programming quantum computers." `https://quantumai.google/cirq`. [Online; accessed April 2025].

[56] Y. Huang and M. Martonosi, "Statistical assertions for validating patterns and finding bugs in quantum programs," in *Proceedings of the 46th International Symposium on Computer Architecture*, ISCA '19, (New York, NY, USA), p. 541–553, 2019.

[57] S. R. Hasan, M. Z. Chowdhury, M. Saiam, and Y. M. Jang, "Quantum Communication Systems: Vision, Protocols, Applications, and Challenges," *IEEE Access*, vol. 11, pp. 15855–15877, 2023.

[58] D. Lancho, J. Martinez, D. Elkouss, M. Soto, and V. Martin, "QKD in Standard Optical Telecommunications Networks," in *Quantum Communication and Quantum Networking*, (Berlin, Heidelberg), pp. 142–149, Springer Berlin Heidelberg, 2010.

[59] J. M. Sáez, A. P. Perales, R. C. Palancar, D. R. Lopez, J. F. Chavarria, V. M. Ayuso, and J. P. Brito Mendez, "Current Status, Gaps, and Future Directions in Quantum Key Distribution Standards: Implications for Industry," in *2024 International Conference on Quantum Communications, Networking, and Computing (QCNC)*, pp. 341–345, 2024.

[60] M. I. García Cid, L. Ortiz Martín, and V. Martín Ayuso, "Madrid Quantum Network: A First Step to Quantum Internet," in *Proceedings of the 16th International Conference on Availability, Reliability and Security*, ARES '21, p. 102, 2021.

[61] L. Velasco, M. Ahmadian, L. Ortiz, J. P. Brito, A. Pastor, J. M. Rivas, S. Barzegar,

J. Comellas, V. Martin, and M. Ruiz, "Scenarios for Optical Encryption Using Quantum Keys," *Sensors*, vol. 24, no. 20, p. 6631, 2024.

[62] H. H. Brunner, C.-H. F. Fung, M. Peev, R. B. Méndez, L. Ortiz, J. P. Brito, V. Martín, J. M. Rivas-Moscoso, F. Jiménez, A. A. Pastor, and D. R. López, "Demonstration of a switched CV-QKD network," *EPJ Quantum Technology*, vol. 10, p. 38, 2023.

[63] M. Garcia-Cid, L. Ortiz, J. Saez, and V. Martin, "Strategies for the Integration of quantum networks for a future quantum internet," *arXiv preprint arXiv:2401.06444*, 2023.

[64] V. Scarani, H. Bechmann-Pasquinucci, N. J. Cerf, M. Dušek, N. Lütkenhaus, and M. Peev, "The security of practical quantum key distribution," *Rev. Mod. Phys.*, vol. 81, pp. 1301–1350, 2009.

[65] H.-K. Lo and H. F. Chau, "Unconditional Security of Quantum Key Distribution over Arbitrarily Long Distances," *Science*, vol. 283, no. 5410, pp. 2050–2056, 1999.

[66] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," in *Proceedings of IEEE International Conference on Computers, Systems, and Signal Processing*, pp. 175–179, 1984.

[67] C. H. Bennett and G. Brassard, "Experimental quantum cryptography: the dawn of a new era for quantum cryptography: the experimental prototype is working]," *SIGACT News*, vol. 20, no. 4, p. 78–80, 1989.

[68] G. Brassard and L. Salvail, "Secret-Key Reconciliation by Public Discussion," in *Advances in Cryptology — EUROCRYPT '93*, (Berlin, Heidelberg), pp. 410–423, Springer Berlin Heidelberg, 1994.

[69] C. H. Bennett, G. Brassard, and J.-M. Robert, "Privacy Amplification by Public Discussion," *SIAM Journal on Computing*, vol. 17, no. 2, pp. 210–229, 1988.

[70] V. Martin, J. Martinez-Mateo, and M. Peev, *Introduction to Quantum Key Distribution*, pp. 1–17. John Wiley & Sons, Ltd, 2017.

[71] C. Elliott, "Building the quantum network*," *New Journal of Physics*, vol. 4, p. 46, jul 2002.

[72] M. Peev, C. Pacher, R. Alléaume, C. Barreiro, J. Bouda, W. Boxleitner, T. Debuisschert, E. Diamanti, M. Dianati, J. F. Dynes, S. Fasel, S. Fossier, M. Fürst, J.-D. Gautier, O. Gay, N. Gisin, P. Grangier, A. Happe, Y. Hasani, M. Hentschel, H. Hübel, G. Humer, T. Länger, M. Legré, R. Lieger, J. Lodewyck, T. Lorünser, N. Lütkenhaus, A. Marhold, T. Matyus, O. Maurhart, L. Monat, S. Nauerth, J.-B. Page, A. Poppe, E. Querasser, G. Ribordy, S. Robyr, L. Salvail, A. W. Sharpe,

A. J. Shields, D. Stucki, M. Suda, C. Tamas, T. Themel, R. T. Thew, Y. Thoma, A. Treiber, P. Trinkler, R. Tualle-Brouri, F. Vannel, N. Walenta, H. Weier, H. Weinfurter, I. Wimberger, Z. L. Yuan, H. Zbinden, and A. Zeilinger, "The SECOQC quantum key distribution network in Vienna," *New Journal of Physics*, vol. 11, no. 7, p. 075001, 2009.

[73] M. Sasaki, M. Fujiwara, H. Ishizuka, W. Klaus, K. Wakui, M. Takeoka, S. Miki, T. Yamashita, Z. Wang, A. Tanaka, K. Yoshino, Y. Nambu, S. Takahashi, A. Tajima, A. Tomita, T. Domeki, T. Hasegawa, Y. Sakai, H. Kobayashi, T. Asai, K. Shimizu, T. Tokura, T. Tsurumaru, M. Matsui, T. Honjo, K. Tamaki, H. Takesue, Y. Tokura, J. F. Dynes, A. R. Dixon, A. W. Sharpe, Z. L. Yuan, A. J. Shields, S. Uchikoga, M. Legré, S. Robyr, P. Trinkler, L. Monat, J.-B. Page, G. Ribordy, A. Poppe, A. Allacher, O. Maurhart, T. Länger, M. Peev, and A. Zeilinger, "Field test of quantum key distribution in the Tokyo QKD Network," *Opt. Express*, vol. 19, no. 11, pp. 10387–10409, 2011.

[74] B. Korzh, C. C. W. Lim, R. Houlmann, N. Gisin, M. J. Li, D. Nolan, B. Sanguinetti, and H. Zbinden, "Provably secure and practical quantum key distribution over 307 km of optical fibre," *Nature Photonics*, vol. 9, pp. 163–168, 2015.

[75] V. Martin, A. Aguado, J. P. Brito, A. L. Sanz, P. Salas, D. R. López, V. López, A. Pastor-Perales, A. Poppe, and M. Peev, "Quantum Aware SDN Nodes in the Madrid Quantum Network," in *2019 21st International Conference on Transparent Optical Networks (ICTON)*, pp. 1–4, 2019.

[76] V. Martin, J. P. Brito, L. Ortíz, R. B. Méndez, J. S. Buruaga, R. J. Vicente, A. Sebastián-Lombraña, D. Rincón, F. Pérez, C. Sánchez, M. Peev, H. H. Brunner, F. Fung, A. Poppe, F. Fröwis, A. J. Shields, R. I. Woodward, H. Griesser, S. Roehrich, F. de la Iglesia, C. Abellán, M. Hentschel, J. M. Rivas-Moscoso, A. Pastor-Perales, J. Folgueira, and D. López, "MadQCI: a heterogeneous and scalable SDN-QKD network deployed in production facilities," *npj Quantum Information*, vol. 10, p. 80, 2024.

[77] S. Wang, Z.-Q. Yin, D.-Y. He, W. Chen, R.-Q. Wang, P. Ye, Y. Zhou, G.-J. Fan-Yuan, F.-X. Wang, W. Chen, Y.-G. Zhu, P. V. Morozov, A. V. Divochiy, Z. Zhou, G.-C. Guo, and Z.-F. Han, "Twin-field quantum key distribution over 830-km fibre," *Nature Photonics*, vol. 16, pp. 154–161, 2022.

[78] Y. Liu, W.-J. Zhang, C. Jiang, J.-P. Chen, C. Zhang, W.-X. Pan, D. Ma, H. Dong, J.-M. Xiong, C.-J. Zhang, H. Li, R.-C. Wang, J. Wu, T.-Y. Chen, L. You, X.-B. Wang, Q. Zhang, and J.-W. Pan, "Experimental Twin-Field Quantum Key Distribution over 1000 km Fiber Distance," *Phys. Rev. Lett.*, vol. 130, p. 210801, 2023.

[79] A. García Callejo, A. Llanos, P. Arteaga-Díaz, A. Ruiz-Chamorro, D. Cano, and

V. Fernández, "Implementation model of free-space QKD routing protocols for ground-to-satellite quantum communication links using drones.," *North Atlantic Treaty Organization*, 2023.

[80] A. Garcia-Callejo, A. Ruiz-Chamorro, D. Cano, and V. Fernandez, "A Review on Continuous-Variable Quantum Key Distribution Security," in *Proceedings of the International Conference on Ubiquitous Computing & Ambient Intelligence (UCAmI 2022)* (J. Bravo, S. Ochoa, and J. Favela, eds.), (Cham), pp. 1073–1085, Springer International Publishing, 2023.

[81] A. Ruiz-Chamorro, A. Garcia-Callejo, and V. Fernandez, "Low-complexity continuous-variable quantum key distribution with true local oscillator using pilot-assisted frequency locking," *Scientific Reports*, vol. 14, p. 10770, 2024.

[82] A. Ruiz-Chamorro, D. Cano, A. Garcia-Callejo, and V. Fernandez, "Effects of experimental impairments on the security of continuous-variable quantum key distribution," *Heliyon*, vol. 9, p. e16670, 2023.

[83] Y. Li, W.-Q. Cai, J.-G. Ren, C.-Z. Wang, M. Yang, L. Zhang, H.-Y. Wu, L. Chang, J.-C. Wu, B. Jin, H.-J. Xue, X.-J. Li, H. Liu, G.-W. Yu, X.-Y. Tao, T. Chen, C.-F. Liu, W.-B. Luo, J. Zhou, H.-L. Yong, Y.-H. Li, F.-Z. Li, C. Jiang, H.-Z. Chen, C. Wu, X.-H. Tong, S.-J. Xie, F. Zhou, W.-Y. Liu, Y. Ismail, F. Petruccione, N.-L. Liu, L. Li, F. Xu, Y. Cao, J. Yin, R. Shu, X.-B. Wang, Q. Zhang, J.-Y. Wang, S.-K. Liao, C.-Z. Peng, and J.-W. Pan, "Microsatellite-based real-time quantum key distribution," *Nature*, vol. 640, pp. 47–54, 2025.

[84] A. K. Ekert, "Quantum cryptography based on Bell's theorem," *Phys. Rev. Lett.*, vol. 67, pp. 661–663, 1991.

[85] C. H. Bennett, "Quantum cryptography using any two nonorthogonal states," *Phys. Rev. Lett.*, vol. 68, pp. 3121–3124, 1992.

[86] C. H. Bennett, G. Brassard, and N. D. Mermin, "Quantum cryptography without Bell's theorem," *Phys. Rev. Lett.*, vol. 68, pp. 557–559, 1992.

[87] V. Scarani, A. Acín, G. Ribordy, and N. Gisin, "Quantum Cryptography Protocols Robust against Photon Number Splitting Attacks for Weak Laser Pulse Implementations," *Phys. Rev. Lett.*, vol. 92, p. 057901, 2004.

[88] H.-K. Lo, M. Curty, and B. Qi, "Measurement-Device-Independent Quantum Key Distribution," *Phys. Rev. Lett.*, vol. 108, p. 130503, 2012.

[89] M. Lucamarini, Z. L. Yuan, J. F. Dynes, and A. J. Shields, "Overcoming the rate-distance limit of quantum key distribution without quantum repeaters," *Nature*, vol. 557, pp. 400–403, 2018.

[90] W. Li, L. Zhang, H. Tan, Y. Lu, S.-K. Liao, J. Huang, H. Li, Z. Wang, H.-K. Mao, B. Yan, Q. Li, Y. Liu, Q. Zhang, C.-Z. Peng, L. You, F. Xu, and J.-W. Pan, "High-rate quantum key distribution exceeding 110 Mb s-1," *Nature Photonics*, vol. 17, pp. 416–421, 2023.

[91] E. Diamanti, H.-K. Lo, B. Qi, and Z. Yuan, "Practical challenges in quantum key distribution," *npj Quantum Information*, vol. 2, p. 16025, 2016.

[92] Y. Li, P. Huang, S. Wang, T. Wang, D. Li, and G. Zeng, "A denial-of-service attack on fiber-based continuous-variable quantum key distribution," *Physics Letters A*, vol. 382, no. 45, pp. 3253–3261, 2018.

[93] A. Broadbent and C. Schaffner, "Quantum cryptography beyond quantum key distribution," *Designs, Codes and Cryptography.*, vol. 78, pp. 351–382, 2016.

[94] D. J. Bernstein and T. Lange, "Post-quantum cryptography," *Nature*, vol. 549, pp. 188–194, 2017.

[95] M. Ajtai, "Generating hard instances of lattice problems (extended abstract)," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pp. 99–108, 1996.

[96] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A ring-based public key cryptosystem," in *Algorithmic Number Theory*, pp. 267–288, 1998.

[97] R. J. McEliece, "A Public-Key Cryptosystem based on Algebraic Coding Theory," *Deep Space Network Progress Report 42–44*, 1978.

[98] R. C. Merkle, "A Certified Digital Signature," in *Advances in Cryptology — CRYPTO' 89 Proceedings*, pp. 218–238, 1990.

[99] A. Petzoldt, S. Bulygin, and J. Buchmann, "Selecting Parameters for the Rainbow Signature Scheme," in *Post-Quantum Cryptography*, pp. 218–240, 2010.

[100] A. Broadbent, J. Fitzsimons, and E. Kashefi, "Universal Blind Quantum Computation," in *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pp. 517–526, 2009.

[101] W. Li and D.-L. Deng, "Quantum delegated and federated learning via quantum homomorphic encryption," *Research Directions: Quantum Technologies*, vol. 3, p. e3, 2025.

[102] D. Main, P. Drmota, D. P. Nadlinger, E. M. Ainley, A. Agrawal, B. C. Nichol, R. Srinivas, G. Araneda, and D. M. Lucas, "Distributed quantum computing across an optical network link," *Nature*, vol. 638, pp. 383–388, 2025.

[103] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On Data Banks and Privacy

Homomorphisms," *Foundations of Secure Computation*, pp. 169–179, 1978.

[104] T. V. T. Doan, M.-L. Messai, G. Gavin, and J. Darmont, "A survey on implementations of homomorphic encryption schemes," *The Journal of Supercomputing*, vol. 79, pp. 15098–15139, 2023.

[105] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, p. 120–126, 1978.

[106] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.

[107] P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes," in *Advances in Cryptology — EUROCRYPT '99*, pp. 223–238, 1999.

[108] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-DNF Formulas on Ciphertexts," in *Theory of Cryptography*, (Berlin, Heidelberg), pp. 325–341, 2005.

[109] C. Marcolla, V. Sucasas, M. Manzano, R. Bassoli, F. H. P. Fitzek, and N. Aaraj, "Survey on Fully Homomorphic Encryption, Theory, and Applications," *Proceedings of the IEEE*, vol. 110, no. 10, pp. 1572–1609, 2022.

[110] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A Survey on Homomorphic Encryption Schemes: Theory and Implementation," *ACM Comput. Surv.*, vol. 51, no. 4, 2018.

[111] C. Gentry, *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.

[112] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, (New York, NY, USA), pp. 309–325, Association for Computing Machinery, 2012.

[113] J. Fan and F. Vercauteren, "Somewhat Practical Fully Homomorphic Encryption," *IACR Cryptol. ePrint Arch.*, 2012.

[114] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic Encryption for Arithmetic of Approximate Numbers," in *Advances in Cryptology – ASIACRYPT 2017*, (Cham), pp. 409–437, Springer International Publishing, 2017.

[115] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "TFHE: Fast Fully Homomorphic Encryption Over the Torus," *Journal of Cryptology*, vol. 33, pp. 34–91, 2020.

[116] M. Liang, "Quantum fully homomorphic encryption scheme based on universal quantum circuit," *Quantum Information Processing*, vol. 14, pp. 2749–2759, 2015.

[117] P. P. Rohde, J. F. Fitzsimons, and A. Gilchrist, "Quantum Walks with Encrypted Data," *Phys. Rev. Lett.*, vol. 109, p. 150501, 2012.

[118] J. Zeuner, I. Pitsios, S.-H. Tan, A. N. Sharma, J. F. Fitzsimons, R. Osellame, and P. Walther, "Experimental quantum homomorphic encryption," *npj Quantum Information*, vol. 7, p. 25, 2021.

[119] M. Liang, "Symmetric quantum fully homomorphic encryption with perfect security," *Quantum Information Processing*, vol. 12, pp. 3675–3687, 2013.

[120] S.-H. Tan, J. A. Kettlewell, Y. Ouyang, L. Chen, and J. F. Fitzsimons, "A quantum approach to homomorphic encryption," *Scientific Reports*, vol. 6, p. 33467, 2016.

[121] L. Yu, C. A. Pérez-Delgado, and J. F. Fitzsimons, "Limitations on information-theoretically-secure quantum homomorphic encryption," *Phys. Rev. A*, vol. 90, p. 050303, 2014.

[122] C.-Y. Lai and K.-M. Chung, "On statistically-secure quantum homomorphic encryption," *Quantum Inf. Comput.*, vol. 18, pp. 0785–0794, 2018.

[123] A. Broadbent and S. Jeffery, "Quantum Homomorphic Encryption for Circuits of Low T-gate Complexity," in *Advances in Cryptology – CRYPTO 2015*, (Berlin, Heidelberg), pp. 609–629, Springer Berlin Heidelberg, 2015.

[124] Y. Dulek, C. Schaffner, and F. Speelman, "Quantum Homomorphic Encryption for Polynomial-Sized Circuits," in *Advances in Cryptology – CRYPTO 2016*, (Berlin, Heidelberg), pp. 3–32, Springer Berlin Heidelberg, 2016.

[125] U. Mahadev, "Classical Homomorphic Encryption for Quantum Circuits," in *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 332–338, 2018.

[126] A. Einstein, B. Podolsky, and N. Rosen, "Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?," *Phys. Rev.*, vol. 47, pp. 777–780, 1935.

[127] M. Liang, "Teleportation-based quantum homomorphic encryption scheme with quasi-compactness and perfect security," *Quantum Information Processing*, vol. 19, p. 28, 2020.

[128] C. Gong, J. Du, Z. Dong, Z. Guo, A. Gani, L. Zhao, and H. Qi, "Grover algorithm-based quantum homomorphic encryption ciphertext retrieval scheme in quantum cloud computing," *Quantum Information Processing*, vol. 19, p. 105, 2020.

[129] C. Gong, Z. Dong, A. Gani, and H. Qi, "Quantum Ciphertext Dimension Reduction Scheme for Homomorphic Encrypted Data," in *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp. 903–910, 2021.

[130] Z.-W. Cheng, X.-B. Chen, G. Xu, L. Ma, and Z.-P. Li, "Quantum one-time pad-based quantum homomorphic encryption schemes for circuits of the non-Clifford gates," *Physica A: Statistical Mechanics and its Applications*, vol. 637, p. 129529, 2024.

[131] W. Chang, Z.-Z. Li, F.-C. You, and X.-B. Pan, "Dynamic quantum fully homomorphic encryption scheme based on universal quantum circuit," *Journal of Information Security and Applications*, vol. 75, p. 103510, 2023.

[132] X.-B. Chen, Y.-R. Sun, G. Xu, and Y.-X. Yang, "Quantum homomorphic encryption scheme with flexible number of evaluator based on (k, n)-threshold quantum state sharing," *Information Sciences*, vol. 501, pp. 172–181, 2019.

[133] J. Liu, Q. Li, J. Quan, C. Wang, J. Shi, and H. Situ, "Efficient quantum homomorphic encryption scheme with flexible evaluators and its simulation," *Designs, Codes and Cryptography*, vol. 90, pp. 577–591, 2022.

[134] X.-Q. Cai, Z.-F. Liu, and T. yin Wang, "Measurement-device-independent quantum homomorphic encryption," *Physics Letters A*, vol. 513, p. 129609, 2024.

[135] T. Shang, S. Wang, Y. Jiang, and J. Liu, "Two-round quantum homomorphic encryption scheme based on matrix decomposition," *Quantum Information Processing*, vol. 22, p. 422, 2023.

[136] Z.-W. Cheng, X.-B. Chen, G. Xu, Y. Chang, L.-H. Miao, Y.-X. Yang, and Y.-L. Wang, "A secure quantum homomorphic encryption ciphertext retrieval scheme," *Soft Computing*, vol. 29, pp. 1497–1509, 2025.

[137] I. Sohn, B. Kim, K. Bae, W. Song, and W. Lee, "Error-correctable efficient quantum homomorphic encryption using Calderbank–Shor–Steane codes," *Quantum Information Processing*, vol. 24, p. 28, 2025.

[138] Y. Li, L. Cao, W. Luo, H. Zhang, H. Cai, M. F. Karim, F. Gao, J. Fitzsimons, Q. Song, and A.-Q. Liu, "Experimental Quantum Homomorphic Encryption Using a Quantum Photonic Chip," *Phys. Rev. Lett.*, vol. 132, p. 200801, 2024.

[139] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*. Cambridge university press, 2010.

[140] A. Galindo and M. A. Martín-Delgado, "Information and computation: Classical and quantum aspects," *Rev. Mod. Phys.*, vol. 74, pp. 347–423, 2002.

[141] D. Gottesman and I. L. Chuang, "Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations," *Nature*, vol. 402, pp. 390–393, 1999.

[142] R. Jozsa, "An introduction to measurement based quantum computation," *arXiv preprint arXiv:quant-ph/0508124*, 2005.

[143] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters, "Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels," *Phys. Rev. Lett.*, vol. 70, pp. 1895–1899, 1993.

[144] P. O. Boykin and V. Roychowdhury, "Optimal encryption of quantum bits," *Phys. Rev. A*, vol. 67, p. 042317, 2003.

[145] D. Deutsch and R. Jozsa, "Rapid solution of problems by quantum computation," *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, vol. 439, pp. 553 – 558, 1992.

[146] P. Londero, C. Dorrer, M. Anderson, S. Wallentowitz, K. Banaszek, and I. A. Walmsley, "Efficient optical implementation of the Bernstein-Vazirani algorithm," *Phys. Rev. A*, vol. 69, p. 010302, 2004.

[147] A. Shukla and P. Vedula, "A generalization of Bernstein–Vazirani algorithm with multiple secret keys and a probabilistic oracle," *Quantum Information Processing*, vol. 22, p. 244, 2023.

[148] H. Li and L. Yang, "A quantum algorithm to approximate the linear structures of Boolean functions," *Mathematical Structures in Computer Science*, vol. 28, no. 1, pp. 1–13, 2018.

[149] H. Xie and L. Yang, "Using Bernstein–Vazirani algorithm to attack block ciphers)," *Designs, Codes and Cryptography*, vol. 87, pp. 1161–1182, 2019.

[150] H. Xie and L. Yang, "A quantum related-key attack based on the Bernstein–Vazirani algorithm," *Quantum Information Processing*, vol. 19, p. 240, 2020.

[151] R.-X. Xu, H.-W. Sun, K.-J. Zhang, G. Du, and D.-D. Li, "Quantum differential cryptanalysis based on Bernstein-Vazirani algorithm," *EPJ Quantum Technology*, vol. 11, p. 83, 2024.

[152] B.-M. Zhou and Z. Yuan, "Quantum key-recovery attack on Feistel constructions: Bernstein–Vazirani meet Grover algorithm," *Quantum Information Processing*, vol. 20, p. 330, 2021.

[153] A. W. Cross, G. Smith, and J. A. Smolin, "Quantum learning robust against noise," *Phys. Rev. A*, vol. 92, p. 012327, 2015.

[154] D. Bacon, "The Recursive and Nonrecursive Bernstein-Vazirani Algorithm (Lecture Notes, CSE 599d-Quantum Computing)." `https://courses.cs.washington.edu/courses/cse599d/06wi/lecturenotes7.pdf`, 2006.

[155] P. Fernández and M. A. Martin-Delgado, "Homomorphic encryption of the k=2 Bernstein–Vazirani algorithm," *Journal of Physics A: Mathematical and Theoretical*, vol. 57, no. 36, p. 365301, 2024.

[156] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, (New York, NY, USA), pp. 212–219, Association for Computing Machinery, 1996.

[157] A. Galindo and M. A. Martín-Delgado, "Family of Grover's quantum-searching algorithms," *Phys. Rev. A*, vol. 62, p. 062303, Nov 2000.

[158] G. Brassard, P. HØyer, and A. Tapp, "Quantum counting," in *Automata, Languages and Programming* (K. G. Larsen, S. Skyum, and G. Winskel, eds.), (Berlin, Heidelberg), pp. 820–831, Springer Berlin Heidelberg, 1998.

[159] G. Brassard, P. HØyer, and A. Tapp, "Quantum cryptanalysis of hash and claw-free functions," in *LATIN'98: Theoretical Informatics*, (Berlin, Heidelberg), pp. 163–169, Springer Berlin Heidelberg, 1998.

[160] A. R. Brown, "Playing Pool with $|\psi\rangle$: from Bouncing Billiards to Quantum Search," *Quantum*, vol. 4, p. 357, nov 2020.

[161] A. Yamamura and H. Ishizuka, "Quantum cryptanalysis of block ciphers," *Algebraic Systems, Formal Languages and Computations*, vol. 1166, pp. 235–243, 2000.

[162] M. Boyer, G. Brassard, P. Høyer, and A. Tapp, "Tight Bounds on Quantum Searching," *Fortschritte der Physik*, vol. 46, pp. 493–505, 1998.

[163] P. Fernández and M. A. Martin-Delgado, "Implementing the Grover algorithm in homomorphic encryption schemes," *Phys. Rev. Res.*, vol. 6, p. 043109, 2024.

[164] S. Arunachalam and R. de Wolf, "Optimizing the number of gates in quantum search," *Quantum Inf. Comput.*, vol. 17, no. 3–4, pp. 0251–0261, 2017.

[165] M. Briański, J. Gwinner, V. Hlembotskyi, W. Jarnicki, S. Pliś, and A. Szady, "Introducing structure to expedite quantum searching," *Phys. Rev. A*, vol. 103, p. 062425, 2021.

[166] D. Maslov, "Advantages of using relative-phase Toffoli gates with an application to multiple control Toffoli optimization," *Phys. Rev. A*, vol. 93, p. 022311, 2016.

[167] Z. Jiang, E. G. Rieffel, and Z. Wang, "Near-optimal quantum circuit for Grover's

unstructured search using a transverse field," *Phys. Rev. A*, vol. 95, p. 062317, 2017.

[168] L. K. Grover and J. Radhakrishnan, "Is partial quantum search of a database any easier?," in *Proceedings of the Seventeenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '05, (New York, NY, USA), pp. 186–194, Association for Computing Machinery, 2005.

[169] V. E. Korepin and L. K. Grover, "Simple Algorithm for Partial Quantum Search," *Quantum Information Processing*, vol. 5, pp. 5–10, 2006.

[170] K. Zhang and V. E. Korepin, "Depth optimization of quantum search algorithms beyond Grover's algorithm," *Phys. Rev. A*, vol. 101, p. 032346, 2020.

[171] K. Zhang, P. Rao, K. Yu, H. Lim, and V. Korepin, "Implementation of efficient quantum search algorithms on NISQ computers," *Quantum Information Processing*, vol. 20, p. 233, 2021.

[172] L. K. Grover, "Trade-offs in the quantum search algorithm," *Phys. Rev. A*, vol. 66, p. 052314, 2002.

[173] M. Grassl, B. Langenberg, M. Roetteler, and R. Steinwandt, "Applying Grover's algorithm to AES: Quantum Resource Estimates," in *Post-Quantum Cryptography*, pp. 29–43, Springer International Publishing, 2016.

[174] M. Rahman and G. Paul, "Grover on KATAN: Quantum Resource Estimation," *IEEE Transactions on Quantum Engineering*, vol. 3, pp. 1–9, 2022.

[175] R. Anand, A. Maitra, and S. Mukhopadhyay, "Grover on SIMON," *Quantum Information Processing*, vol. 19, p. 340, 2020.

[176] K.-B. Jang, G.-J. Song, H.-J. Kim, and H.-J. Seo, "Grover on Simplified AES," in *2021 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, pp. 1–4, 2021.

[177] S. Jaques, M. Naehrig, M. Roetteler, and F. Virdia, "Implementing Grover Oracles for Quantum Key Search on AES and LowMC," in *Advances in Cryptology – EUROCRYPT 2020*, pp. 280–310, Springer International Publishing, 2020.

[178] M. Almazrooie, A. Samsudin, R. Abdullah, and K. N. Mutter, "Quantum reversible circuit of AES-128," *Quantum Information Processing*, vol. 17, p. 112, 2018.

[179] Y. Aharonov, L. Davidovich, and N. Zagury, "Quantum random walks," *Phys. Rev. A*, vol. 48, pp. 1687–1690, 1993.

[180] E. Farhi and S. Gutmann, "Quantum computation and decision trees," *Physical*

Review A, vol. 58, p. 915, 1998.

[181] A. Ambainis, "Quantum walk algorithm for element distinctness," *SIAM Journal on Computing*, vol. 37, pp. 210–239, 2007.

[182] N. Shenvi, J. Kempe, and K. B. Whaley, "Quantum random-walk search algorithm," *Physical Review A*, vol. 67, p. 052307, 2003.

[183] F. Magniez, M. Santha, and M. Szegedy, "Quantum Algorithms for the Triangle Problem," *SIAM Journal on Computing*, vol. 37, pp. 413–424, 2007.

[184] R. Portugal, *Quantum Walks and Search Algorithms*. New York: Springer, 2013.

[185] M. Szegedy, "Quantum speed-up of Markov chain based algorithms," *45th Annual IEEE Symposium on Foundations of Computer Science*, pp. 32–41, 2004.

[186] D. Aharonov, A. Ambainis, J. Kempe, and U. Vazirani, "Quantum walks on graphs," in *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, STOC '01, pp. 50–59, 2001.

[187] A. Childs, "Quantum algorithms: LECTURE 14. Discrete-time quantum walk," *University of Waterloo*, 2008.

[188] T. G. Wong, "Equivalence of Szegedy's and coined quantum walks," *Quantum Information Processing*, vol. 16, p. 215, 2017.

[189] T. G. Wong, "Coined quantum walks on weighted graphs," *Journal of Physics A: Mathematical and Theoretical*, vol. 50, p. 475301, 2017.

[190] G. D. Paparo and M. A. Martin-Delgado, "Google in a Quantum Network," *Scientific Reports*, vol. 2, p. 444, 2012.

[191] G. D. Paparo, M. M., F. Comellas, and M. A. Martin-Delgado, "Quantum Google in a Complex Network," *Scientific Reports*, vol. 3, p. 2773, 2013.

[192] S. A. Ortega and M. A. Martin-Delgado, "Generalized quantum PageRank algorithm with arbitrary phase rotations," *Physical Review Research*, vol. 5, p. 013061, 2023.

[193] J. Lemieux, B. Heim, D. Poulin, K. Svore, and M. Troyer, "Efficient Quantum Walk Circuits for Metropolis-Hastings Algorithm," *Quantum*, vol. 4, p. 287, 2020.

[194] P. A. M. Casares, R. Campos, and M. A. Martin-Delgado, "QFold: Quantum Walks and Deep Learning to Solve Protein Folding," *Quantum Science and Technology*, vol. 7, p. 025013, 2022.

[195] R. Campos, P. A. Casares, and M. A. Martin-Delgado, "Quantum Metropolis

Solver: QMS," *Quantum Machine Intelligence*, vol. 5, p. 28, 2023.

[196] G. Escrig, R. Campos, P. A. Moreno Casares, and M. A. Martin-Delgado, "Parameter Estimation of Gravitational Waves with a Quantum Metropolis Algorithm," *Classical and Quantum Gravity*, vol. 40, p. 045001, 2022.

[197] G. Escrig, R. Campos, H. Qi, and M. A. Martin-Delgado, "Quantum Bayesian Inference with Renormalization for Gravitational Waves.," *The Astrophysical Journal Letters*, vol. 979, p. L36, 2025.

[198] F. Magniez, A. Nayak, J. Roland, and M. Santha, "Search via Quantum Walk.," *SIAM Journal on Computing*, vol. 40, pp. 142–164, 2011.

[199] H. Wang, J. Wu, X. Yang, P. Chen, and X. Yi, "An Enhanced Quantum PageRank Algorithm Integrated with Quantum Search," *2014 Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, vol. IEEE, pp. 74–81, 2014.

[200] R. A. Santos, "Szegedy's quantum walk with queries," *Quantum Information Processing*, vol. 15, pp. 4461–4475, 2016.

[201] G. D. Paparo, V. Dunjko, A. Makmal, M. A. Martin-Delgado, and H. J. Briegel, "Quantum Speedup for Active Learning Agents," *Physical Review X*, vol. 4, p. 031002, 2014.

[202] S. A. Ortega and M. A. Martin-Delgado, "Complex-phase extensions of the Szegedy quantum walk on graphs," *Phys. Rev. A*, vol. 111, p. 032216, 2025.

[203] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Comput. Netw. ISDN Syst.*, vol. 30, no. 1–7, p. 107–117, 1998.

[204] S. A. Ortega and M. A. Martin-Delgado, "Discrete-time Semiclassical Szegedy Quantum Walks," *Physica A*, vol. 625, p. 129021, 2023.

[205] S. A. Ortega and M. A. Martin-Delgado, "Randomized SearchRank: A Semiclassical Approach to a Quantum Search Engine," *Physical Review Research*, vol. 6, p. 043014, 2024.

[206] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of State Calculations by Fast Computing Machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[207] W. K. Hastings, "Monte Carlo Sampling Methods Using Markov Chains and Their Applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970.

[208] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[209] S. A. Ortega, P. Fernández, and M. A. Martin-Delgado, "Implementing semiclassical Szegedy walks in classical-quantum circuits for homomorphic encryption," *Journal of Physics: Complexity*, vol. 6, no. 2, p. 025010, 2025.

[210] C. F. Chiang, D. Nagaj, and P. Wocjan, "Efficient circuits for quantum walks.," *Quantum Information and Computation*, vol. 10, pp. 420–434, 2010.

[211] S. A. Ortega and M. A. Martin-Delgado, "SQUWALS: A Szegedy QUantum WALks Simulator," *Advanced Quantum Technologies*, p. 2400022, 2024.

[212] S. Ganjian, C. Paddock, and A. Broadbent, "Demonstrating Quantum Homomorphic Encryption Through Simulation.," *arXiv:2406.16247*, 2024.

[213] M. Yarter, G. Uehara, and A. Spanias, "Implementation and analysis of quantum homomorphic encryption.," in *2022 13th International Conference on Information, Intelligence, Systems and Applications (IISA). IEEE*, pp. 1–5, 2022.

[214] T. Loke and J. B. Wang, "Efficient quantum circuits for Szegedy quantum walks," *Annals of Physics*, vol. 382, pp. 64–84, 2017.

[215] A. Hagberg, D. S Chult, and P. Swart, "Exploring Network Structure, Dynamics, and Function using NetworkX," in *Proceedings of the 7th Python in Science Conference* (G. Varoquaux, T. Vaught, and J. Millman, eds.), (Pasadena, CA USA), pp. 11 – 15, 2008.

[216] N. J. Ross and P. Selinger, "Optimal ancilla-free Clifford+T approximation of z-rotations.," *Quantum Information and Computation*, vol. 16, pp. 901–953, 2016.

[217] S. A. Ortega, P. Fernández, and M. A. Martin-Delgado, "CQC-QHE: Classical-Quantum Circuits for Quantum Homomorphic Encryption." `https://github.com/OrtegaSA/cqc-qhe-repo`, 2024.

[218] B. Zindorf and S. Bose, "Efficient Implementation of Multi-Controlled Quantum Gates," *arXiv:2404.02279*, 2024.