

Quantum Walks and Application to Quantum Money

Jake Doliskani* and Seyed Ali Mousavi†

Department of Computing and Software, McMaster University

*jake.doliskani@mcmaster.ca

†mousas26@mcmaster.ca

1 Introduction to Quantum Computation

Introduction

Quantum computing is a revolutionary field that combines quantum mechanics and information theory to redefine computation and information processing. In the latter part of the 20th century, scientists explored the fusion of quantum mechanics and information theory, leading to the birth of quantum information science. This field challenges the classical view of computation by introducing novel concepts like quantum bits (qubits), entanglement, and quantum superposition, which have enabled new algorithms and protocols that outpace their classical counterparts in specific tasks. Classical computers are grounded in bits, which take binary values (0 or 1). Quantum computing introduces the qubit, a fundamental unit of quantum information that can exist in a superposition of states, such as 0 and 1 simultaneously. This foundational difference enables quantum computers to process information in fundamentally new ways. Unlike technologies like DNA computing or optical computing, which describe changes in the physical substrate while retaining classical computational principles, quantum computing changes the computational paradigm itself. A quantum computer uses the principles of quantum mechanics, such as superposition and entanglement, rather than classical mechanics, to process information. In the 1980s, pioneers like Richard Feynman and David Deutsch recognized that certain quantum phenomena could not be efficiently simulated on classical computers. These insights led to the exploration of quantum Turing machines and quantum circuit models, which provided a theoretical framework for quantum computing. The discovery of quantum gates and their role in quantum algorithms formalized the field.

Early quantum algorithms demonstrated that quantum computing could solve certain problems more efficiently than classical methods. Notably, in 1994, Peter Shor introduced a polynomial-time quantum algorithm for integer factorization, which threatened classical cryptographic protocols relying on the difficulty of factoring large integers. Entanglement, a uniquely quantum phenomenon, allows particles to exhibit correlations that defy classical explanation. This property is crucial for many quantum algorithms and protocols, as it enables quantum computers to process and store information in a way that classical computers cannot. Two notable algorithms underscore the potential of quantum computing: Shor's Algorithm, which efficiently factors integers, undermining RSA encryption and other cryptographic systems based on the hardness of factoring, and Grover's Algorithm, which provides a quadratic speedup for unstructured search problems, such as searching an unsorted database. These algorithms exemplify the theoretical advantages of quantum computing over classical approaches, even though practical implementations remain challenging.

Quantum systems are fragile and susceptible to decoherence, where quantum states lose their coherence due to interactions with the environment. This makes maintaining quantum states for computation a significant challenge. Quantum error correction methods were developed to address decoherence and other quantum noise. The breakthrough work of Shor and Steane in the mid-1990s introduced error-correction codes that allowed reliable computation despite quantum noise. Building scalable quantum computers requires advances in hardware and experimental techniques. As of now, only small-scale quantum systems with a few qubits have been implemented successfully in laboratories. Quantum computing does not offer universal speedups for all problems. For example, Grover's algorithm provides only a quadratic speedup for unstructured search, and certain problems remain equally challenging for both quantum and classical computers.

Quantum key distribution protocols, like BB84 and Ekert's protocol, offer provably secure methods for communication based on the principles of quantum mechanics. Unlike classical cryptography, which relies on computational assumptions, quantum cryptography guarantees security through physical principles. The quantum perspective has provided new insights into classical

computing and inspired novel classical algorithms. It has also advanced simulation techniques for quantum systems, benefiting fields like material science and chemistry. Quantum information processing has deepened our understanding of quantum mechanics, shedding light on foundational questions about quantum measurement and entanglement. For example, experiments testing Bell's inequalities have confirmed the non-classical correlations predicted by quantum theory.

While practical quantum computing is still in its infancy, significant progress has been made. Quantum hardware companies like IBM, Google, and Rigetti have developed small-scale quantum processors capable of performing limited computations. Platforms such as Qiskit and Cirq enable researchers to experiment with quantum programming and algorithm development. Efforts to build scalable, fault-tolerant quantum computers are ongoing, alongside investigations into alternative models of quantum computation, such as topological and cluster-state quantum computing. Despite these advances, many open questions remain about the scope and ultimate power of quantum computation. While quantum computers will not replace classical ones for all tasks, they promise to revolutionize fields where their unique capabilities provide exponential speedups or new forms of computation.

Quantum computing represents a profound shift in how we understand and leverage computation. By replacing classical mechanics with quantum mechanics as the foundation for processing information, quantum computing has opened new avenues for scientific discovery and technological innovation. While challenges persist, the theoretical and experimental advances made thus far underscore the transformative potential of this exciting field.

Foundations of Quantum Computation

Understanding a Qubit

A qubit (quantum bit) is the fundamental unit of quantum information, analogous to a classical bit. However, unlike a classical bit, which can only exist in one of two definite states (0 or 1), a qubit can exist in a linear combination, or superposition, of both states simultaneously. Mathematically, the state of a single qubit is represented as:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1}$$

Here:

- $|0\rangle$ and $|1\rangle$ are the basis states (analogous to 0 and 1 in classical computing).
- α and β are complex numbers known as probability amplitudes.

The normalization condition $|\alpha|^2 + |\beta|^2 = 1$ ensures that the probabilities of measuring the qubit in either the $|0\rangle$ or $|1\rangle$ state sum to 1.

The qubit's unique ability to exist in a superposition of states is what gives quantum computers their immense computational potential, enabling them to process and store information in fundamentally different ways than classical computers.

Superposition

Superposition is a quantum phenomenon where a qubit exists in a combination of multiple states simultaneously. For example, a qubit in the state:

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \quad (2)$$

is in an equal superposition of $|0\rangle$ and $|1\rangle$. This means that if we measure the qubit, there is an equal probability (50%) of observing it in either state.

Key properties of superposition:

- **Parallelism:** When a quantum system is in superposition, it can perform computations for all possible states simultaneously. For instance, a single qubit can encode two states ($|0\rangle$ and $|1\rangle$) at the same time, while n qubits can encode 2^n states.
- **Measurement Collapse:** When a qubit in superposition is measured, it collapses into one of its basis states, $|0\rangle$ or $|1\rangle$, with a probability given by $|\alpha|^2$ and $|\beta|^2$, respectively.

Superposition allows quantum systems to explore multiple possibilities in parallel, which is critical for quantum algorithms such as Grover’s search or Shor’s factoring.

Entanglement

Entanglement is a uniquely quantum phenomenon where two or more qubits become correlated in such a way that the state of one qubit is directly related to the state of the other, regardless of the physical distance between them. When qubits are entangled, the measurement of one qubit instantly determines the state of the other.

An example of an entangled state for two qubits is the Bell state:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \quad (3)$$

Here:

- $|00\rangle$ means both qubits are in the $|0\rangle$ state.
- $|11\rangle$ means both qubits are in the $|1\rangle$ state.

If one qubit is measured to be $|0\rangle$, the other qubit will instantly collapse to $|0\rangle$, and similarly for $|1\rangle$, regardless of the distance between them.

Key features of entanglement:

- **Non-Local Correlations:** Entanglement defies classical intuition, as it suggests that measurement outcomes are correlated even across vast distances, a phenomenon supported by experiments validating Bell’s Theorem.
- **Applications:** Entanglement is a resource for many quantum technologies, including quantum teleportation, quantum cryptography (e.g., secure communication via the BB84 protocol), and quantum error correction.

Entanglement, along with superposition, forms the backbone of quantum computing and quantum communication, enabling capabilities that are impossible in the classical world.

Quantum Gates and Circuits

Quantum logic gates are the fundamental building blocks of quantum circuits, manipulating qubits—the basic units of quantum information. Unlike classical logic gates, quantum gates are reversible and represented by unitary matrices, ensuring the preservation of quantum information.

Key Quantum Logic Gates

1. Pauli-X Gate (NOT Gate)

Operation: Flips the state of a qubit from $|0\rangle$ to $|1\rangle$ and vice versa.

Matrix Representation:

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

2. Hadamard Gate (H Gate)

Operation: Creates a superposition state, transforming $|0\rangle$ into $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|1\rangle$ into $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$.

Matrix Representation:

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

3. Controlled-NOT Gate (CNOT Gate)

Operation: A two-qubit gate that flips the state of the target qubit if the control qubit is in the state $|1\rangle$.

Matrix Representation:

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

4. Toffoli Gate (CCNOT Gate)

Operation: A three-qubit gate that flips the state of the target qubit if both control qubits are in the state $|1\rangle$.

Matrix Representation:

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

Universal Gates

A set of gates is considered **universal** if any unitary operation can be approximated to arbitrary precision using a sequence of gates from this set. For example, single-qubit rotation gates combined with the CNOT gate form a universal set, enabling the construction of any quantum algorithm.

2 Introduction to Quantum Walks

The concept of a *quantum walk* is a quantum analog of the classical random walk and plays a significant role in quantum algorithms. There are two main types of quantum walks: *continuous-time* and *discrete-time* quantum walks, both of which exhibit behaviors that are significantly different from classical random walks.

Continuous-Time Quantum Walks

Classical continuous-time random walks (Markov chains) are a foundational concept in stochastic processes, where time is treated as a continuous variable. In this framework, a walker moves from one vertex of a graph to an adjacent vertex at any time, with probabilities evolving over time. The probability of the walker being at a vertex can be visualized as a liquid seeping from one vertex to its neighbors. Initially, the walker is most likely to be found at the starting vertex, but as time progresses, the probability shifts to neighboring vertices. This dynamic is governed by a constant transition rate γ , uniform across all vertices and time.

To model this mathematically, the probability of transitioning from one vertex to another in an infinitesimal time interval Δt is proportional to $\gamma\Delta t$. For a vertex x_j with degree d_j (the number of neighboring vertices), the probability of the walker staying on x_j is

$$1 - d_j\gamma\Delta t,$$

while the probability of transitioning to a neighboring vertex is

$$\gamma\Delta t.$$

The evolution of probabilities over time is captured by the transition matrix $M(t)$, where the entry $M_{ij}(t)$ represents the probability of transitioning from vertex x_j to vertex x_i in time t . The dynamics of the system are described by a differential equation derived from the transition matrix and an auxiliary matrix H , known as the generating matrix. This equation,

$$\frac{dM(t)}{dt} = -HM(t),$$

has a solution expressed as

$$M(t) = e^{-Ht},$$

which allows for the computation of probability distributions at any given time. The solution is verified using the Taylor series expansion of the exponential function.

The passage from classical continuous-time Markov chains to quantum walks involves a standard quantization process. In classical random walks, probabilities evolve via the transition matrix M . In contrast, the quantization process replaces the vector of probabilities with a state vector (probability amplitudes) and the non-unitary transition matrix with a unitary evolution operator, as required by quantum mechanics. This involves transforming the generating matrix H into a Hermitian operator and then constructing a unitary operator

$$U(t) = e^{-iHt}$$

by multiplying H with the imaginary unit i .

In a continuous-time quantum walk, the quantum state evolves over time according to the unitary operator $U(t)$. If the initial state of the system is $|\psi(0)\rangle$, the state at time t is given by

$$|\psi(t)\rangle = U(t)|\psi(0)\rangle.$$

The probability distribution for observing the walker at a specific vertex k is computed as

$$p_k = |\langle k|\psi(t)\rangle|^2,$$

where $|k\rangle$ represents the state corresponding to vertex k .

The continuous-time quantum walk introduces several key differences from its classical counterpart. Unlike classical random walks, which rely on probabilities, quantum walks involve probability amplitudes, enabling interference effects. These effects allow quantum walks to explore the graph more efficiently, leading to applications in quantum algorithms with speedups over classical approaches. Additionally, the transition matrix in classical random walks is replaced with a Hermitian operator in the quantum case, ensuring unitary evolution.

*****Examples and further discussions here*****

Discrete-Time Quantum Walks

***** The discrete-time coined quantum walk (DTQW) is a quantum analog of classical random walks, leveraging the principles of quantum mechanics to achieve unique computational and dynamical properties. It is defined on a graph or lattice, with the walker's state described by both its position and an internal "coin" state. The walker's position n on a one-dimensional lattice (line) is represented by a vector $|n\rangle$ in an infinite-dimensional Hilbert space \mathcal{H}_P , where the computational basis for \mathcal{H}_P is

$$\{|n\rangle : n \in \mathbb{Z}\}.$$

The walker's internal state, or "coin," is represented by a two-dimensional Hilbert space \mathcal{H}_C with basis states $|0\rangle$ and $|1\rangle$. The coin state determines the direction of the walker's movement, and the total state of the system resides in the tensor product space

$$\mathcal{H} = \mathcal{H}_C \otimes \mathcal{H}_P,$$

combining the coin and position spaces.

The dynamics of the walk are governed by two key operators: the coin operator and the shift operator. The coin is represented by a unitary matrix C acting on \mathcal{H}_C . Common choices include the Hadamard matrix, which creates superpositions of the coin states. The Hadamard coin operator H is defined as

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

The shift operator S moves the walker based on the coin state. For example, if the coin state is $|0\rangle$, the walker moves to $|n+1\rangle$, and if the coin state is $|1\rangle$, the walker moves to $|n-1\rangle$. The shift operator is explicitly given by

$$S = \sum_n |n+1\rangle\langle n| \otimes |0\rangle\langle 0| + |n-1\rangle\langle n| \otimes |1\rangle\langle 1|.$$

The evolution of the system is described by the unitary operator

$$U = S \cdot (I \otimes C),$$

where I is the identity operator on \mathcal{H}_P . The state of the walker at time $t + 1$ is given by

$$|\psi(t + 1)\rangle = U|\psi(t)\rangle.$$

Consider a quantum walk on a one-dimensional lattice. The walker's state evolves in superposition, allowing it to explore multiple paths simultaneously. Interference between these paths leads to faster spreading compared to classical random walks.

Key features of DTQW include superposition, where the walker exists in a superposition of positions and coin states, enabling parallel exploration of multiple paths; interference, where quantum interference enhances or suppresses certain paths, leading to non-classical behavior; and quadratic speedup, where quantum walks spread faster than classical random walks, often quadratically, which is advantageous for algorithmic applications.

DTQWs are used in quantum algorithms, such as quantum search and graph traversal, and in simulating physical systems like energy transport in photosynthesis. They also serve as a framework for universal quantum computation. The discrete-time quantum walk is a powerful tool in quantum computing, offering a rich interplay between quantum mechanics and graph theory. Its ability to exploit superposition and interference enables applications in algorithm design, simulation, and computation, making it a cornerstone of quantum information science.

*****Examples and further discussions here*****

Comparison of Models

From a computational standpoint, both continuous-time and discrete-time quantum walks offer distinct advantages. The continuous-time quantum walk is often considered more natural because it does not require expanding the state space, making it simpler to define and typically easier to analyze. For example, while the exponential speedup demonstrated by continuous-time quantum walks is widely believed to extend to the discrete-time model, the dynamics of the latter are more complex. To date, no rigorous analysis has been provided to confirm this extension.

In contrast, the discrete-time quantum walk is generally more straightforward to implement using quantum circuits. Implementing continuous-time quantum walks, however, often relies on techniques that require the graph's maximum degree to be small. This constraint makes certain algorithms challenging to replicate using continuous-time quantum walks. Therefore, while continuous-time quantum walks excel in simplicity and analytical tractability, discrete-time quantum walks offer greater practicality and ease of implementation in quantum computing applications.

3 Group Actions

Group actions provide a framework for understanding how groups interact with sets, formalizing the concept of applying transformations systematically. They capture the symmetries and structure-preserving transformations of objects, making them fundamental in mathematics and its applications. Group actions describe transformations through a set of rules that ensure consistency and compatibility with the underlying group operations. This concept is pivotal in cryptography, where the symmetries and operations of groups underpin many secure systems.

In cryptography, group actions are critical to protocols like Elliptic Curve Cryptography (ECC), which relies on the difficulty of reversing group-based operations to ensure security. Isogeny-based

cryptography, a post-quantum cryptographic candidate, uses group actions on elliptic curves to create complex mathematical problems resistant to quantum attacks. Similarly, lattice-based and code-based cryptography use group actions to study transformations in structures that form the basis for secure communication systems. Non-abelian group actions are being explored for their potential to create quantum-resistant algorithms.

In quantum computing, group actions are central to quantum walks, a quantum equivalent of classical random walks. Quantum walks leverage group actions to describe the evolution of quantum states and exploit the symmetries of these states for tasks like search, optimization, and cryptographic protocol design. This makes them valuable in constructing secure quantum algorithms and protocols, particularly in quantum key distribution and graph-based cryptography. Overall, group actions bridge abstract mathematical theory with practical applications in classical and quantum cryptography.

Definition of Group Action

Group actions form a bridge between group theory and other mathematical structures, offering a systematic way to study symmetry, transformations, and invariants. Let us delve deeper into their properties, classifications, and applications.

Formal Definition

A **group action** of a group G on a set X is a map $\cdot : G \times X \rightarrow X$ that satisfies:

1. **Identity Property:**

$$e \cdot x = x \quad \text{for all } x \in X,$$

where e is the identity element of G .

2. **Compatibility (Associativity):**

$$(gh) \cdot x = g \cdot (h \cdot x) \quad \text{for all } g, h \in G, x \in X.$$

Alternatively, the group action can be viewed as a homomorphism $\phi : G \rightarrow \text{Sym}(X)$, where $\text{Sym}(X)$ is the group of all permutations of X .

Types of Group Actions

1. **Faithful Action:** The action is faithful if $g \cdot x = x$ for all $x \in X$ implies $g = e$. Equivalently, the homomorphism ϕ is injective.
2. **Transitive Action:** The action is transitive if, for any $x, y \in X$, there exists $g \in G$ such that $g \cdot x = y$. In this case, X consists of a single orbit.
3. **Free Action:** The action is free if $g \cdot x = x$ implies $g = e$ for all $x \in X$.
4. **Regular Action:** The action is both transitive and free, meaning there is exactly one $g \in G$ for each pair (x, y) such that $g \cdot x = y$.
5. **Effective Action:** The action is effective if the only group element acting as the identity on X is e .

Key Concepts in Group Actions

1. **Orbits:** For $x \in X$, the **orbit** of x is:

$$\text{Orb}(x) = \{g \cdot x \mid g \in G\}.$$

Orbits partition the set X , and each orbit is a subset where the action appears “connected.”

2. **Stabilizer Subgroup:** The **stabilizer** of $x \in X$ is:

$$\text{Stab}_G(x) = \{g \in G \mid g \cdot x = x\}.$$

It is a subgroup of G and describes the symmetries that leave x unchanged.

3. **Orbit-Stabilizer Theorem:** This theorem links the size of an orbit and its stabilizer:

$$|G| = |\text{Orb}(x)| \cdot |\text{Stab}_G(x)|.$$

4. **Fixed Points:** A point $x \in X$ is a **fixed point** if $g \cdot x = x$ for all $g \in G$. The set of fixed points is:

$$\text{Fix}(G) = \{x \in X \mid g \cdot x = x, \forall g \in G\}.$$

5. **Invariant Subsets:** A subset $Y \subseteq X$ is **invariant** under the action if $g \cdot y \in Y$ for all $g \in G$ and $y \in Y$.

Groups and Characters

4 Abelian Groups

An **abelian group** (also called a **commutative group**) is a set G equipped with a binary operation \cdot (usually written as addition $+$ or multiplication \cdot) that satisfies the following four group axioms, along with an additional **commutativity** property.

4.1 Axioms of an Abelian Group

A set G with a binary operation \cdot is an **abelian group** if the following conditions hold:

1. **Closure:** For all $a, b \in G$, the operation $a \cdot b$ produces another element in G :

$$a \cdot b \in G.$$

2. **Associativity:** For all $a, b, c \in G$, the operation satisfies:

$$(a \cdot b) \cdot c = a \cdot (b \cdot c).$$

3. **Identity Element:** There exists an element $e \in G$ such that for all $a \in G$:

$$e \cdot a = a \cdot e = a.$$

This element e is called the **identity element**.

4. **Inverse Element:** For every element $a \in G$, there exists an element $a^{-1} \in G$ such that:

$$a \cdot a^{-1} = a^{-1} \cdot a = e.$$

5. **Commutativity (Abelian Property):** For all $a, b \in G$, the operation is **commutative**:

$$a \cdot b = b \cdot a.$$

Since abelian groups satisfy commutativity, they allow more flexible mathematical operations and structure, leading to applications in algebra, number theory, and topology.

4.2 Characters of Abelian Groups

Let G be an abelian group. A function

$$\chi : G \rightarrow \mathbb{C}^\times$$

that maps G to the group of nonzero complex numbers $\mathbb{C}^\times = \mathbb{C} \setminus \{0\}$, is called a **character** of G if it is a **group homomorphism**. That is, for all $g_1, g_2 \in G$, the function satisfies:

$$\chi(g_1 g_2) = \chi(g_1) \chi(g_2).$$

Character Values for Finite Groups

If χ is a character of a finite group (or more generally, a torsion group) G , then each function value $\chi(g)$ is a root of unity. This follows because, for each $g \in G$, there exists some integer $k \in \mathbb{N}$ such that:

$$g^k = e.$$

Applying χ to both sides, we obtain:

$$\chi(g)^k = \chi(g^k) = \chi(e) = 1.$$

Number of Characters in a Finite Abelian Group

A **finite abelian group** of order n has exactly n distinct characters, denoted by:

$$\chi_1, \chi_2, \dots, \chi_n.$$

The function χ_1 , defined by:

$$\chi_1(g) = 1, \quad \forall g \in G,$$

is called the **principal character** of G . The remaining $n - 1$ characters are referred to as **non-principal characters**.

Character Group

If G is an abelian group, the set of characters $\{\chi_k\}$ forms an abelian group under pointwise multiplication, defined as:

$$(\chi_j \chi_k)(g) = \chi_j(g) \chi_k(g), \quad \forall g \in G.$$

This group is called the **character group** of G and is often denoted as \hat{G} . The **identity element** of \hat{G} is the principal character χ_1 , and the **inverse** of a character χ_k is its reciprocal:

$$(\chi_k)^{-1} = \frac{1}{\chi_k}.$$

If G is finite of order n , then the character group \hat{G} is also of order n . In this case, since

$$|\chi_k(g)| = 1, \quad \forall g \in G,$$

the inverse of a character is simply its **complex conjugate**:

$$\chi_k^{-1}(g) = \overline{\chi_k(g)}.$$

5 Quantum Fourier Transform

The **Quantum Fourier Transform (QFT)** is one of the most fundamental unitary transformations in quantum computing. It plays a crucial role in quantum algorithms, including Shor's algorithm for integer factorization and quantum phase estimation. The QFT is the quantum analog of the classical discrete Fourier transform (DFT) but operates on quantum states, enabling efficient quantum computation of Fourier coefficients.

In this section, we discuss the QFT in the context of abelian groups, with a focus on its implementation over \mathbb{Z}_{2^n} , one of the most commonly used finite groups in quantum algorithms, including quantum walks.

6 Quantum Fourier Transform Over an Abelian Group

Let G be an abelian group. The **Quantum Fourier Transform** over G is defined as:

$$F_G = \frac{1}{\sqrt{|G|}} \sum_{x \in G} \sum_{y \in \hat{G}} \chi_y(x) |y\rangle \langle x|$$

where:

- \hat{G} is a complete set of characters of G .
- $\chi_y(x)$ denotes the y th character of G , evaluated at x .
- $|x\rangle$ and $|y\rangle$ are quantum states corresponding to elements of G and its dual group \hat{G} .

Since G and \hat{G} are isomorphic, it is often useful to label the elements of \hat{G} using elements of G . The unitary nature of the QFT follows from the **orthogonality of characters**.

6.1 QFT Over \mathbb{Z}_{2^n}

The QFT over $G = \mathbb{Z}_{2^n}$ is defined as:

$$F_{\mathbb{Z}_{2^n}} = \frac{1}{\sqrt{2^n}} \sum_{x,y \in \mathbb{Z}_{2^n}} \omega_{2^n}^{xy} |y\rangle \langle x|$$

where:

$$\omega_m = e^{2\pi i/m}$$

is the **primitive** m th root of unity. This transformation maps an input basis state $|x\rangle$ as follows:

$$|x\rangle \rightarrow \frac{1}{\sqrt{2^n}} \sum_{y \in \mathbb{Z}_{2^n}} \omega_{2^n}^{xy} |y\rangle.$$

This equation represents a **linear transformation** where each input state $|x\rangle$ is mapped to a superposition of all basis states, weighted by phase factors.

6.2 Binary Representation and QFT Implementation

To implement this transformation efficiently, it is useful to express x and y in binary form:

$$x = x_{n-1}x_{n-2} \dots x_1x_0.$$

The exponent in the Fourier transform formula can then be rewritten using its binary expansion:

$$\omega_{2^n}^{xy} = e^{2\pi i x \sum_{k=0}^{n-1} y_k 2^k / 2^n}.$$

Rewriting the transformation explicitly:

$$|x\rangle \rightarrow \frac{1}{\sqrt{2^n}} \sum_{y \in \mathbb{Z}_{2^n}} e^{2\pi i x \sum_{k=0}^{n-1} y_k 2^k / 2^n} |y\rangle.$$

This can be decomposed into a **tensor product of single-qubit states**, where each qubit state depends on the least significant bits of x :

$$\bigotimes_{k=0}^{n-1} |z_k\rangle.$$

where

$$|z_k\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(x_0 2^{-n+k} + x_1 2^{-n+k+1} + \dots + x_{n-1} 2^{-1})} |1\rangle).$$

This decomposition highlights the **hierarchical dependence of qubits on the binary digits of x** , making it possible to implement the QFT using **Hadamard gates and controlled phase shift gates**.

Cayley graphs

A Cayley graph is a graphical representation of a group based on a chosen set of generators. It is widely used in algebra, combinatorics, and theoretical computer science to study group structures and their applications in network theory, quantum computing, and cryptography.

Definition of Cayley Graphs

A *Cayley graph* is a graph that encodes the structure of a group with respect to a chosen generating set. Formally, let G be a group and S be a subset of G that serves as a generating set, meaning every element of G can be expressed as a finite product of elements from S . The Cayley graph $\Gamma(G, S)$ is defined as follows. The vertex set of the graph consists of the elements of G , where each vertex corresponds to a unique group element. The edge set is constructed by drawing a directed edge from vertex g to vertex gs for each $g \in G$ and $s \in S$. If the generating set S is symmetric, meaning that whenever $s \in S$, its inverse s^{-1} is also in S , then the Cayley graph is undirected; otherwise, it is a directed graph. For example, For the cyclic group $G = \mathbb{Z}_n = \{0, 1, 2, \dots, n-1\}$ with the generating set $S = \{\pm 1\}$, the Cayley graph forms a cycle graph. In this graph, each vertex j is connected to $j+1 \pmod n$ and $j-1 \pmod n$, creating a cyclic structure.

The adjacency matrix of Γ can be expressed as

$$A = \sum_{a \in G} \lambda_a |\hat{a}\rangle \langle \hat{a}|,$$

where $|\hat{a}\rangle$ is the quantum Fourier transform of $|a\rangle$. The eigenvalues λ are given by

$$\lambda_a = \sum_{q \in Q} \chi(a, q).$$

Note that the eigenvectors $|\hat{a}\rangle$ of A depend only on G and not on the set Q .

To see this, we have

$$\begin{aligned} A|\hat{a}\rangle &= A \cdot \frac{1}{\sqrt{|G|}} \sum_{y \in G} \chi(a, y) |y\rangle = \frac{1}{\sqrt{|G|}} \sum_{y \in G} \chi(a, y) \cdot A|y\rangle \\ &= \frac{1}{\sqrt{|G|}} \sum_{y \in G} \chi(a, y) \cdot \sum_{q \in Q} |qy\rangle \end{aligned}$$

Consider $\beta = qy$. Then:

$$\begin{aligned} &= \frac{1}{\sqrt{|G|}} \sum_{q \in Q} \chi(a, q) \sum_{\beta} \chi(a, \beta) |\beta\rangle = \sum_{q \in Q} \chi(a, q) \cdot |\hat{a}\rangle \\ &= \lambda_a |\hat{a}\rangle \end{aligned}$$

Quantum walks on Cayley graphs

Quantum walks on Cayley graphs play a crucial role in quantum computing, quantum search algorithms, and mathematical physics due to their deep connection with group theory and symmetry. The structured nature of Cayley graphs, derived from group elements and generators, ensures uniform exploration of the graph, leading to faster propagation compared to classical random walks. This property is particularly valuable in quantum algorithms, where quantum walks on Cayley graphs have been used to achieve exponential speedups in search and optimization problems. Additionally, their spectral properties, determined by the adjacency matrix and Fourier analysis on groups, provide insights into mixing times, state localization, and quantum transport phenomena. These advantages make Cayley graphs an essential framework for designing scalable and efficient quantum algorithms, offering a powerful bridge between algebraic structures and quantum mechanics.

Discrete-Time Quantum Walk on Cayley Graphs

A discrete-time quantum walk on a Cayley graph requires a coin operator to maintain unitary evolution.

State Space

The Hilbert space of the quantum walk is:

$$\mathcal{H} = \mathcal{H}_G \otimes \mathcal{H}_C,$$

where:

- \mathcal{H}_G is the space spanned by group elements (vertices).
- \mathcal{H}_C is the coin space, which determines movement directions.

Each state is represented as:

$$|\psi\rangle = \sum_{g \in G} \sum_{s \in S} \alpha_{g,s} |g\rangle \otimes |s\rangle,$$

where $|g\rangle$ is the vertex state and $|s\rangle$ represents the coin state.

Evolution Operators

Coin Flip Operator (C): This unitary operator acts on the coin space to create a superposition of movement directions. A common choice is the Grover or Hadamard operator.

Shift Operator (S): Moves the walker according to the group structure:

$$S|g\rangle \otimes |s\rangle = |gs\rangle \otimes |s\rangle.$$

This means that if the walker is at vertex g with a coin state s , it moves to gs .

Full Step Evolution:

$$U = S(C \otimes I),$$

where U is the total unitary evolution per step.

The probability of finding the walker at a particular vertex g after t steps is given by:

$$P(g, t) = \sum_{s \in S} |\langle g, s | \psi(t) \rangle|^2.$$

Continuous-Time Quantum Walk on Cayley Graphs

A continuous-time quantum walk on a Cayley graph follows a different evolution, governed by the Hamiltonian.

Hamiltonian and Evolution

The evolution of the quantum walk is determined by the Schrödinger equation:

$$i \frac{d}{dt} |\psi(t)\rangle = H |\psi(t)\rangle.$$

Here, the Hamiltonian H is usually chosen as:

$$H = A(\Gamma(G, S)),$$

where A is the adjacency matrix of the Cayley graph:

$$A_{g,h} = \begin{cases} 1, & \text{if } h = gs \text{ for some } s \in S, \\ 0, & \text{otherwise.} \end{cases}$$

Alternatively, the Laplacian L can be used:

$$H = L = D - A,$$

where D is the degree matrix:

$$D_{ij} = \begin{cases} \deg(v_i), & \text{if } i = j, \\ 0, & \text{otherwise.} \end{cases}$$

Here, $\deg(v_i)$ is the degree of vertex v_i , which is the number of edges connected to v_i .

The solution to the quantum walk is given by:

$$|\psi(t)\rangle = e^{-iHt} |\psi(0)\rangle.$$

This means the probability amplitude at a vertex g spreads over time according to the unitary evolution operator e^{-iHt} .

Example: Cyclic Group \mathbb{Z}_n

Consider the Cayley graph of \mathbb{Z}_n with generator set $S = \{\pm 1\}$, which forms a cycle graph. The adjacency matrix is:

$$A_{jk} = \begin{cases} 1, & \text{if } k = j \pm 1 \pmod n, \\ 0, & \text{otherwise.} \end{cases}$$

Its eigenvalues are:

$$\lambda_k = 2 \cos \left(\frac{2\pi k}{n} \right),$$

with eigenvectors given by the discrete Fourier transform (DFT) basis:

$$\psi_k(j) = e^{\frac{2\pi i k j}{n}}.$$

This result generalizes to other Abelian groups, where the eigenvectors are given by the Fourier basis of the group.

6.3 Simulating continuous-time walks

Childs [?] and Berry, Childs, and Kothari [?] showed that continuous-time quantum walks can be simulated using discrete-time quantum walks. Since our approach relies on their methods, we begin by outlining the relevant notation and framework.

Let H be a Hamiltonian of dimension $N = 2^n$, and define $\|H\|_{\max} = \max_{i,j} |H_{ij}|$. To facilitate the simulation, we expand the Hilbert space from \mathbb{C}^N to $\mathbb{C}^{2N} \otimes \mathbb{C}^{2N}$ by appending an ancilla qubit initialized to $|0\rangle$ and duplicating the resulting space.

To define the discrete-time walk operator, we first construct an orthonormal set of states:

$$\begin{aligned} |\phi_{j0}\rangle &:= \frac{1}{\sqrt{d}} \sum_{\ell \in F_j} |\ell\rangle \left(\sqrt{\frac{H_{j\ell}^*}{K}} |0\rangle + \sqrt{1 - \frac{|H_{j\ell}^*|}{K}} |1\rangle \right), \\ |\phi_{j1}\rangle &:= |0\rangle |1\rangle, \end{aligned} \quad (4)$$

where F_j denotes the set of nonzero entries in column j of H , and $K \geq \|H\|_{\max}$ is a fixed constant.

Based on these states, we define the isometry $T : \mathbb{C}^{2N} \rightarrow \mathbb{C}^{2N} \otimes \mathbb{C}^{2N}$ as follows:

$$T := \sum_{j=0}^{N-1} \sum_{b \in \{0,1\}} (|j\rangle\langle j| \otimes |b\rangle\langle b|) \otimes |\phi_{jb}\rangle. \quad (5)$$

The discrete-time quantum walk operator is then defined as

$$W = iS(2TT^* - \mathbb{1}),$$

where S is the swap operator acting as

$$S|j_1\rangle|b_1\rangle|j_2\rangle|b_2\rangle = |j_2\rangle|b_2\rangle|j_1\rangle|b_1\rangle,$$

for all $0 \leq j_1, j_2 < N$ and $b_1, b_2 \in \{0, 1\}$.

To efficiently simulate the continuous-time evolution e^{-iHt} , it suffices to implement the isometry T , its adjoint T^* , and the walk operator W efficiently. Assuming black-box (oracle) access to the entries of H , the following result holds:

Theorem 6.1 ([?, Theorem 1]). *Let H be a d -sparse Hamiltonian acting on n qubits. Then the unitary e^{-iHt} can be approximated to within error ϵ using*

$$O\left(\tau \frac{\log(\tau/\epsilon)}{\log \log(\tau/\epsilon)}\right)$$

queries to H , and

$$O\left(\tau \left[n + \log^{5/2}(\tau/\epsilon)\right] \frac{\log(\tau/\epsilon)}{\log \log(\tau/\epsilon)}\right)$$

additional elementary gates, where $\tau = d \cdot \|H\|_{\max} \cdot t$.

Quantum walks from Group Actions

Let G be an abelian group, and let $Q = \{q_1, q_2, \dots, q_k\} \subset G$ be a symmetric generating set, meaning that $q \in Q$ if and only if $-q \in Q$. The Cayley graph associated with G and Q is defined as $\Gamma = (V, E)$, where the vertex set is $V = G$, and the edge set consists of all pairs $(a, b) \in G \times G$ such that $b = q + a$ for some $q \in Q$.

The adjacency matrix A of Γ admits a spectral decomposition:

$$A = \sum_{a \in G} \lambda_a |\hat{a}\rangle \langle \hat{a}|,$$

where $|\hat{a}\rangle$ denotes the quantum Fourier transform of $|a\rangle$, and the eigenvalues λ_a are given by

$$\lambda_a = \sum_{q \in Q} \chi(a, q).$$

Importantly, the eigenvectors $|\hat{a}\rangle$ depend solely on the group G , not on the choice of the generating set Q .

Cayley graphs can also be generalized through group actions. Let $(G, X, *)$ be a regular group action with a fixed element $x \in X$, and let $Q = \{q_1, q_2, \dots, q_k\} \subset G$ be a subset as before. Define the Cayley graph $\Gamma = (X, E)$, where the vertex set is X , and the edge set consists of all pairs $(x, y) \in X \times X$ such that $y = q * x$ for some $q \in Q$.

The adjacency matrix of this graph can be written as:

$$A = \sum_{h \in G} \lambda_h |G^{(h)} * x\rangle \langle G^{(h)} * x|,$$

where

$$\lambda_h = \sum_{q \in Q} \chi(h, q), \quad \text{and} \quad |G^{(h)} * x\rangle = \frac{1}{\sqrt{|G|}} \sum_{g \in G} \chi(g, h) |g * x\rangle.$$

As in the previous case, the eigenvectors $|G^{(h)} * x\rangle$ are determined solely by the structure of the group G . This construction generalizes the standard Cayley graph: setting $X = G$ and the group action $*$ as group multiplication recovers the original definition.

To see this, we have:

$$\begin{aligned} A|G^{(h)} * x\rangle &= \frac{1}{\sqrt{|G|}} \cdot \sum_{g \in G} \chi(g, h) A|g * x\rangle \\ &= A|G^{(h)} * x\rangle = \frac{1}{\sqrt{|G|}} \cdot \sum_{g \in G} \chi(g, h) \sum_{q \in Q} |q * (g * x)\rangle \end{aligned}$$

Consider $\beta = qg$. Then:

$$\begin{aligned} &= \frac{1}{\sqrt{|G|}} \sum_{q \in Q} \sum_{\beta \in G} \chi(\beta, h) \chi(q, h) |\beta * x\rangle \\ &= \sum_{q \in Q} \chi(q, h) |G^{(h)} * x\rangle = \lambda_h |G^{(h)} * x\rangle \end{aligned}$$

Since the group action $(G, X, *)$ is regular, the two Cayley graph constructions are isomorphic. In the first construction, the vertex set is G , and the adjacency matrix is indexed by elements of G . In the second construction, the vertex set is X , with the adjacency matrix indexed by elements of X . The isomorphism between these graphs is given by the bijection

$$\begin{aligned} \phi : G &\longrightarrow X \\ g &\longmapsto g * x, \end{aligned}$$

which maps each group element $g \in G$ to the corresponding point in its orbit under the action.

When the group action $(G, X, *)$ is assumed to be cryptographic, the isomorphism ϕ becomes a one-way function: given $g \in G$, it is efficient to compute $g * x$, but given the pair $(x, g * x)$, it is computationally hard to recover g . As a result, while the two Cayley graphs are mathematically isomorphic, computational tasks associated with them may require fundamentally different approaches. A particularly illustrative example is the implementation of quantum walks on these graphs.¹

From this point forward, we assume the generating set Q is small, i.e., $|Q| = \text{poly}(\log|G|)$. In the first construction (where $X = G$), the walk e^{-iAt} can be efficiently implemented even for values of t that are exponentially large in $\log|G|$. This is a consequence of the identity:

$$e^{-iAt} = \text{QFT}_G \left(\sum_{a \in G} e^{-i\lambda_a t} |a\rangle\langle a| \right) \text{QFT}_G^*, \quad (6)$$

where QFT_G and its inverse can be implemented in $\text{poly}(\log|G|)$ time. Moreover, the diagonal phase unitary $|a\rangle \mapsto e^{-i\lambda_a t} |a\rangle$ is also efficient to implement, since the eigenvalues λ_a can be computed classically to arbitrary precision in $\text{poly}(\log|G|)$ time.

In contrast, the group-action setting presents greater computational difficulty. Although the states $|G^{(h)} * x\rangle$ play a role analogous to the Fourier basis states $|\hat{h}\rangle$, the presence of the action $*$ makes the situation more subtle. In particular, implementing general transformations over these states is difficult; we are typically limited to operations of the form $|y\rangle \mapsto |a * y\rangle$ for $a \in G$. As a result, no efficient decomposition similar to (6) is known in the group-action setting. Instead, the best we can express is the spectral decomposition:

$$e^{-iAt} = \sum_{h \in G} e^{-i\lambda_h t} |G^{(h)} * x\rangle\langle G^{(h)} * x|.$$

Despite this limitation, the adjacency matrix A retains a high degree of sparsity and algebraic structure. In the next section, we will show that for $t = \text{poly}(\log|G|)$, the walk e^{-iAt} can still be efficiently approximated to polynomial accuracy.

Simulating group action quantum walks. Assume $t = \text{poly}(\log|G|)$. We show that the continuous-time walk $W = e^{-iAt}$ can be efficiently simulated using the discrete-time quantum walk framework developed in [?, ?]. Recall the isometry T defined in Equation (5), where the states $|\phi_b\rangle$ are as given in Equation (4). To carry out the simulation, it suffices to show that the isometry T , its adjoint T^* , and the discrete-time walk operator

$$W = iS(2TT^* - \mathbb{1})$$

can all be efficiently implemented in the group-action setting.

Let $\Gamma = (X, E)$ be the Cayley graph associated with the group action $(G, X, *)$, and let A denote its adjacency matrix. The Hamiltonian H of interest is simply A , indexed by elements of X . In this context, the isometry T takes the form:

$$\begin{aligned} T &:= \sum_{y \in X} \sum_{b \in \{0,1\}} (|y\rangle\langle y| \otimes |b\rangle\langle b|) \otimes |\phi_{yb}\rangle \\ &= \sum_{y \in X} \sum_{b \in \{0,1\}} |y\rangle\langle y| \otimes |\phi_{yb}\rangle\langle b|. \end{aligned}$$

¹Continuous-time quantum walks have been explored in the context of cryptographic group actions in [?, ?], specifically in supersingular isogeny graphs. These walks can be viewed as group-action quantum walks.

Because of the structure of Γ , the states (4) simplify as follows. First, since the set Q is symmetric, the graph Γ is undirected and the adjacency matrix A is symmetric with non-negative entries; the nonzero entries of A are all equal to 1. Therefore, we can set $K = \|A\|_{\max} = 1$. As a result, for $y \in X$, the state $|\phi_{y0}\rangle$ becomes

$$|\phi_{y0}\rangle = \frac{1}{\sqrt{|Q|}} \sum_{q \in Q} |q * y\rangle |0\rangle.$$

We now describe how to implement the isometry T by constructing efficient unitaries U_0 and U_1 , where each U_b acts as

$$U_b : |0\rangle|y, b\rangle|0, 0\rangle \mapsto |0\rangle|y, b\rangle|\phi_{yb}\rangle, \quad b \in \{0, 1\},$$

with the first register serving as an ancilla.

The unitary U_1 is straightforward to implement, since $|\phi_{y1}\rangle = |0\rangle|1\rangle$ is independent of y . We now focus on the construction of U_0 .

Because $Q \subset G$ is a small set (i.e., $|Q| = \text{poly}(\log|G|)$), we can construct an efficient unitary $V_Q : \mathbb{C}^Q \rightarrow \mathbb{C}^Q$ such that

$$V_Q|0\rangle = \frac{1}{\sqrt{|Q|}} \sum_{q \in Q} |q\rangle.$$

Applying $V_Q \otimes \mathbb{1}$ to the state $|0\rangle|y, 0\rangle|0, 0\rangle$ produces:

$$\frac{1}{\sqrt{|Q|}} \sum_{q \in Q} |q\rangle|y, 0\rangle|0, 0\rangle.$$

Next, we apply the unitary V_1 , defined by:

$$V_1 : |q\rangle|y, 0\rangle|0, 0\rangle \mapsto |q\rangle|y, 0\rangle|q * y, 0\rangle.$$

Following this, we uncompute the first register to recover $|0\rangle|y, 0\rangle|\phi_{y0}\rangle$. This is achieved by applying the unitary V_2 , defined as:

$$V_2 : |q\rangle|y, 0\rangle|q * y, 0\rangle \mapsto |0\rangle|y, 0\rangle|q * y, 0\rangle.$$

Since Q is small, we can efficiently determine q from the pair $(y, q * y)$ by exhaustively checking each $q \in Q$. Thus, V_2 (and therefore U_0) can be implemented efficiently.

To implement $U_0^* : |0\rangle|y, 0\rangle|\phi_{y0}\rangle \mapsto |0\rangle|y, 0\rangle|0, 0\rangle$, we sequentially apply V_2^* , followed by V_1^* , and finally $V_Q^* \otimes \mathbb{1}$. Since each of these unitaries can be implemented efficiently, it follows that U_0^* is also efficient.

We now construct the isometry T using a conditional unitary U_T , which applies either U_0 or U_1 depending on the value of the qubit b . That is,

$$U_T = \text{Controlled-}U_b \quad \text{with control on } b \in \{0, 1\}.$$

Next, we demonstrate that the walk unitary

$$W = iS(2TT^* - \mathbb{1})$$

can be applied efficiently. This reduces to implementing the reflection $2TT^* - \mathbb{1}$, which we analyze as follows:

$$2|0\rangle\langle 0| \otimes TT^* - \mathbb{1} = 2|0\rangle\langle 0| \otimes \sum_{y \in X} \sum_{b \in \{0, 1\}} |y, b\rangle\langle y, b| |\phi_{yb}\rangle\langle \phi_{yb}| - \mathbb{1}$$

$$\begin{aligned}
&= 2 \sum_{y \in X} \sum_{b \in \{0,1\}} |0\rangle|y,b\rangle|\phi_{yb}\rangle\langle 0|\langle y,b|\langle\phi_{yb}| - \mathbb{1} \\
&= U_T \left(2 \sum_{y \in X} \sum_{b \in \{0,1\}} |0\rangle|y,b\rangle|0,0\rangle\langle 0|\langle y,b|\langle 0,0| - \mathbb{1} \right) U_T^* \\
&= U_T (2|0\rangle\langle 0| \otimes \mathbb{1}_{X,b} \otimes |0,0\rangle\langle 0,0| - \mathbb{1}) U_T^*.
\end{aligned}$$

Since U_T , U_T^* , and the operator $2|0\rangle\langle 0| \otimes \mathbb{1}_{X,b} \otimes |0,0\rangle\langle 0,0| - \mathbb{1}$ can all be implemented efficiently, it follows that the reflection $2|0\rangle\langle 0| \otimes TT^* - \mathbb{1}$ is also efficient.

Finally, for any input state $|\psi\rangle$, the reflection acts as:

$$(2|0\rangle\langle 0| \otimes TT^* - \mathbb{1}) |0\rangle|\psi\rangle = |0\rangle (2TT^* - \mathbb{1}) |\psi\rangle,$$

confirming that the full walk operator W can be implemented efficiently in the group-action setting.

7 Applications

8 Quantum Money

Introduction to Quantum Money

Quantum money is a revolutionary concept in cryptography that uses quantum mechanics to create a form of currency that is provably secure against forgery. Originally introduced by physicist Stephen Wiesner in the 1970s, quantum money represents one of the earliest proposed applications of quantum information science. By encoding information into quantum states, quantum money exploits the fundamental principles of quantum mechanics to provide security features unattainable by classical systems.

Benefits of Quantum Money

- **Unforgeable:** Classical currency, both physical and digital, can be counterfeited with enough effort and resources. Quantum money, however, is fundamentally unforgeable due to the laws of quantum physics.
- **Decentralized Verification:** In some theoretical models, quantum money can be verified without contacting the issuing authority, enabling decentralized systems for authentication.
- **Enhanced Privacy:** The unique encoding of each quantum bill could allow for privacy-preserving transactions, as the details of the transaction need not be linked to the bill's verification.

8.1 Quantum Money From Group Actions

A public-key quantum money scheme consists of two QPT algorithms:

- **Gen(1^λ):** This algorithm takes a security parameter λ as input and outputs a pair (s, ρ_s) , where s is a binary string called the serial number, and ρ_s is a quantum state called the banknote. The pair (s, ρ_s) , or simply ρ_s , is sometimes denoted by \$.

- **Ver**(s, ρ_s) This algorithm takes a serial number and an alleged banknote as input and outputs either 1 (accept) or 0 (reject).

The quantum money scheme is said to be *correct* if genuine banknotes generated by **Gen** are accepted by **Ver** with high probability. More formally:

$$\Pr[\mathbf{Ver}(s, \rho_s) = 1 : (s, \rho_s) \leftarrow \mathbf{Gen}(1^\lambda)] \geq 1 - \text{negl}(\lambda).$$

where the probability is taken over the randomness of **Gen** and **Ver**. The scheme (**Gen**, **Ver**) is said to be secure if, given a genuine bill (s, ρ_s) , no QPT algorithm \mathcal{A} can produce two (possibly entangled) bills (s, ρ_1) and (s, ρ_2) that are both accepted by **Ver** with non-negligible probability. More formally:

$$\Pr \left[\mathbf{Ver}(s, \rho_1) = \mathbf{Ver}(s, \rho_2) = 1 : \begin{smallmatrix} (s, \rho_s) \leftarrow \mathbf{Gen}(1^\lambda) \\ (\rho_1, \rho_2) \leftarrow \mathcal{A}(s, \rho_s) \end{smallmatrix} \right] \leq \text{negl}(\lambda).$$

We now briefly outline the quantum money construction from [?], which is based on abelian group actions. Let $\{(G_\lambda, X_\lambda, *)\}_{\lambda \in J}$, where $J \subset \mathbb{N}$, be a collection of cryptographic group actions for abelian groups G_λ , and let $x_\lambda \in X_\lambda$ be a fixed element. The **Gen** and **Ver** algorithms are as follows:

- **Gen**(1^λ). Begin with the state $|0\rangle|x_\lambda\rangle$, and apply the quantum Fourier transform over G_λ to the first register producing the superposition

$$\frac{1}{\sqrt{|G_\lambda|}} \sum_{g \in G_\lambda} |g\rangle|x_\lambda\rangle.$$

Next, apply the unitary transformation $|h\rangle|y\rangle \mapsto |h\rangle|h * y\rangle$ to this state, followed by the quantum Fourier transform on the first register. This results in

$$\frac{1}{|G_\lambda|} \sum_{h \in G_\lambda} \sum_{g \in G_\lambda} \chi(g, h) |h\rangle|g * x_\lambda\rangle = \frac{1}{\sqrt{|G_\lambda|}} \sum_{h \in G_\lambda} |h\rangle|G^{(h)} * x_\lambda\rangle$$

where,

$$|G^{(h)} * x_\lambda\rangle = \frac{1}{\sqrt{|G_\lambda|}} \sum_{g \in G_\lambda} \chi(g, h) |g * x_\lambda\rangle$$

. Measure the first register to obtain a random $h \in G_\lambda$, collapsing the state to $|G^{(h)} * x_\lambda\rangle$. Return the pair $(h, |G^{(h)} * x_\lambda\rangle)$.

- **Ver**($h, |\psi\rangle$). First, check whether $|\psi\rangle$ has support in X_λ . If not, return 0. Then, apply **cmplIndex** to the state $|\psi\rangle|0\rangle$, and measure the second register to obtain some $h' \in G_\lambda$. If $h' = h$, return 1; otherwise return 0.

From this point forward, to simplify the notation, we make the security parameter λ implicit, and use G for G_λ , X for X_λ , and so on.

Quantum Hartley Transform (QHT)

The **Quantum Hartley Transform (QHT)** is a linear unitary transform that operates on quantum states. It transforms the basis states $|x\rangle$ as follows:

$$|x\rangle \xrightarrow{\text{QHT}} \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \text{cas}\left(\frac{2\pi xk}{N}\right) |k\rangle$$

where:

- x and k are integers in $\{0, 1, \dots, N-1\}$,
- $\text{cas}(x) = \cos(x) + \sin(x)$,
- N is the dimension of the transform, typically 2^n for n -qubit systems.

For a quantum state $|\psi\rangle$ in an N -dimensional Hilbert space, represented as:

$$|\psi\rangle = \sum_{x=0}^{N-1} \psi_x |x\rangle,$$

the QHT transforms the state into:

$$|\psi'\rangle = \text{QHT}|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \left(\sum_{x=0}^{N-1} \psi_x \text{cas}\left(\frac{2\pi xk}{N}\right) \right) |k\rangle.$$

Properties of the QHT

1. **Unitary Transformation:** The QHT matrix U_{QHT} is unitary:

$$U_{\text{QHT}} U_{\text{QHT}}^\dagger = I,$$

preserving quantum state normalization.

2. **Symmetry:** The QHT is symmetric because $\text{cas}(x) = \text{cas}(-x)$.
3. **Efficient Implementation:** Like the Quantum Fourier Transform (QFT), the QHT can be implemented with $O(\log^2 N)$ gates for $N = 2^n$.

Efficient implementation of Hartley transform

Our algorithm for the quantum Hartley transform, QHT_N , is modeled after the recursive structure of the quantum Fourier transform algorithm QFT_N . To motivate this, we begin by reviewing the standard approach for computing QFT_N . For clarity, we assume $N = 2^n$, allowing each element of \mathbb{Z}_N to be represented using exactly n qubits. The same recursive framework can be extended to handle general values of N . For any $a \in \mathbb{Z}_N$, the quantum Fourier transform is defined as:

$$\begin{aligned} \text{QFT}_N |a\rangle &= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega_N^{ay} |y\rangle \\ &= \frac{1}{\sqrt{N}} \sum_{y=0}^{N/2-1} \omega_N^{ay} |y\rangle + (-1)^a \frac{1}{\sqrt{N}} \sum_{y=0}^{N/2-1} \omega_N^{ay} |y + N/2\rangle \\ &= \frac{1}{\sqrt{N/2}} \sum_{y=0}^{N/2-1} \omega_N^{ay} \frac{1}{\sqrt{2}} (|0\rangle + (-1)^a |1\rangle) |y\rangle, \end{aligned} \tag{7}$$

In the final expression, we have isolated the first qubit for clarity. Let $|a\rangle = |t\rangle|b\rangle$, where b denotes the least significant bit of a , implying that $a = 2t + b$ for some $t \in \mathbb{Z}_{N/2}$. Applying $\text{QFT}_{N/2}$ to the first register yields the state:

$$\frac{1}{\sqrt{N/2}} \sum_{y=0}^{N/2-1} \omega_N^{2ty} |y\rangle |b\rangle.$$

Next, we apply the phase unitary $P(y, b)$, defined by $|y\rangle |b\rangle \mapsto \omega_N^{by} |y\rangle |b\rangle$, followed by a Hadamard transform on the final qubit. This results in the state given in Equation (7).

We now describe our algorithm for efficiently computing the quantum Hartley transform QHT_N . The key idea is to leverage the recursive structure of QHT_N , analogous to the strategy used for QFT_N . To facilitate this, we begin by rewriting the summation in Equation (??) to expose its recursive form. Our approach is as follows:

$$\frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \text{cas}\left(\frac{2\pi ay}{N}\right) |y\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N/2-1} \text{cas}\left(\frac{2\pi ay}{N}\right) |y\rangle + \frac{1}{\sqrt{N}} \sum_{y=N/2}^{N-1} \text{cas}\left(\frac{2\pi ay}{N}\right) |y\rangle. \quad (8)$$

The second summation on the right-hand side can be expressed as:

$$\begin{aligned} \sum_{y=N/2}^{N-1} \text{cas}\left(\frac{2\pi ay}{N}\right) |y\rangle &= \sum_{y=0}^{N/2-1} \text{cas}\left(\frac{2\pi ay}{N} + \pi a\right) |y + N/2\rangle \\ &= (-1)^a \sum_{y=0}^{N/2-1} \text{cas}\left(\frac{2\pi ay}{N}\right) |y + N/2\rangle, \end{aligned}$$

The second equality follows from the angle-sum identity for the cas function,

$$\text{cas}(\alpha + \beta) = \cos(\alpha)\text{cas}(\beta) + \sin(\alpha)\text{cas}(-\beta),$$

along with the fact that $\cos(\pi a) = (-1)^a$ for all integers a . Substituting this into Equation (8) yields:

$$\begin{aligned} \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \text{cas}\left(\frac{2\pi ay}{N}\right) |y\rangle &= \frac{1}{\sqrt{N}} \sum_{y=0}^{N/2-1} \text{cas}\left(\frac{2\pi ay}{N}\right) (|y\rangle + (-1)^a |y + N/2\rangle) \\ &= \frac{1}{\sqrt{N/2}} \sum_{y=0}^{N/2-1} \text{cas}\left(\frac{2\pi ay}{N}\right) \frac{1}{\sqrt{2}} (|0\rangle + (-1)^a |1\rangle) |y\rangle, \end{aligned} \quad (9)$$

In the final equality, the most significant qubit has been isolated to emphasize its contribution to the overall transformation.

We now demonstrate how to compute QHT_N recursively. For a given $a \in \mathbb{Z}_N$, we decompose the basis state as $|a\rangle = |t\rangle |b\rangle$, where b denotes the least significant bit, such that $a = 2t + b$ for some $t \in \mathbb{Z}_{N/2}$. Assuming the existence of an efficient quantum circuit for computing $\text{QHT}_{N/2}$, we proceed to construct QHT_N in a recursive fashion. To begin the transformation on $|a\rangle$, we introduce an ancilla qubit initialized to $|0\rangle$, yielding the joint state $|0\rangle |t\rangle |b\rangle$. We then proceed as follows:

$$\begin{aligned} |0\rangle |t\rangle |b\rangle &\mapsto \frac{1}{\sqrt{N/2}} \sum_{y=0}^{N/2-1} \text{cas}\left(\frac{2\pi ty}{N/2}\right) |0\rangle |y\rangle |b\rangle && (\mathbf{1} \otimes \text{QHT}_{N/2} \otimes \mathbf{1}) \\ &= \frac{1}{\sqrt{N/2}} \sum_{y=0}^{N/2-1} \text{cas}\left(\frac{4\pi ty}{N}\right) |0\rangle |y\rangle |b\rangle \end{aligned}$$

$$\mapsto \frac{1}{\sqrt{N}} \sum_{y=0}^{N/2-1} \text{cas}\left(\frac{4\pi ty}{N}\right) (|0\rangle + |1\rangle) |y\rangle |b\rangle. \quad (H \otimes \mathbb{1})$$

Next, we apply a controlled negation operation, where the ancilla qubit is flipped if and only if the least significant bit b is equal to 1.

$$V : |0\rangle |y\rangle \mapsto |0\rangle |y\rangle, \quad |1\rangle |y\rangle \mapsto |1\rangle |N/2 - y\rangle,$$

$$\frac{1}{\sqrt{N}} \sum_{y=0}^{N/2-1} \text{cas}\left(\frac{4\pi ty}{N}\right) |0\rangle |y\rangle |b\rangle + \frac{1}{\sqrt{N}} \sum_{y=0}^{N/2-1} \text{cas}\left(\frac{4\pi ty}{N}\right) |1\rangle |-y\rangle |b\rangle.$$

By performing a change of variables in the second summation and using the identity $\text{cas}\left(\frac{4\pi t(N/2-y)}{N}\right) = \text{cas}\left(-\frac{4\pi ty}{N}\right)$, we obtain the state:

$$\frac{1}{\sqrt{N}} \sum_{y=0}^{N/2-1} \left(\text{cas}\left(\frac{4\pi ty}{N}\right) |0\rangle + \text{cas}\left(-\frac{4\pi ty}{N}\right) |1\rangle \right) |y\rangle |b\rangle.$$

We define the single-qubit rotation $R(\theta)$ as the unitary operator acting on a single qubit given by:

$$R(y, b) = \begin{bmatrix} \cos(2\pi by/N) & \sin(2\pi by/N) \\ -\sin(2\pi by/N) & \cos(2\pi by/N) \end{bmatrix}, \quad (10)$$

Now, consider the unitary U_R defined by

$$U_R : |c\rangle |y\rangle |b\rangle \mapsto (R(y, b)|c\rangle) |y\rangle |b\rangle,$$

where $R(y, b)$ is a single-qubit rotation depending on y and b . Applying U_R followed by the controlled negation V , we obtain the state:

$$\begin{aligned} |\phi_1\rangle &= \frac{1}{\sqrt{N}} \sum_{y=0}^{N/2-1} \left(\text{cas}\left(\frac{2\pi ay}{N}\right) |0\rangle + \text{cas}\left(-\frac{2\pi a(N/2-y)}{N}\right) |1\rangle \right) |y\rangle |b\rangle \\ &= \frac{1}{\sqrt{N}} \sum_{y=0}^{N/2-1} \text{cas}\left(\frac{2\pi ay}{N}\right) (|0\rangle + (-1)^b |1\rangle) |y\rangle |b\rangle, \end{aligned}$$

Here, we have used the identity $\text{cas}\left(-\pi a + \frac{2\pi ay}{N}\right) = (-1)^a \text{cas}\left(\frac{2\pi ay}{N}\right)$, along with the fact that $(-1)^a = (-1)^b$. Applying the Hadamard transform to the first qubit yields:

$$|\psi\rangle = \frac{1}{\sqrt{N/2}} \sum_{y=0}^{N/2-1} \text{cas}\left(\frac{2\pi ay}{N}\right) |b\rangle |y\rangle |b\rangle$$

Next, we uncompute the first qubit by applying a CNOT gate with the last qubit as control. We then apply a Hadamard transform followed by a swap operation, resulting in the state:

$$|\psi\rangle \mapsto \frac{1}{\sqrt{N/2}} \sum_{y=0}^{N/2-1} \text{cas}\left(\frac{2\pi ay}{N}\right) |0\rangle |y\rangle |b\rangle \quad (\text{CNOT})$$

$$\begin{aligned}
&\mapsto \frac{1}{\sqrt{N/2}} \sum_{y=0}^{N/2-1} \text{cas}\left(\frac{2\pi ay}{N}\right) |0\rangle|y\rangle \frac{1}{\sqrt{2}}(|0\rangle + (-1)^b|1\rangle) & (\mathbb{1} \otimes H) \\
&= \frac{1}{\sqrt{N/2}} \sum_{y=0}^{N/2-1} \text{cas}\left(\frac{2\pi ay}{N}\right) |0\rangle|y\rangle \frac{1}{\sqrt{2}}(|0\rangle + (-1)^a|1\rangle). & ((-1)^b = (-1)^a)
\end{aligned}$$

The final summation matches precisely with Equation (9), which defines the quantum Hartley transform of $|a\rangle$. Thus, we have successfully implemented the transformation:

$$\begin{aligned}
|a\rangle &\mapsto \text{QHT}_N|a\rangle. \\
|0\rangle|a\rangle &\mapsto |0\rangle\text{QHT}_N|a\rangle.
\end{aligned}$$

The complete procedure is summarized in the following algorithm.

Algorithm 1 (QHT_N).

Input: quantum state $|\psi\rangle \in \mathbb{C}^N$, where $N = 2^n$

Output: quantum state $\text{QHT}_N|\psi\rangle$

1. Initialize an ancilla qubit to 0 to obtain the state $|0\rangle|\psi\rangle$
2. Compute $\mathbb{1} \otimes \text{QHT}_{N/2} \otimes \mathbb{1}$ recursively.
3. Apply $H \otimes \mathbb{1}$.
4. Apply the controlled negation $|0\rangle|y\rangle \mapsto |0\rangle|y\rangle, |1\rangle|y\rangle \mapsto |1\rangle|N/2 - y\rangle$ to the first two registers.
5. Apply the unitary U_R .
6. Apply $H \otimes \mathbb{1}$
7. Apply CNOT to the first and last qubits.
8. Apply $\mathbb{1} \otimes H$.
9. Trace out the first qubit

Theorem 8.1. *Algorithm 1 is correct and can be implemented using $\approx \log^2 N + O(\log N)$ elementary gates.*

Proof. The correctness of the algorithm follows from the preceding discussion. Except for the unitary U_R and the negation unitary in Step 4, all steps of the algorithm can be implemented using $O(1)$ elementary gates. The negation operation in Step 4 can be realized using approximately $\lceil \log N \rceil$ elementary gates.

To implement the unitary U_R , which requires constructing the conditional rotation operator $R(y, b)$ for arbitrary y and b , we make use of the two-qubit controlled rotations:

$$R_j = |0\rangle\langle 0| \otimes \mathbb{1} + |1\rangle\langle 1| \otimes \begin{bmatrix} \cos\left(\frac{2\pi 2^j}{N}\right) & \sin\left(\frac{2\pi 2^j}{N}\right) \\ -\sin\left(\frac{2\pi 2^j}{N}\right) & \cos\left(\frac{2\pi 2^j}{N}\right) \end{bmatrix},$$

for $j = 0, 1, \dots, n-1$. When $b = 0$, we have $R(y, 0) = \mathbb{1}$, and when $b = 1$, the operator $R(y, 1)$ is the product of those R_j for which the j th bit of y is 1. Therefore, for a binary string y of length $k = \lceil \log y \rceil$, we can implement U_R using at most k gates from the set $\{R_j\}$.

Let $T(N)$ denote the gate complexity of Algorithm 1 for an input of dimension N . Assuming access to the R_j gates as elementary operations, the recurrence relation becomes

$$T(N) \approx T(N/2) + 2\log N + O(1).$$

Solving this recurrence yields $T(N) \approx \log^2 N + O(\log N)$, as claimed. \square

9 Quantum Money With The Hartley Transform

The quantum money scheme described earlier can also be instantiated using the quantum Hartley transform in place of the quantum Fourier transform. However, this substitution disrupts the original verification procedure. In the following sections, we demonstrate how quantum walks can be employed to resolve this issue. To pinpoint the source of the problem, we begin by restating the **Gen** and **Ver** algorithms for this Hartley-based scheme, mirroring the structure of the Fourier-based version but replacing QFT_G with QHT_G . For simplicity, we take $G = \mathbb{Z}_N$, and fix an element $x \in \mathbb{Z}_N$.

- **Gen**: Start with the state $|0\rangle|x\rangle$, and apply the quantum Hartley transform over \mathbb{Z}_N to the first register, resulting in the superposition

$$\frac{1}{\sqrt{N}} \sum_{g \in \mathbb{Z}_N} |g\rangle|x\rangle.$$

Next, apply the unitary transformation $|h\rangle|y\rangle \mapsto |h\rangle|h * y\rangle$, followed by a Hartley transform QHT_N on the first register. This yields the state

$$\frac{1}{N} \sum_{h \in \mathbb{Z}_N} \sum_{g \in \mathbb{Z}_N} \text{cas}\left(\frac{2\pi gh}{N}\right) |h\rangle|g * x\rangle = \frac{1}{\sqrt{N}} \sum_{h \in \mathbb{Z}_N} |h\rangle|\mathbb{Z}_N^{(h)} * x\rangle_H,$$

where

$$|\mathbb{Z}_N^{(h)} * x\rangle_H = \frac{1}{\sqrt{N}} \sum_{g \in \mathbb{Z}_N} \text{cas}\left(\frac{2\pi gh}{N}\right) |g * x\rangle.$$

Measure the first register to obtain a random $h \in \mathbb{Z}_N$, collapsing the state to $|\mathbb{Z}_N^{(h)} * x\rangle_H$. Return the pair $(h, |\mathbb{Z}_N^{(h)} * x\rangle_H)$.

- **Ver**($h, |\psi\rangle$): First, check whether $|\psi\rangle$ has support in X . If not, return 0. Otherwise, apply cmplndex to the state $|0\rangle|\psi\rangle$, and measure the first register to obtain some $h' \in \mathbb{Z}_N$. Return 1 if $h' = h$, and 0 otherwise.

Verification using quantum walks

Given a state of the form $|\mathbb{Z}_N^{(h)} * x\rangle_H$, we demonstrate how the value of h can be extracted using continuous-time quantum walks. For any $u \in \mathbb{Z}_N$, consider the Cayley graph $\Gamma = (\mathbb{Z}_N, E)$ generated by $Q = -u, u$, as introduced in Section ???. Let A denote the adjacency matrix of Γ . The eigenvectors of A are given by $|\mathbb{Z}_N^{(h)} * x\rangle$, with corresponding eigenvalues $\lambda_h = 2 \cos(2\pi uh/N)$ for each $h \in \mathbb{Z}_N$. According to Theorem 6.1 and the discussion that follows, the unitary evolution operator $W = e^{iAt}$ can be simulated efficiently to exponential precision. To proceed, we first state the following lemma.

Lemma 9.1. *The money state $|\mathbb{Z}_N^{(h)} * x\rangle_H$ is an eigenstate of W with eigenvalue $e^{i\lambda_h t}$.*

Proof. We have

$$e^{iAt} |\mathbb{Z}_N^{(h)} * x\rangle_H = \sum_{g \in \mathbb{Z}_N} e^{i\lambda_g t} |\mathbb{Z}_N^{(g)} * x\rangle \langle \mathbb{Z}_N^{(g)} * x | \mathbb{Z}_N^{(h)} * x \rangle_H$$

$$\begin{aligned}
&= \sum_{g \in \mathbb{Z}_N} e^{i\lambda_g t} |\mathbb{Z}_N^{(g)} * x\rangle \langle \mathbb{Z}_N^{(g)} * x| \left(\frac{1-i}{2} |\mathbb{Z}_N^{(h)} * x\rangle + \frac{1+i}{2} |\mathbb{Z}_N^{(-h)} * x\rangle \right) \\
&= e^{i\lambda_h t} \frac{1-i}{2} |\mathbb{Z}_N^{(h)} * x\rangle + \frac{1+i}{2} e^{i\lambda_{-h} t} |\mathbb{Z}_N^{(-h)} * x\rangle \\
&= e^{i\lambda_h t} |\mathbb{Z}_N^{(h)} * x\rangle_H,
\end{aligned}$$

where the second equality follows from the identity in (??), and the last equality follows from the fact that $\lambda_h = \lambda_{-h}$. \square

By setting $t = \text{poly}(\log N)$, Lemma 9.1 ensures that phase estimation can be performed using the unitary W and the eigenstate $|\mathbb{Z}_N^{(h)} * x\rangle_H$, yielding an approximation $\tilde{\lambda}_h$ of λ_h such that

$$|\tilde{\lambda}_h - \lambda_h| \leq \frac{1}{\text{poly}(\log N)}.$$

From this approximation, we can extract a real number $0 \leq \theta \leq 1$ satisfying

$$\left| \theta - \frac{uh}{N} \right| \leq \frac{1}{\text{poly}(\log N)}.$$

Since phase estimation can be applied with various choices of u , we can obtain multiple approximations of $\frac{uh}{N}$. As shown in [?], selecting u appropriately allows us to reconstruct h exactly from these estimates.

References