# 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

## Abstract

This thesis explores the foundations of quantum computation, focusing on quantum walks and their application to quantum money. Quantum walks, particularly continuous-time quantum walks based on group actions, serve as a powerful computational tool with applications in search algorithms and cryptographic protocols. We examine their mathematical structure and their advantages over classical random walks, emphasizing their efficiency in state evolution and probability distribution spreading. As a part of this work, we examine efficient implementations of transforms such as the Quantum Fourier Transform (QFT) and the Quantum Hartley Transform (QHT), analyzing their role in encoding quantum states for secure cryptographic applications. In particular, we discuss a novel instantiation of a quantum money scheme based on QHT, leveraging its unique properties for improved security and efficiency. To ensure the robustness of this quantum money scheme, we develop a verification mechanism utilizing quantum walks. Unlike previous approaches, which rely on standard quantum state measurements, our method employs continuous-time quantum walks to authenticate quantum money, preventing counterfeiting while maintaining computational feasibility. Additionally, we present a detailed discussion on the efficient implementation of this scheme, including optimized circuit designs and error mitigation strategies.

## 1 Introduction to Quantum Computation

Quantum Money and the Hartley Transform We now present our public-key quantum money scheme based on group actions and the Hartley transform. We begin by reviewing Zhandry's quantum money scheme from abelian group actions (which uses the Fourier transform) as a baseline. Then we describe our Hartley-transform-based variant and discuss its properties. The quantum Hartley transform will play a central role in both the minting and verification of banknotes.

## 2 Public-Key Quantum Money: Definitions

A public-key quantum money scheme consists of two algorithms, traditionally called Mint and Verify:

- $\mathsf{Mint}(1^n)$ is a probabilistic quantum polynomial-time (QPT) algorithm that, given a security parameter $n$, outputs a pair $(s, |\$\rangle)$. Here $s$ is a classical string called the *serial number* (or public key) and $|\$\rangle$ is a quantum state representing the banknote. We often call the pair $(s, |\$\rangle)$ a banknote, and $s$ by itself the serial number of that banknote.
- $\mathsf{Verify}(s, |\$\rangle)$ is a QPT algorithm (which may be implemented as a quantum circuit acting on the state $|\$\rangle$ and some ancillary workspace). It takes as input a serial number $s$ and an alleged quantum banknote state $|\$\rangle$. It outputs `accept` (valid) or `reject` (invalid).

We require two properties from such a scheme:

**Correctness:** For any valid output $(s, |\$\rangle)$ of $\mathsf{Mint}(1^n)$, the verification algorithm accepts with high probability. Formally, $\Pr[\mathsf{Verify}(s, |\$\rangle) = \texttt{accept}]$ should be very close to 1 (typically we require it to be $1 - \mathrm{negl}(n)$, where $\mathrm{negl}(n)$ denotes a negligible function in $n$).

**Security (Unforgeability):** No efficient adversary, given one or more genuine banknotes, can produce a new valid banknote. In the strongest formulation, if an adversary is given a single valid $(s, |\$\rangle)$, it should be infeasible for them to produce two quantum states $|\$'\rangle$ and $|\$''\rangle$ such that $\mathsf{Verify}(s, |\$'\rangle) = \mathsf{Verify}(s, |\$''\rangle) = \texttt{accept}$ (i.e., they cannot create a second copy that passes verification while retaining the original). More generally, even given multiple banknotes, the adversary should not be able to create new ones with fresh serial numbers that pass verification. This captures the idea that quantum money cannot be counterfeited.

In a *public-key* scheme, the verifier uses only the public information (here, the serial number and any scheme-wide public parameters) to check validity, and does not need any secret key. The bank (mint) may have a secret key to produce the states, but after minting, the bank's involvement is not needed. In our scheme, the serial number will effectively contain the information needed for public verification (similar to how a classical bill might have a number anyone can look up, except here it's used in a quantum procedure).

We next outline Zhandry's scheme, which our scheme builds upon.

## 3 Fourier-Based Quantum Money from Group Actions

Zhandry's quantum money scheme [31] is built on an abelian group action $G \curvearrowright X$. We have a family of group actions parameterized by the security parameter $n$ (so the size of $G$ grows with $n$). For simplicity, assume $|G| = N$. The scheme also fixes a particular element $x_0 \in X$ that everyone knows (a public base point in $X$).

The minting algorithm Mint in Zhandry's scheme works as follows (informally):

1. On input $1^n$, sample a random group element $g \leftarrow G$. This $g$ will serve as the serial number $s$ for the banknote.

2. Prepare two registers: the first is a group element register, initialized to $|0_G\rangle$ (an encoding of the identity element of $G$), and the second is an $X$-register initialized to $|x_0\rangle$.

3. Apply the quantum Fourier transform over $G$ to the first register. This creates a uniform superposition over $G$:
$$\frac{1}{\sqrt{N}} \sum_{h \in G} |h\rangle \otimes |x_0\rangle .$$

4. Now apply the group action in a controlled manner: for each basis state $|h\rangle$ in the first register, apply $h$'s action on the second register. This transforms the joint state into
$$\frac{1}{\sqrt{N}} \sum_{h \in G} |h\rangle \otimes |h * x_0\rangle ,$$
which is an entangled state between a superposition of group elements and their corresponding $X$-elements.

5. Apply the inverse QFT (which is the adjoint of the QFT) on the first register. This recombines the amplitudes in the first register. The result (by properties of the QFT) can be shown to be
$$\frac{1}{\sqrt{N}} \sum_{h \in G} \omega^{\langle g,\, h\rangle} |h\rangle \otimes |h * x_0\rangle ,$$
where $\omega^{\langle g,\, h\rangle}$ denotes the phase picked up that depends on $g$ and $h$. In fact, this state can be written as $|g\rangle \otimes \frac{1}{\sqrt{N}} \sum_{h \in G} |h * x_0\rangle$; essentially, the first register ends up in $|g\rangle$ (the serial number) and the second register is an equal superposition of all elements of $X$ in the orbit of $x_0$ (which is $X$ itself, since the action is transitive) but weighted by a phase related to $g$. This is the so-called group-action Fourier state.

6. The first register (which contains $|g\rangle$) is measured in the computational basis, yielding the random serial number $g$ and collapsing the second register to a state that we will denote $|\$_g\rangle$. This $|\$_g\rangle$ is the actual quantum money state corresponding to serial $g$.

7. Output the pair $(s = g,\ |\$\rangle = |\$_g\rangle)$.

In the above, a bit of algebra reveals that $|\$_g\rangle = \frac{1}{\sqrt{N}} \sum_{h \in G} \chi_g(h)\, |h * x_0\rangle$, where $\chi_g(h) = \omega^{\langle g,\, h\rangle}$ is essentially the character of $G$ indexed by $g$. In other words, $|\$_g\rangle$ is (up to global phase) the state one gets by applying the representation (character) $\chi_g$ to weight the superposition over the orbit $X$. These states for different $g$ are almost orthogonal and can be distinguished with the right measurement.

The verification algorithm $\mathsf{Verify}(g, |\$_g\rangle)$ for Zhandry's scheme goes as follows:

1. Given the purported serial number $g \in G$ and an input state $|\psi\rangle$ (which should equal $|\$_g\rangle$ if genuine), attach an auxiliary register prepared in $|x_0\rangle$.

2. Perform a controlled group action "kickback" using $g$ on the auxiliary register. Specifically, perform the unitary that maps $|y\rangle \otimes |x_0\rangle \mapsto |y\rangle \otimes |g * x_0\rangle$ for each basis element $|y\rangle$ of the first register. (This operation uses the classical $g$ to act on the second register conditioned

on the first—essentially, it multiplies the second register's state by $g$ if the first register is in an appropriate basis state. There are various ways to implement this, but conceptually it's a controlled operation using the known $g$.)

3. Now measure the second register (the auxiliary one) in the computational basis of $X$ to see if it is still $|x_0\rangle$. If the state $|\psi\rangle$ was the correct Fourier state $|\$_g\rangle$, one can show that this measurement will yield $x_0$ with high probability (constructive interference causes the $x_0$ component to be strong). If $|\psi\rangle$ was not the correct state, the measurement is unlikely to give $x_0$.

4. Accept if and only if the measurement outcome is $x_0$ (i.e. the second register returns to the base state).

The intuition is that the genuine state $|\$_g\rangle$ contains a phase that exactly cancels out the action of $g$ in the verification step, causing the auxiliary register to end up back at $x_0$. A counterfeit state, by contrast, would not generally have the right phase relationship, and the auxiliary register's state would be some superposition over $X$ that is not concentrated on $x_0$, hence likely giving a different outcome upon measurement.

Zhandry's scheme is shown to be correct (valid notes pass) and is believed to be secure under the assumption that the group action is one-way. The hardness essentially boils down to: given $x_0$ and $|\$_g\rangle$, an adversary cannot figure out a different $g'$ that would produce a second valid state $|\$_{g'}\rangle$ or replicate $|\$_g\rangle$ (because that would solve the hidden subgroup or discrete log type problem in the group action).

## 4 Hartley-Transform Money Scheme Construction

Our scheme modifies the above Fourier-based construction by utilizing the **quantum Hartley transform (QHT)** in place of the QFT. Since the Hartley transform is closely related to the Fourier transform (essentially it produces the real and imaginary parts in a single real-valued transform), one might expect this substitution to be straightforward. And indeed, for the minting procedure it largely is: we will use the QHT to create real-amplitude superposition states.

Concretely, assume again a group action $G \curvearrowright X$ with $|G| = N$. We fix the same generating set $S$ and basepoint $x_0$. The $\mathsf{Mint}_H$ (Hartley mint) algorithm:

1. Sample a random $g \leftarrow G$ to be the serial number.

2. Prepare $|0_G, x_0\rangle$ as before.

3. Apply the *quantum Hartley transform* over $G$ to the first register. The Hartley transform (on an abelian group of order $N$) is defined as:

$$\mathrm{QHT} : |h\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{k \in G} \mathrm{cas}\Big(\frac{2\pi \langle k, h\rangle}{N}\Big)|k\rangle,$$

where $\mathrm{cas}(\theta) = \cos\theta + \sin\theta$ and $\langle k, h\rangle$ denotes some pairing like an exponent or dot product (for cyclic groups this is just $kh$). In simpler terms, $\mathrm{QHT}(|0_G\rangle) = \frac{1}{\sqrt{N}} \sum_{k\in G} |k\rangle$ (because $\mathrm{cas}(0) = 1$), *the same uniform superposition as the QFT produces for* $|0\rangle$. So after this step, we again have $\frac{1}{\sqrt{N}} \sum_{h\in G} |h\rangle|x_0\rangle$, but crucially the coefficients are all *real* and positive (in fact all $1/\sqrt{N}$).

4. Perform the controlled group action by the first register on the second: $\frac{1}{\sqrt{N}}\sum_h |h\rangle|h * x_0\rangle$.

5. Apply the inverse QHT on the first register (which being its own inverse up to a constant factor, is the same operation again since Hartley is symmetric like Fourier). Now here is where things differ: after this operation, the joint state will be

$$\frac{1}{\sqrt{N}}\sum_{h\in G}\mathrm{cas}\Big(\frac{2\pi\langle g,h\rangle}{N}\Big)\,|h\rangle\otimes|h * x_0\rangle.$$

Because the Hartley transform combines what would be cosine and sine terms, this state is not as nicely factorable as in the Fourier case. In the Fourier case, we got $|g\rangle\otimes$ (orbit state). In the Hartley case, we do not get a single basis state in the first register; instead we get a superposition in the first register with real coefficients $c_h = \mathrm{cas}(2\pi\langle g,h\rangle/N)$. However, note that $c_h$ is just some real number in $[-1,1]$ depending on $g$ and $h$. Another way to express the above state is:

$$|\$\rangle = \frac{1}{\sqrt{N}}\sum_{x\in X}\Big(\frac{1}{\sqrt{N}}\sum_{h:h*x_0=x}\mathrm{cas}\Big(\frac{2\pi\langle g,h\rangle}{N}\Big)\Big)\,|x\rangle.$$

Since for a given $x\in X$, all $h$ such that $h * x_0 = x$ have the form $h = ug$ for a unique $u\in G$ (because the action is free and transitive), one can simplify the internal sum using properties of the cas function. But we need not dive into that here.

6. No measurement of the first register is performed. In fact, we do not have a nice product state separation, so we cannot simply measure one register to get $g$. Instead, the output of minting is just the quantum state $|\$\rangle$ (which is entangled between what used to be two registers) and the known classical serial number $g$ (we know what $g$ we picked).

The resulting money state $|\$\rangle$ has only real amplitudes (because the initial superposition was real and the controlled operation doesn't introduce complex numbers, and the Hartley transform yields real coefficients). In particular, in the computational basis of $X$, the state $|\$\rangle$ is some real-vector state (with both positive and negative values possible). We can think of it as a "Hartley orbit state" corresponding to $g$.

The verification algorithm in our Hartley-based scheme attempts to mirror the Fourier-based verification: it will use the given serial $g$ and try to check the state. A first idea would be: do exactly the same as Zhandry's verify (the group-action kickback and measure). However, this fails in certain cases. Specifically, the Hartley transform being real causes an ambiguity: it turns out that there exist distinct $g\neq g'$ for which the Hartley-based states $|\$\rangle_g$ and $|\$\rangle_{g'}$ are not orthogonal, and the original verification might accept a forged state that is a certain superposition. In fact, the verification algorithm might accept an illegitimate state with some probability bounded away from zero. This is the "breakdown" of the straightforward Hartley substitution that we alluded to earlier.

To address this, our verification algorithm takes a more sophisticated approach:

1. Given $(g,|\psi\rangle)$, it first uses the **quantum walk-based phase estimation** method (to be detailed in Chapter 6) to *compute the eigenphase corresponding to* $g$. In short, we will treat $|\psi\rangle$ as an eigenvector of the group action adjacency matrix with a certain eigenvalue $\lambda$ related to $g$. We run a procedure to estimate $\lambda$ to high precision.

2. From the estimated eigenvalue(s), we reconstruct a candidate group element $g^*$ that we believe generated the state. (This uses the fact that by running phase estimation for multiple carefully chosen walk durations, one can solve for $g$ as shown in [31].)

3. We check if $g^* = g$ (the serial number provided). If not, we reject, since the state does not match the serial. If yes, we accept.

This procedure essentially extracts the hidden group element from the state and compares it to the claimed serial. If an adversary tries to forge a state $|\psi\rangle$ for a known serial $g$, the only way to consistently pass verification is if $|\psi\rangle$ truly encodes the same $g$ in its eigen-spectrum. A fake note that was causing trouble for the naive Hartley verification (e.g., a state that is a mixture of two different "Hartley orbit" states) would yield an eigenphase that does not correspond to a single valid $g$, and the algorithm would detect the mismatch.

The key quantum tool enabling this verification is the continuous-time quantum walk on the group action graph and the ability to simulate it, which we prepared in Chapter 4. The walk Hamiltonian $A(X, S)$ has the genuine money state $|\$\rangle_g$ as an eigenstate with a known eigenvalue $\lambda(g)$. By running phase estimation on $e^{-iAt}$ with the state $|\psi\rangle$, we attempt to measure that eigenvalue. We repeat for a few different values of $t$ (the "twists" or different phases) to get enough information to pinpoint $g$. This approach can be viewed as adding extra "twists" to the verification: effectively, each choice of $t$ in phase estimation is like looking at the state in a different interference fringe pattern, analogous to applying different group actions as twists in the verification procedure. This is why we described the new verification as using group action twists—by varying a continuous parameter related to the group action's eigenvalues, we break the symmetry that allowed a counterfeit to slip through.

We defer the full analysis of the success probabilities and how many repetitions are needed to Chapter 6, where we describe the verification algorithm in detail. For now, the takeaway is that by using the quantum walk approach, we can reliably verify Hartley-based money states, restoring security while still enjoying the properties of real amplitudes.

## 5 Efficient Quantum Hartley Transform Implementation

Before moving on to the verification algorithm, we briefly discuss how we implement the quantum Hartley transform (QHT) efficiently, as this is an important practical aspect of our scheme. The Fourier transform is well-known to have efficient circuits; for the Hartley transform, fewer results are available, so we contributed a new circuit construction.

The QHT we need is over an abelian group, which for concreteness one can think of as $\mathbb{Z}_N$ (the general finite abelian group case can usually be reduced to a direct sum of cyclic groups, applying QHT on each cyclic component). Classical Hartley transforms have a recursive structure similar to fast Fourier transforms. In fact, there are known formulas to express a Hartley transform of size $N$ in terms of two Hartley transforms of size $N/2$ plus some additional linear operations [25]. We leveraged such structures to design a divide-and-conquer quantum circuit.

Our algorithm for QHT works roughly as follows: - If $N$ is even, we can express the length-$N$ Hartley transform in terms of two length-$N/2$ Hartley transforms plus a combination of cheap operations (additions, permutations of data, etc.). We implement this decomposition recursively as a quantum circuit. The base of the recursion is when the size is small (like $N = 2$ or 4, where the transform can be done with a constant number of gates). - We found that by carefully optimizing this recursion and using some known quantum subroutines for certain linear combinations, the overall gate count is improved compared to naive methods or previous proposals [21, 22]. In particular, our circuit avoids introducing any quantum Fourier transform internally (unlike [22] which computed Hartley via Fourier plus some adjustments), and this yields a purely real rotation-based circuit. - The resulting complexity for an $N$-point QHT is $O(N \log N)$ basic quantum gates (up to polylog factors for precision), which is on par (up to constant factors) with the complexity of a QFT circuit. We also derived explicit constants for small cases to show improvements over prior work.

Additionally, once we have an efficient QHT, we can also implement other related transforms. We demonstrate this with the example of the **quantum sine transform** (QST). The sine transform is another real transform closely related to Hartley (in fact, a sine transform can be implemented by a Hartley transform plus some pre- and post-processing). We gave a construction that uses one QHT as a subroutine to realize the QST on $N$ points. This had a similar complexity of $O(N \log N)$ gates. The broader implication is that a family of real-valued transforms (Hartley, sine, cosine, etc.) can all be efficiently performed on quantum hardware, which may have independent applications in signal processing or other quantum algorithms.

For the purposes of this thesis, the main point is: *we can implement the Hartley transform needed for our money scheme with polynomial efficiency.* Thus, switching to the Hartley transform does not introduce any prohibitive cost. All steps of minting and verification remain efficient. Having covered the construction of the money scheme and the necessary tools, we now proceed to the final piece: the detailed verification algorithm using quantum walks, and an analysis of how it validates genuine banknotes and foils counterfeit attempts.

Verification Algorithm Using Quantum Walks The Hartley-based quantum money scheme introduced in the previous chapter requires a new verification procedure to address the issues that arise from using real amplitude states. In this chapter, we describe and analyze the verification algorithm, which leverages continuous-time quantum walks on the group action graph (and their efficient simulation from Chapter 4) to extract the information needed to authenticate a banknote.

## 6 Challenges with Naïve Verification

Before detailing the new algorithm, let us briefly recap why the straightforward approach fails. In the Fourier-based scheme, verification was done by a single "kickback" operation using the claimed serial $g$, and measuring an auxiliary register. In the Hartley-based scheme, if we attempted the analogous one-step verification, we would perform the controlled-$g$ action and measure the auxiliary system. A genuine state $|\$_g\rangle$ (the Hartley money state for serial $g$) would cause some interference pattern in the auxiliary register, but unlike the Fourier case, it does not return the auxiliary to $|x_0\rangle$ deterministically. In fact, there is an ambiguity: certain superpositions of eigenstates corresponding to $g$ and $-g$ (or other group-related variants) can produce the same measurement statistics in that one-step test. This means an adversary might prepare a counterfeit state that is not a legitimate $|\$_g\rangle$ but still passes the one-step verification with non-zero probability. Essentially, the Hartley transform being real means we lost some phase information, and a single measurement cannot distinguish some mirrored states.

To overcome this, our strategy is to perform a more complete measurement of the state's "phase spectrum." Instead of just one operation and measurement, we will use the quantum walk (with Hamiltonian $A = A(X, S)$ as defined earlier) to perform a form of phase estimation.

## 7 Using Quantum Walks to Extract the Serial

The central observation is that the money state $|\$_g\rangle$ is an eigenstate of the walk's Hamiltonian $A$ on the group action Cayley graph. To see this, note that $|\$_g\rangle$ (Fourier or Hartley) lies in the subspace spanned by $\{|x\rangle : x \in X\}$ (we no longer explicitly keep the group element register, since after minting it's entangled or measured away). Consider the adjacency operator $A$ acting on a basis state $|x\rangle$. By definition,

$$A|x\rangle = \sum_{s\in S} |s * x\rangle,$$

summing over all neighbors under the generators. Now, $|\$_g\rangle$ is (up to normalization) $\sum_{h\in G} f(h)\, |h * x_0\rangle$ for some coefficient function $f(h)$. If one applies $A$ to this state:

$$A|\$_g\rangle = \sum_{s\in S}\sum_{h\in G} f(h)\, |s * (h * x_0)\rangle = \sum_{h\in G} f(h) \sum_{s\in S} |(sh) * x_0\rangle.$$

Now change variable $h' = sh$ in the inner sum. Since $s$ runs over all of $S$, $h'$ runs over $Sh = \{sh : s \in S\}$ which is exactly the set of neighbors of $h$ in the Cayley graph of $G$. For an abelian group, one can show (and it is known from spectral graph theory) that $\sum_{s\in S} |sh * x_0\rangle$ corresponds to the same state as $\lambda_g|h * x_0\rangle$ where $\lambda_g = \sum_{s\in S} \chi_g(s)$ for Fourier states, or $\lambda'_g = \sum_{s\in S} \operatorname{cas}(2\pi\langle g, s\rangle/N)$ for Hartley states. In short, $|\$_g\rangle$ is an eigenvector of $A$ with eigenvalue $\lambda'_g$ (a real number that depends on $g$ and $S$). This fact was formalized earlier: Lemma 8.2 showed that the money state is an eigenstate of $A$.

For example, if $G = \mathbb{Z}_N$ and $S = \{1, -1\}$, a Fourier-based $|\$_g\rangle$ is essentially the character $\chi_g$, and indeed $\chi_g$ is an eigenfunction of the adjacency (which is like a cosine operator) with eigenvalue $2\cos(2\pi g/N)$. The Hartley-based state corresponds to a combination of $\chi_g$ and $\chi_{-g}$ (the real and imaginary parts combined), and ends up with the same eigenvalue $2\cos(2\pi g/N)$ as well. So either way, the "frequency" $g$ maps to a specific eigenvalue $\lambda$.

The verification algorithm will do the following: it will run phase estimation on the unitary $U = e^{iA\tau}$ for some carefully chosen time $\tau$, using the state $|\psi\rangle$ as input. Phase estimation is a standard quantum algorithm that, given an eigenstate of $U$, yields an estimate of the eigenphase $2\pi\phi$ (where $e^{i2\pi\phi}$ is the eigenvalue of $U$) to some specified precision. In our case, since $A$ has eigenvalue $\lambda$ on $|\psi\rangle = |\$_g\rangle$, the unitary $e^{iA\tau}$ has eigenvalue $e^{i\lambda\tau}$. Thus phase estimation will give us an estimate of $\lambda\tau \pmod{2\pi}$. By choosing $\tau$ appropriately, we can ensure $\lambda\tau$ uniquely corresponds to $\lambda$ (within the precision we work at).

We cannot run $e^{iA\tau}$ directly, but Chapter 4 showed how to simulate it efficiently with a discrete-time algorithm. So when we say "phase estimation on $U = e^{iA\tau}$," we imply using the simulation routine as a subroutine in the phase estimation circuit. This will incur some error $\epsilon$, but we can make it negligible with enough resources.

Now, because $\lambda$ (the eigenvalue of $A$) is related to $g$, obtaining $\lambda$ in effect gives us information about $g$. However, one eigenvalue might correspond to two possible $g$ values (e.g., $\cos(2\pi g/N) = \cos(2\pi(N - g)/N)$). So a single run of phase estimation may not fully determine $g$. This is where performing the procedure for multiple different times $\tau$ (or perhaps using a slightly different Hamiltonian) helps. By getting $\lambda$ at different scales, we can solve for $g$ using the known functional form of $\lambda$ in terms of $g$.

In [31], it was shown that by using two or three carefully chosen multiples of time (or equivalently using the fact that $\lambda$ as a function of $g$ has a known form like a cosine series), one can uniquely recover $g$. In our scheme, since $S$ is known and fixed, and presumably $S$ is such that the map $g \mapsto \lambda_g = \sum_{s\in S} \chi_g(s)$ is injective up to trivial symmetries, a few phase estimates suffice. For instance, if $S = \{1, -1\}$, knowing $\cos(2\pi g/N)$ to high precision lets you determine $g$ up to the symmetry $g \leftrightarrow N - g$. But if we also had a way to break the symmetry (maybe by using an asymmetric choice of $\tau$ or an additional generator in $S$ that is not symmetric), we can get the sign too. Alternatively, we run phase estimation on a slightly modified Hamiltonian $A'$ (maybe corresponding to a different generating set that breaks symmetry) if available.

For simplicity, let's suppose two runs with different $\tau$ values are enough. The output of the verification algorithm is then determined by: - If the estimated $g^*$ from the phase(s) matches the provided serial $g$, output `accept`. - Otherwise, output `reject`.

Because a genuine $|\$_g\rangle$ is exactly an eigenstate with eigenvalue $\lambda_g$, phase estimation will return (with high probability) a number very close to $\lambda_g \tau$ modulo $2\pi$. Given the precision, we decode it to the nearest valid $\lambda_{g'}$ value, which should be $\lambda_g$. Hence we recover $g' = g$ and accept. For a counterfeit state, two things can happen: 1. The state $|\psi\rangle$ is a superposition of eigenstates (not a single eigenstate). In that case, phase estimation will give some distribution of eigenvalues, effectively picking one at random (with probabilities proportional to the projection of $|\psi\rangle$ onto those eigencomponents). It is highly unlikely that this random outcome consistently points to the same $g^*$ that the adversary wants (especially if they try to match a specific $g$ that they announced as serial). In fact, unless the counterfeit state was itself aligned with one particular eigenvalue, the measurement will likely produce a result that does not correspond to the claimed serial, leading to rejection. 2. The state $|\psi\rangle$ happens to be an eigenstate, but with a wrong eigenvalue (i.e., corresponding to some $g' \neq g$). In that case, the phase estimation will reliably give $\lambda_{g'}$, and we will decode $g'$ which won't match the claimed $g$, so we reject. This scenario would occur if the adversary somehow prepared a valid-looking state for a different serial $g'$ but presented it with serial $g$—clearly that should be rejected.

Thus, the only way to pass is to present a state that is an eigenstate with eigenvalue matching the serial. But the only known way to produce such a state is essentially to follow the minting procedure (due to the one-wayness of the group action, they can't derive $|\$_g\rangle$ without knowing $g$ or breaking the assumption).

One might wonder: phase estimation is a probabilistic algorithm; what if it sometimes yields the correct $g^*$ and sometimes not, could an adversary repeat the verification or something? In practice, the banknote verification either passes or fails once; an adversary doesn't get to try multiple times on the same note without potentially decohering it or being caught. We design our parameters such that the completeness (accepting a good note) is very high and the soundness (accepting a forgery) is extremely low, say negligible in $n$. This might involve repeating the phase estimation a constant number of times to amplify confidence, or using a high precision so that the error probability is negligible.

Finally, our algorithm implicitly uses the ability to perform the controlled-$U$ operations for phase estimation. That is non-trivial but our quantum walk simulation provided a way to implement $e^{iA\tau}$, and controlled versions can be constructed by controlling each gate in the simulation (which increases complexity by at most a factor poly(n)). We have to ensure that the simulation error is accounted for in the phase estimation accuracy—this can be handled by choosing simulation parameters fine enough that the error is smaller than the inverse of our runtime or something along those lines.

## 8 Detailed Verification Procedure

To summarize, here is a more step-by-step description of the verification algorithm for a banknote $(g, |\psi\rangle)$:

1. **Preparation:** Determine a set of one or more time parameters $\{\tau_1, \tau_2, \ldots, \tau_m\}$ to be used for phase estimation. These might be hardcoded or depend on $N$ and $S$. For example, $\tau_1$ could be a basic time and $\tau_2$ another that breaks degeneracy.

2. **Phase Estimation Rounds:** For each $\tau_j$:

(a) Use the quantum walk simulation circuit to implement controlled-$e^{iA\tau_j}$ on $|\psi\rangle$. (In practice, we append an ancilla register to accumulate the phase, as in standard phase estimation algorithms. We prepare a uniform superposition over some range of time steps, apply controlled-$U$ gates, and perform inverse Fourier transform on the phase register to get an estimate. This is the standard routine.)

(b) Measure the phase register to obtain an estimate $\tilde{\phi}_j$ of $\lambda_g \tau_j/(2\pi)$ (mod 1). Multiply by $2\pi/\tau_j$ to get an estimate $\tilde{\lambda}_j$ of $\lambda_g$.

3. **Compute Serial Candidate:** From the collection $\{\tilde{\lambda}_j\}$, solve for the group element $g^*$ that would produce those eigenvalues. This might involve inverting the known relation $\lambda = \sum_{s\in S} \mathrm{cas}(2\pi\langle g, s\rangle/N)$ or whatever the formula is in our context. Because of estimation error, we choose the nearest matching $g^*$ in $G$ that would give eigenvalues close to the $\tilde{\lambda}_j$ observed.

4. **Accept/Reject:** If $g^* = g$ (and all rounds of phase estimation yielded results consistent with a single $g^*$, which should happen for a valid note), then output `accept`. Otherwise, output `reject`.

This algorithm may seem complex, but it can be optimized. In practice, we could combine multiple phase estimation steps into one larger quantum circuit that extracts $g$ in one go, but conceptually, it's easier to think of them separately.

## 9 Analysis of Security and Efficiency

The new verification algorithm is more involved than the original Fourier-based one, so we must check that it is still efficient and that it indeed thwarts counterfeiting.

**Efficiency:** Each phase estimation round uses polynomial-time quantum operations. The Hamiltonian simulation of $e^{iA\tau}$ for sparse $A$ on $N$-dimensional space can be done in poly$(\log N)$ time (since queries to the group action oracle are $O(1)$ and $|S|$ is small). The phase estimation itself might need $O(\log N)$ qubits of precision to distinguish different $g$ values, and $O(\log N)$ repetitions of $U$ in superposition. Overall, each round is poly$(n)$ time. A constant number of rounds doesn't change the polynomial class. So yes, verification is QPT. In fact, in asymptotic terms, if minting took $\tilde{O}(n^c)$ time, verification might take $\tilde{O}(n^{c+1})$ or similar due to extra overhead, but still polynomial.

**Completeness:** A genuine banknote state $|\$_g\rangle$ will be accepted with high probability. The state is an eigenstate of $A$, so phase estimation yields exactly $\lambda_g$ (with small error $\epsilon$). Decoding that yields $g^* = g$. The only possible errors are from phase estimation not being perfect: there is a small probability it gives the wrong eigenvalue estimate (exponentially small in the number of ancilla qubits used). By choosing parameters, we ensure this error is negligible, say $< 2^{-n}$ or similar. So the chance a real note is mistakenly rejected is negligible.

**Soundness:** A counterfeiter's goal is to have $\mathsf{Verify}(g, |\psi\rangle) = \texttt{accept}$ for some $|\psi\rangle$ that they created without running $\mathsf{Mint}$. Essentially, they either try to reuse a state they already saw or combine states to get two accepted notes, etc. If they attempt to create a new state for an unused serial $g$, they face the following: - If $|\psi\rangle$ is not an eigenstate of $A$, when we run phase estimation, the outcome is a random eigenvalue from its spectral decomposition. It would be a huge coincidence if that random eigenvalue corresponded to exactly the claimed $g$. And even if by some luck one phase estimation round gave a favorable result, with two rounds the chance to consistently impersonate the same $g$ is even smaller. In fact, by the union bound, the probability that a wrong state passes all checks should be negligible. A rigorous argument would be that if $|\psi\rangle$ has overlap on eigenstates $\{|e_j\rangle\}$ with eigenvalues $\lambda_j$, then the verification accepts only if for all $j$ with nonzero overlap, the decoded $g_j$ equals the claimed $g$. If any overlap component has a different underlying group element, that component will cause failure in at least one of the phase estimation rounds with high probability. Unless the state was aligned to one particular $g'$, in which case the best they can do is claim it is $g'$. - If $|\psi\rangle$ *is* an eigenstate but for $g' \neq g$, then our algorithm will find $g'$ and compare to $g$ and reject. So the adversary would have to guess the serial number's eigenstate perfectly, which means they basically guessed the bank's secret or solved the one-way problem.

The most cunning forgery could be to use a genuine note $(g, |\$_g\rangle)$ and try to produce a second independent state $|\$_g\rangle$ for the same $g$. But that is essentially copying a quantum state which is unknown to them (they only know $g$ classically). Given $g$, can they produce $|\$_g\rangle$? That's the one-way group action problem: producing $|\$_g\rangle$ is as hard as computing $g * x_0$ for all group elements in superposition, which requires knowledge of $g$ in a way that they don't have (they know $g$ but to create the superposition exactly with correct phases is akin to solving for the hidden phase structure—likely hard without the trapdoor). In fact, if they could do that efficiently for any given $g$, they would break the scheme by minting their own money. So by assumption, that's infeasible.

Thus, the only resource the adversary has is perhaps other genuine notes they possess. Could they entangle two notes or something to produce two valid ones? If they have one note, trying to make two will likely destroy the original or produce two imperfect copies that fail verification (no-cloning strikes again). If they have two different notes $(g_1, |\$_{g_1}\rangle)$ and $(g_2, |\$_{g_2}\rangle)$, those states are independent, and forging a new one still reduces to above.

Therefore, we argue the scheme remains secure: forging in any meaningful sense would require solving a hard problem (inverting the group action or cloning a quantum state).

In conclusion, by using continuous-time quantum walks and phase estimation, our verification algorithm retrieves the hidden serial number encoded in the quantum money state and compares it to the claimed serial. This not only fixes the issue introduced by using the Hartley transform (real amplitudes), but it does so in polynomial time. We have thus achieved a public-key quantum money scheme based on standard assumptions (one-way group actions) with an extra novel feature of using real-valued quantum states and demonstrating the power of quantum walks in quantum cryptanalysis and verification.

## 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).
- $\alpha$ and $\beta$ 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:

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \tag{2}$$

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:

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

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:**

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

### 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.

## 10 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 $\gamma$, uniform across all vertices and time.

To model this mathematically, the probability of transitioning from one vertex to another in an infinitesimal time interval $\Delta t$ 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.

### Example: Continuous-Time Quantum Walk on the Line

To illustrate the behavior of continuous-time quantum walks, consider the example of a particle evolving on the *infinite one-dimensional lattice*, also known as the *integer line*. In this graph, the vertices are labeled by integers $\mathbb{Z}$, and each vertex $n \in \mathbb{Z}$ is connected to its immediate neighbors $n-1$ and $n+1$. The adjacency matrix $A$ for this graph has entries:

$$A_{i,j} = \begin{cases} 1 & \text{if } |i-j| = 1, \\ 0 & \text{otherwise.} \end{cases}$$

Let the walker initially be located at the origin, so the initial state is $|\psi(0)\rangle = |0\rangle$. The quantum evolution of the system is governed by the Schrödinger equation:

$$i\frac{d}{dt}|\psi(t)\rangle = A|\psi(t)\rangle,$$

with the solution:

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

For this specific graph, the amplitude of being at position $n$ at time $t$ has an elegant closed-form expression:

$$\psi_n(t) = i^n J_n(2t),$$

where $J_n$ is the *Bessel function of the first kind* of order $n$. The corresponding probability of finding the walker at vertex $n$ at time $t$ is:

$$p_n(t) = |\psi_n(t)|^2 = |J_n(2t)|^2.$$

This probability distribution displays quintessential quantum behavior: it is **non-Gaussian**, **oscillatory**, and **spreads faster** than in the classical case. In contrast to the diffusive behavior of classical walks, where the standard deviation grows as $\sigma(t) \propto \sqrt{t}$, the quantum walk exhibits *ballistic spread* with $\sigma(t) \propto t$.

These interference-driven properties make CTQWs particularly powerful for applications in quantum information and quantum algorithms, where the speed and spread of information are crucial.

## Discrete-Time Quantum Walks

Discrete-time quantum walks (DTQWs) serve as the discrete counterpart of quantum random walks and are particularly useful for algorithmic constructions in quantum computing. Unlike continuous-time quantum walks, DTQWs require an additional coin degree of freedom to preserve unitarity in the evolution. In this section, we explore the discrete-time coined quantum walk on the infinite one-dimensional lattice (integer line), which provides a clear and tractable example of how interference and superposition govern quantum behavior.

### Hilbert Space and Coin Space

The state space of the system is the tensor product $\mathcal{H} = \mathcal{H}_C \otimes \mathcal{H}_P$, where:

- $\mathcal{H}_C$ is the 2-dimensional Hilbert space of the quantum coin, with basis states $\{|0\rangle, |1\rangle\}$, often interpreted as “spin up” and “spin down”.
- $\mathcal{H}_P$ is the position Hilbert space, spanned by $\{|n\rangle : n \in \mathbb{Z}\}$, representing the particle’s location on the integer line.

### Coin and Shift Operators

The walk consists of two operations applied in sequence:

1. **Coin operator** $C$: A unitary operator acting on $\mathcal{H}_C$, typically chosen to create superposition. A common choice is the *Hadamard coin*:
$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$
2. **Shift operator** $S$: A unitary operator on $\mathcal{H}_C \otimes \mathcal{H}_P$ that moves the walker depending on the coin state:
$$S|0\rangle|n\rangle = |0\rangle|n+1\rangle, \quad S|1\rangle|n\rangle = |1\rangle|n-1\rangle.$$

One full step of the walk is implemented by applying the unitary operator:

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

where $I_P$ is the identity operator on $\mathcal{H}_P$. Repeated application of $U$ evolves the quantum state over time:

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

**Example: Hadamard Walk from the Origin**

Suppose the initial state of the system is localized at the origin with coin state $|0\rangle$:

$$|\psi(0)\rangle = |0\rangle \otimes |0\rangle.$$

Applying the Hadamard operator and then the shift, the system evolves after one step into a superposition:

$$|\psi(1)\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle|1\rangle + |1\rangle|-1\rangle\right).$$

The particle is now in a coherent superposition of being at positions $+1$ and $-1$, determined by its coin state.

This process continues iteratively. For example, the second and third steps yield:

$$\begin{aligned}
|\psi(2)\rangle &= \frac{1}{2}\left(-|1\rangle|-2\rangle + (|0\rangle + |1\rangle)|0\rangle + |0\rangle|2\rangle\right),\\
|\psi(3)\rangle &= \frac{1}{2\sqrt{2}}\left(|1\rangle|-3\rangle - |0\rangle|-1\rangle + (2|0\rangle + |1\rangle)|1\rangle + |0\rangle|3\rangle\right).
\end{aligned}$$

**Quantum vs Classical Behavior**

The key difference between quantum and classical random walks lies in the presence of *interference* and *superposition*:

- In classical random walks, probabilities are updated step-by-step based on stochastic transitions, leading to a binomial distribution that approximates a Gaussian as time progresses.
- In contrast, discrete-time quantum walks generate probability amplitudes that can interfere constructively or destructively depending on the history of the evolution.

As a result, the probability distribution of the DTQW is not symmetric or Gaussian, and it spreads much faster than its classical counterpart. Specifically, the standard deviation of a DTQW grows linearly with time $\sigma(t) \sim t$, while in the classical case it grows as $\sqrt{t}$.

If a position measurement is performed after every step, the quantum walk collapses into a classical random walk due to decoherence. However, by deferring measurement and allowing the system to evolve unitarily over many steps, quantum correlations between paths persist, yielding a rich interference pattern in the final probability distribution.

## 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.

## 11 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 $\cdot : G \times X \to X$ 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 $\phi : G \to \mathrm{Sym}(X)$, where $\mathrm{Sym}(X)$ 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 $\phi$ 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:
$$\mathrm{Orb}(x) = \{g \cdot x \mid g \in G\}.$$
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:
$$\mathrm{Stab}_G(x) = \{g \in G \mid g \cdot x = x\}.$$
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:
$$|G| = |\mathrm{Orb}(x)| \cdot |\mathrm{Stab}_G(x)|.$$
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:
$$\mathrm{Fix}(G) = \{x \in X \mid g \cdot x = x,\, \forall g \in G\}.$$
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

# 12 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.

### 12.1 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.

### 12.2 Characters of Abelian Groups

Let $G$ be an abelian group. A function
$$\chi : G \to \mathbb{C}^\times$$
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 $\chi$ 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 $\chi$ 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 $\chi_1$, 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 $\chi_1$, and the **inverse** of a character $\chi_k$ 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**:

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

## 13 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.

# 14 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**.

## 14.1 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:

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

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.

## 14.2 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:

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

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

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

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 $Q$ 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 $Q$. The Cayley graph $\Gamma(G, Q)$ 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 $gq$ for each $g \in G$ and $q \in Q$. If the generating set $Q$ is symmetric, meaning that whenever $q \in Q$, its inverse $q^{-1}$ is also in $Q$, then the Cayley graph is undirected; otherwise, it is a directed graph. For example, For the cyclic group $G = \mathbb{Z}_n = \{0, 1, 2, \ldots, n-1\}$ with the generating set $Q = \{\pm 1\}$, the Cayley graph forms a cycle graph. In this graph, each vertex $j$ is connected to $j + 1 \mod n$ and $j - 1 \mod n$, creating a cyclic structure.

The adjacency matrix of $\Gamma$ 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 $\lambda$ 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

$$A|\hat{a}\rangle = A.\frac{1}{\sqrt{|G|}} \sum_{y \in G} \chi(a, y)|y\rangle = \frac{1}{\sqrt{|G|}} \sum_{y \in G} \chi(a, y).A|y\rangle$$

$$= \frac{1}{\sqrt{|G|}} \sum_{y \in G} \chi(a, y). \sum_{q \in Q} |qy\rangle$$

Consider $\beta = qy$. Then:

$$= \frac{1}{\sqrt{|G|}} \sum_{q \in Q} \chi(a, q) \sum_{\beta} \chi(a, \beta) |\beta\rangle = \sum_{q \in Q} \chi(a, q).|\hat{a}\rangle$$

$$= \lambda_a |\hat{a}\rangle$$

## 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:

$$|\psi\rangle = \sum_{g \in G} \sum_{q \in Q} \alpha_{g,q} |g\rangle \otimes |q\rangle,$$

where $|g\rangle$ is the vertex state and $|q\rangle$ 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:

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

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.

## 14.3 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:

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

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 $T : \mathbb{C}^{2N} \to \mathbb{C}^{2N} \otimes \mathbb{C}^{2N}$ as follows:

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

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 14.1** ([**?**, Theorem 1])**.** *Let $H$ be a $d$-sparse Hamiltonian acting on $n$ qubits. Then the unitary $e^{-iHt}$ can be approximated to within error $\epsilon$ 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 $Q = \{q_1, q_2, \ldots, q_k\} \subset G$ 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 $\Gamma$ 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 $\lambda_a$ 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 $Q = \{q_1, q_2, \ldots, q_k\} \subset G$ 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:

$$A|G^{(h)} * x\rangle = \frac{1}{\sqrt{|G|}} . \sum_{g \in G} \chi(g, h) A . |g * x\rangle$$

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

Consider $\beta = qg$. Then:

$$== \frac{1}{\sqrt{|G|}} \sum_{q\in Q} \sum_{\beta\in G} \chi(\beta,h)\chi(q,h)|\beta * x\rangle$$

$$= \sum_{q\in Q} \chi(q,h).|G^{(h)} * x\rangle = \lambda_h |G^{(h)} * x\rangle$$

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 $\phi$ 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., $|Q| = \mathrm{poly}(\log|G|)$. 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:

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

where $\mathsf{QFT}_G$ and its inverse can be implemented in $\mathrm{poly}(\log|G|)$ time. Moreover, the diagonal phase unitary $|a\rangle \mapsto e^{-i\lambda_a t}|a\rangle$ is also efficient to implement, since the eigenvalues $\lambda_a$ can be computed classically to arbitrary precision in $\mathrm{poly}(\log|G|)$ 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 (**??**) 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 $t = \mathrm{poly}(\log|G|)$, the walk $e^{-iAt}$ can still be efficiently approximated to polynomial accuracy.

[1]Continuous-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.

**Simulating group action quantum walks.** Assume $t = \mathrm{poly}(\log|G|)$. 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 (**??**), where the states $|\phi_b\rangle$ are as given in Equation (**??**). 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:

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

Because of the structure of $\Gamma$, the states (**??**) simplify as follows. First, since the set $Q$ is symmetric, the graph $\Gamma$ 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., $|Q| = \mathrm{poly}(\log|G|)$), we can construct an efficient unitary $V_Q : \mathbb{C}^Q \to \mathbb{C}^Q$ 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:

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

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.

## 15 Applicaions

## 16 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.

## How Quantum Money Works

At its core, quantum money relies on two key principles of quantum mechanics: the *no-cloning theorem* and the *observer effect*.

### Encoding Information in Quantum States

- Each quantum bill contains a unique quantum state, such as a set of qubits encoded in superposition.
- These states are generated and stored by the issuing authority (e.g., a central bank) using a secret algorithm.

### Verification Process

- The issuing authority also generates a verification protocol, allowing a legitimate quantum bill to be authenticated.
- When a user presents a quantum bill for verification, the authority measures the encoded quantum states using the pre-determined protocol.
- If the measured states align with the expected values, the bill is deemed valid.

### Unforgeability

- Due to the *no-cloning theorem*, it is impossible to copy an unknown quantum state without altering it.
- Any attempt to measure or duplicate the quantum state results in a disturbance detectable during verification.

## Key Principles Underpinning Quantum Money

1. **No-Cloning Theorem:** The no-cloning theorem states that an unknown quantum state cannot be perfectly copied. This makes quantum money inherently secure, as counterfeiters cannot reproduce the quantum states encoded in legitimate currency.
2. **Measurement Disturbance:** Observing or measuring a quantum state generally alters it. This ensures that any unauthorized attempt to inspect the quantum money will render it invalid, as the encoded states will no longer match their original form.
3. **Randomness and Superposition:** Quantum money utilizes superposition to encode information. For example, a single qubit in superposition may represent both 0 and 1 simultaneously until measured. The randomness of these states makes predicting or reproducing them without knowledge of the original encoding impossible.
4. **Entanglement (Optional Feature):** Some implementations of quantum money involve quantum entanglement, where pairs of quantum states are interconnected. Changes to one entangled state directly affect its pair, adding another layer of security against forgery.

## 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.

### 16.1 Quantum Money From Group Actions

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

- $\mathsf{Gen}(1^\lambda)$: This algorithm takes a security parameter $\lambda$ as input and outputs a pair $(s, \rho_s)$, where $s$ is a binary string called the serial number, and $\rho_s$ is a quantum state called the banknote. The pair $(s, \rho_s)$, or simply $\rho_s$, is sometimes denoted by \$.
- $\mathsf{Ver}(s, \rho_s)$ 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 $\mathsf{Gen}$ are accepted by $\mathsf{Ver}$ with high probability. More formally:

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

where the probability is taken over the randomness of $\mathsf{Gen}$ and $\mathsf{Ver}$. The scheme $(\mathsf{Gen}, \mathsf{Ver})$ is said to be secure if, given a genuine bill $(s, \rho_s)$, no QPT algorithm $\mathcal{A}$ can produce two (possibly entangled) bills $(s, \rho_1)$ and $(s, \rho_2)$ that are both accepted by $\mathsf{Ver}$ with non-negligible probability. More formally:

$$\Pr\left[\mathsf{Ver}(s, \rho_1) = \mathsf{Ver}(s, \rho_2) = 1 : \substack{(s,\rho_s) \leftarrow \mathsf{Gen}(1^\lambda) \\ (\rho_1,\rho_2) \leftarrow \mathcal{A}(s,\rho_s)}\right] \leq \mathrm{negl}(\lambda).$$

We now briefly outline the quantum money construction from [**?**], which is based on abelian group actions. We will the $\mathsf{cmpIndex}$ algorithm:

**The $\mathsf{cmpIndex}$ Algorithm.** Given access to the state $|G^{(h)} * x\rangle$, one can efficiently recover the value of $h$ via a quantum algorithm. In particular, there exists a unitary transformation that maps

$$|G^{(h)} * x\rangle|0\rangle \mapsto |G^{(h)} * x\rangle|h\rangle,$$

leveraging the technique of *phase kickback*. The process proceeds as follows:

Begin with the state $|G^{(h)} * x\rangle|0\rangle$ and apply the quantum Fourier transform (QFT) to the second register. Then apply the unitary operator

$$\sum_{k \in G} U_k \otimes |k\rangle\langle k|$$

to both registers. This operation encodes the character information into the phase, resulting in the state

$$\frac{1}{\sqrt{|G|}} \sum_{k \in G} |G^{(h)} * x\rangle \chi(-k, h) |k\rangle.$$

Finally, applying the inverse QFT to the second register disentangles the phase and reveals $h$, yielding the final state $|G^{(h)} * x\rangle|h\rangle$.

Let $\{(G_\lambda, X_\lambda, *)\}_{\lambda \in J}$, where $J \subset \mathbb{N}$, be a collection of cryptographic group actions for abelian groups $G_\lambda$, and let $x_\lambda \in X_\lambda$ be a fixed element. The $\mathsf{Gen}$ and $\mathsf{Ver}$ algorithms are as follows:

- $\mathsf{Gen}(1^\lambda)$. Begin with the state $|0\rangle|x_\lambda\rangle$, and apply the quantum Fourier transform over $G_\lambda$ 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 $(h, |G^{(h)} * x_\lambda\rangle)$.

- $\mathsf{Ver}(h, |\psi\rangle)$. First, check whether $|\psi\rangle$ has support in $X_\lambda$. If not, return 0. Then, apply $\mathsf{cmplndex}$ 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 $\lambda$ implicit, and use $G$ for $G_\lambda$, $X$ for $X_\lambda$, 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:

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

where:

- $x$ and $k$ are integers in $\{0, 1, \ldots, N-1\}$,
- $\operatorname{cas}(x) = \cos(x) + \sin(x)$,
- $N$ 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:

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

## Properties of the QHT

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

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

   preserving quantum state normalization.

2. **Symmetry**: The QHT is symmetric because $\operatorname{cas}(x) = \operatorname{cas}(-x)$.

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 Heartly transform

Our algorithm for the quantum Hartley transform, $\mathsf{QHT}_N$, is modeled after the recursive structure of the quantum Fourier transform algorithm $\mathsf{QFT}_N$. To motivate this, we begin by reviewing the standard approach for computing $\mathsf{QFT}_N$. 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:

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

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 $\mathsf{QFT}_{N/2}$ 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 (**??**).

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

$$\frac{1}{\sqrt{N}}\sum_{y=0}^{N-1}\operatorname{cas}\Big(\frac{2\pi ay}{N}\Big)|y\rangle = \frac{1}{\sqrt{N}}\sum_{y=0}^{N/2-1}\operatorname{cas}\Big(\frac{2\pi ay}{N}\Big)|y\rangle + \frac{1}{\sqrt{N}}\sum_{y=N/2}^{N-1}\operatorname{cas}\Big(\frac{2\pi ay}{N}\Big)|y\rangle. \tag{8}$$

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

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

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

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

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

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

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 $\mathsf{QHT}_N$ 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 $\mathsf{QHT}_{N/2}$, we proceed to construct $\mathsf{QHT}_N$ 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:

$$\begin{aligned}|0\rangle|t\rangle|b\rangle &\mapsto \frac{1}{\sqrt{N/2}}\sum_{y=0}^{N/2-1}\operatorname{cas}\Big(\frac{2\pi ty}{N/2}\Big)|0\rangle|y\rangle|b\rangle && (\mathbb{1}\otimes\mathsf{QHT}_{N/2}\otimes\mathbb{1}) \\ &= \frac{1}{\sqrt{N/2}}\sum_{y=0}^{N/2-1}\operatorname{cas}\Big(\frac{4\pi ty}{N}\Big)|0\rangle|y\rangle|b\rangle \\ &\mapsto \frac{1}{\sqrt{N}}\sum_{y=0}^{N/2-1}\operatorname{cas}\Big(\frac{4\pi ty}{N}\Big)(|0\rangle+|1\rangle)|y\rangle|b\rangle. && (H\otimes\mathbb{1})\end{aligned}$$

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.

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

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

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

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

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

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

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:

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

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

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

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:

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

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

$$|a\rangle \mapsto \mathsf{QHT}_N|a\rangle.$$

$$|0\rangle|a\rangle \mapsto |0\rangle\mathsf{QHT}_N|a\rangle.$$

The complete procedure is summarized in the following algorithm.

**Algorithm 1** ($\mathsf{QHT}_N$)**.**
*Input:* quantum state $|\psi\rangle \in \mathbb{C}^N$, where $N = 2^n$
*Output:* quantum state $\mathsf{QHT}_N|\psi\rangle$

1. Initialize an ancilla qubit to 0 to obtain the state $|0\rangle|\psi\rangle$
2. Compute $\mathbb{1} \otimes \mathsf{QHT}_{N/2} \otimes \mathbb{1}$ 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 16.1.** *Algorithm* **??** *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 **??**, all steps of the algorithm can be implemented using $O(1)$ elementary gates. The negation operation in Step **??** 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 $j = 0, 1, \ldots, n-1$. 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 **??** 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. □

## 17 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 $\mathsf{QFT}_G$ with $\mathsf{QHT}_G$. 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 $\mathsf{QHT}_N$ on the first register. This yields the state

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

  where

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

  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 $(h, |\mathbb{Z}_N^{(h)} * x\rangle_H)$.

- $\mathsf{Ver}(h, |\psi\rangle)$: First, check whether $|\psi\rangle$ has support in $X$. If not, return 0. Otherwise, apply cmpIndex 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 $Q = \{-u, u\}$. Let $A$ denote the adjacency matrix of $\Gamma$. 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 **??** 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 17.1.** *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

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

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

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 $t = \mathrm{poly}(\log N)$, Lemma **??** 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 $\lambda_h$ such that

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

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

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

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

[1] Dominic W Berry, Andrew M Childs, and Robin Kothari. Hamiltonian simulation with nearly optimal dependence on all parameters. In *2015 IEEE 56th annual symposium on foundations of computer science*, pages 792–809. IEEE, 2015.

[2] Jeremy Booher, Ross Bowden, Javad Doliskani, Tako Boris Fouotsa, Steven D Galbraith, Sabrina Kunzweiler, Simon-Philipp Merz, Christophe Petit, Benjamin Smith, Katherine E Stange, et al. Failing to hash into supersingular isogeny graphs. *The Computer Journal*, 67(8):2702–2719, 2024.

[3] Andrew M Childs. On the relationship between continuous-and discrete-time quantum walk. *Communications in Mathematical Physics*, 294:581–603, 2010.

[4] Javad Doliskani. How to sample from the limiting distribution of a continuous-time quantum walk. *IEEE Transactions on Information Theory*, 69(11):7149–7159, 2023.

[5] Mark Zhandry. Quantum money from abelian group actions. In *15th Innovations in Theoretical Computer Science Conference (ITCS 2024)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2024.