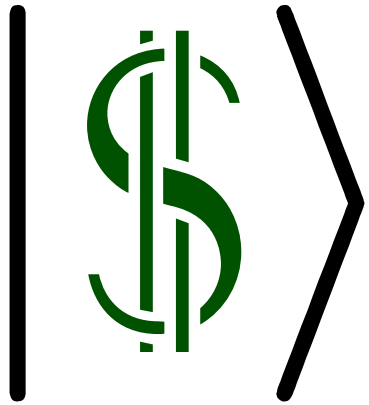
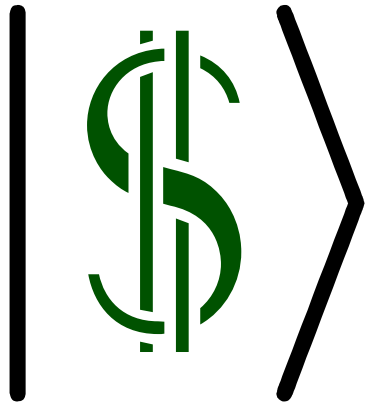
Quantum Money



Peter W. Shor
M.I.T., Cambridge, MA, U.S.A.

Quantum Money



Peter W. Shor
M.I.T., Cambridge, MA, U.S.A.

Grants:
NSF CCF-1525130,
NSF CCF-1729369 (EPIQC Collaborative Research)
NSF CCF-0939370 (STC on Science of Information)
ARO Grant W911NF-17-1-0433

Outline:

- Quantum Money: Motivation and History
- Background
- Lattice money (work in progress)

Motivation

One problem with money is that you can make copies.

Motivation

One problem with money is that you can make copies.

Quantum states satisfy the no-cloning theorem, which says you cannot make a copy of an unknown quantum state.

Motivation

One problem with money is that you can make copies.

Quantum states satisfy the no-cloning theorem, which says you cannot make a copy of an unknown quantum state.

One might think this will immediately let us use quantum states for money.
This was Wiesner's idea [1983; original manuscript ca. 1969]

Motivation

One problem with money is that you can make copies.

Quantum states satisfy the no-cloning theorem, which says you cannot make a copy of an unknown quantum state.

One might think this will immediately let us use quantum states for money.
This was Wiesner's idea [1983; original manuscript ca. 1969]
His scheme had some drawbacks.
It's quite a bit harder to come up with a quantum money system that doesn't have severe drawbacks.

Motivation

One problem with money is that you can make copies.

Quantum states satisfy the no-cloning theorem, which says you cannot make a copy of an unknown quantum state.

One might think this will immediately let us use quantum states for money.
This was Wiesner's idea [1983; original manuscript ca. 1969]
His scheme had some drawbacks.
It's quite a bit harder to come up with a quantum money system that doesn't have severe drawbacks.

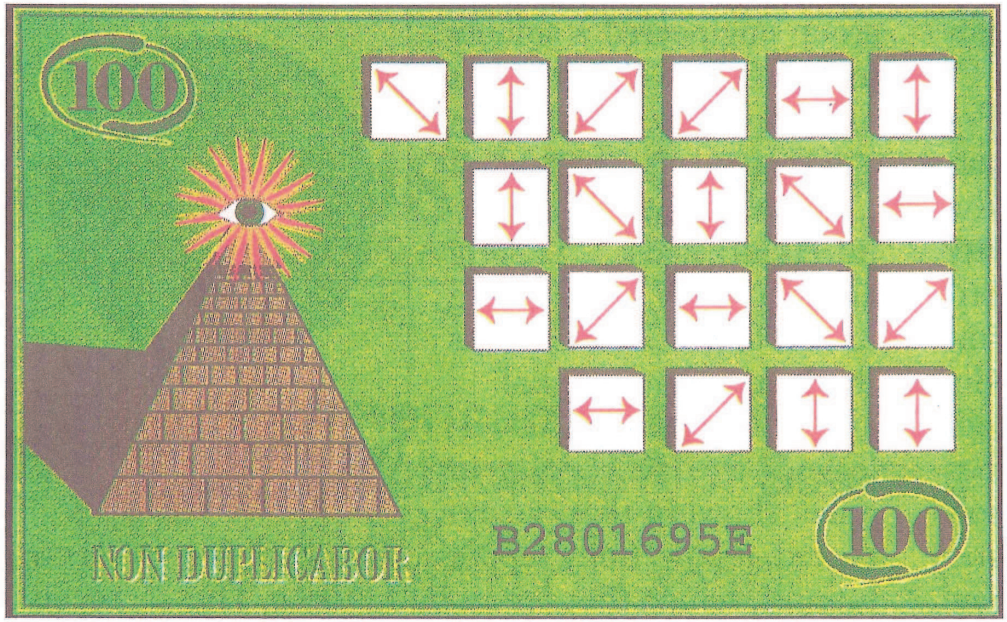
We give a new protocol for creating unforgeable quantum states.

Proposals for Quantum Money

Farhi, Gosset, Hassidim, Lutomirski, Shor (based on knot invariants, 2009)
Aaronson and Christiano (based on subspaces, 2012) broken
Mark Zhandry (complexity-theoretic, 2017) broken?
Daniel Kane (based on modular forms, 2018)
Shor (based on lattice cryptography, 2020)

History

One of the first proposed quantum computing ideas was quantum money (Stephen Wiesner, 1970, 1983).



In each bill, there is a sequence of quantum states in one of two complementary bases (so one of $|\updownarrow\rangle, |\leftrightarrow\rangle$ $|\nearrow\rangle, |\nwarrow\rangle$). By the quantum no-cloning theorem, anyone who does not know the polarizations of these states cannot copy them.
(Wiesner proposed this before the no-cloning theorem had been formally proven, although it's clear that he knew it intuitively.)

Problems with Wiesner's Money

How to check the money? The mint knows the polarizations, and so can easily check it.

Problems with Wiesner's Money

How to check the money? The mint knows the polarizations, and so can easily check it.

We want the merchant to be able to verify that the bill is legit without sending it back to the mint.
If the merchant knows the quantization axis and eigenvalue of each qubit, then the merchant can verify the money.

Problems with Wiesner's Money

How to check the money? The mint knows the polarizations, and so can easily check it.

We want the merchant to be able to verify that the bill is legit without sending it back to the mint.
If the merchant knows the quantization axis and eigenvalue of each qubit, then the merchant can verify the money.

However, he could also make new bills exactly like the one he got.

Problems with Wiesner's Money

How to check the money? The mint knows the polarizations, and so can easily check it.

We want the merchant to be able to verify that the bill is legit without sending it back to the mint.
If the merchant knows the quantization axis and eigenvalue of each qubit, then the merchant can verify the money.

However, he could also make new bills exactly like the one he got.

We would like a verification procedure that does not allow the merchant to make fresh bills.

Cryptography Background and Motivation

For many years, cryptography was done with *ad hoc* cryptosystems, many of which were eventually broken.
Over the last few decades, cryptography has become much more mathematical, and theoretical computer scientists try to prove security of cryptosystems.
There are two kinds of proofs of security in cryptography (both classical and quantum): security through information and security through complexity.

Definitions

Informationally Secure

Computationally Secure

Definitions

Informationally Secure

No matter how powerful a computer an adversary has, he will not be able to break the cryptosystem, because he doesn't have access to enough information.

Computationally Secure

Definitions

Informationally Secure

No matter how powerful a computer an adversary has, he will not be able to break the cryptosystem, because he doesn't have access to enough information.

Computationally Secure

The security of the cryptosystem relies on the difficulty of solving some computationally hard problem

Quantum cryptography

The BB84 protocol for quantum key distribution can be proved informationally secure, assuming the laws of quantum mechanics.

This solves a task which is impossible to perform with an informationally secure protocol and classical computing.

Quantum cryptography

The BB84 protocol for quantum key distribution can be proved informationally secure, assuming the laws of quantum mechanics.

This solves a task which is impossible to perform with an informationally secure protocol and classical computing.

This quantum money research started with us considering the question of whether there were any cryptographic tasks that a quantum computer might perform with computational security, but which were impossible for a digital computer to perform.

Task: Quantum Money

We would like one of the players in the protocol (we will call her the mint) to be able to make a state $|\$_i\rangle$, and a verification protocol P_i, so that

Task: Quantum Money

We would like one of the players in the protocol (we will call her the mint) to be able to make a state $|\$_i\rangle$, and a verification protocol P_i, so that

a) $|\$_i\rangle$ passes the test P_i,

Task: Quantum Money

We would like one of the players in the protocol (we will call her the mint) to be able to make a state $|\$_i\rangle$, and a verification protocol P_i, so that

a) $|\$_i\rangle$ passes the test P_i,

b) the test P_i does not destroy $|\$_i\rangle$,

Task: Quantum Money

We would like one of the players in the protocol (we will call her the mint) to be able to make a state $|\$_i\rangle$, and a verification protocol P_i, so that

a) $|\$_i\rangle$ passes the test P_i,

b) the test P_i does not destroy $|\$_i\rangle$,

c) a possible counterfeiter holding both the state $|\$_i\rangle$ and knowing the protocol P_i cannot produce a state of two quantum systems (possibly entangled) that both pass the test P_i.

One-of-a-Kind States

In fact, in our knot-invariant money protocol, in Kane's protocol based on modular forms, and in our lattice money protocol, we believe that not even the mint can efficiently make another copy of the state $|\$_i\rangle$ that pases the test P_i.

One-of-a-Kind States

In fact, in our knot-invariant money protocol, in Kane's protocol based on modular forms, and in our lattice money protocol, we believe that not even the mint can efficiently make another copy of the state $|\$_i\rangle$ that pases the test P_i.

Called *public key quantum money* by Aaronson.

Related to *quantum lightning*, defined by Mark Zhandry (lightning never strikes twice in the same place.)

How to Use Unforgeable States as Money

The mint makes quantum states, and gets pairs $|\$_i\rangle$, P_i.

The mint publishes a list of valid pairs i, P_i somewhere secure (so nobody can add an extra pair to the list).

It then hands out some $|\$_i\rangle$, together with i, to a customer who wants quantum money.

How to Use Unforgeable States as Money

The mint makes quantum states, and gets pairs $|\$_i\rangle$, P_i.

The mint publishes a list of valid pairs i, P_i somewhere secure (so nobody can add an extra pair to the list).

It then hands out some $|\$_i\rangle$, together with i, to a customer who wants quantum money.

Then anybody with $|\$_i\rangle$ who knows i (and has a quantum computer) can check that it is a valid quantum money state; i.e., that i is on the list, and $|\$_i\rangle$ passes the test P_i.

Uses for Unforgeable States: Quantum ID Cards

You could put a unforgeable quantum state into an ID card. These ID cards could be stolen, but they could not be forged. Of course, for both money and quantum ID cards, you need to have long-lived quantum states.

Uses for Unforgeable States: Quantum ID Cards

You could put a unforgeable quantum state into an ID card.

These ID cards could be stolen, but they could not be forged.

Of course, for both money and quantum ID cards, you need to have long-lived quantum states.

Uses for Unforgeable States: Quantum ID Cards

You could put a unforgeable quantum state into an ID card.

These ID cards could be stolen, but they could not be forged.

Of course, for both money and quantum ID cards, you need to have long-lived quantum states.

Question: Could this property be of some use as a subroutine for some other quantum cryptographic protocols? This use might not require such long-lived quantum states.

How does our quantum money protocol work?

Outline

We will

1. Sketch our first candidate for quantum lattice money,
2. Explain why it doesn't work.

How does our quantum money protocol work?

Outline

We will

1. Sketch our first candidate for quantum lattice money,
2. Explain why it doesn't work.
3. Sketch our next candidate for quantum lattice money.
4. Explain why it still doesn't work.

How does our quantum money protocol work?

Outline

We will

1. Sketch our first candidate for quantum lattice money,
2. Explain why it doesn't work.
3. Sketch our next candidate for quantum lattice money.
4. Explain why it still doesn't work.
5. Sketch our current candidate for quantum lattice money.
6. Sketch the proof that it works.

Short vectors in a lattice

A lattice is the set of all integer combination of n vectors $\{v_i\}$ in n dimensions.

$$L = \left\{ \sum_{k=1}^{n} i_k v_k | i_1, i_2 \ldots, i_n \in \mathbb{Z} \right\}.$$

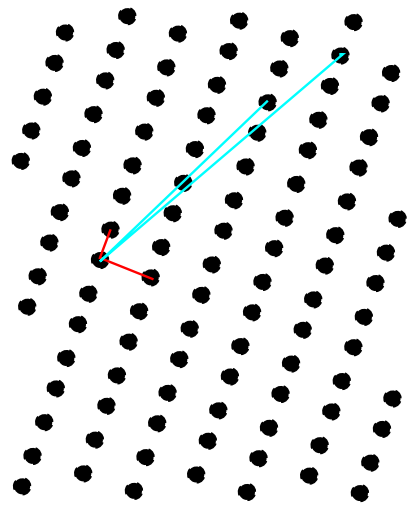
Short vectors in a lattice

A lattice is the set of all integer combination of n vectors $\{v_i\}$ in n dimensions.

$$L = \left\{ \sum_{k=1}^{n} i_k v_k | i_1, i_2 \ldots, i_n \in \mathbb{Z} \right\}.$$

Problem: Given a basis of long vectors for L, find a basis of short vectors.

L^3 algorithm: finds a basis exponentially longer (exponential in n) than the shortest possible basis.



Bounded Distance Decoding

Suppose we have a vector x that is very close to a lattice point v. Then we can find that lattice point in polynomial time.

Bounded Distance Decoding

Suppose we have a vector x that is very close to a lattice point v. Then we can find that lattice point in polynomial time.

What does very close mean?

It means exponentially closer than the shortest vector in the lattice.

Gaussian Sampling

If we have a big enough ball around some point x, we can sample lattice points v with probability proportional to

$$\exp\left(\frac{-(v-x)^2}{2\sigma^2}\right)$$

Gaussian Sampling

If we have a big enough ball around some point x, we can sample lattice points v with probability proportional to

$$\exp(\frac{-(v-x)^2}{2\sigma^2})$$

What does "big enough" mean?
It means σ should be exponentially larger than the shortest basis of the lattice.

Gaussian superposition

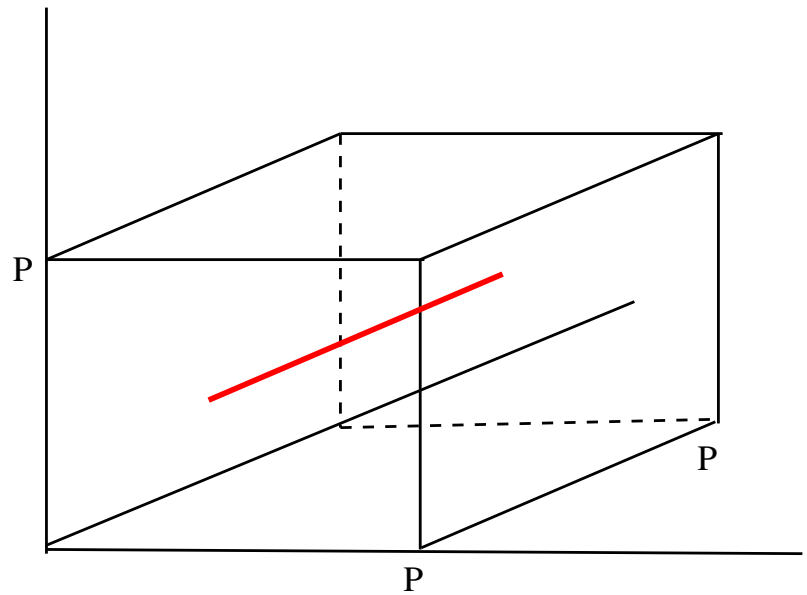
If σ is exponentially larger than the shortest basis, we can create the superposition of lattice points in a Gaussian ball around x in quantum polynomial time:

$$\frac{1}{Q}\sum_{v\in\mathcal{L}}\exp\left(\frac{-(v-x)^2}{4\sigma^2}\right)|\,v\rangle$$

This is done with the same technique as Gaussian sampling, but adapted to quantum algorithms.

Subclass of lattices

Consider lattices in n dimensions, on a P^n cube of integers (so everything is done mod P), where there is one lattice point in every hyper-column.

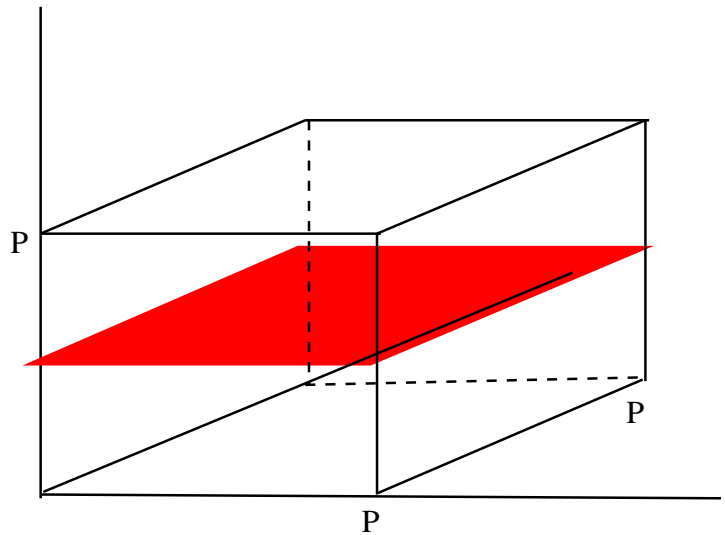


There are P^{n-1} lattice vectors in the cube. Here, P isn't necessarily a prime (although you can assume it is for intuition).

Dual of these lattices

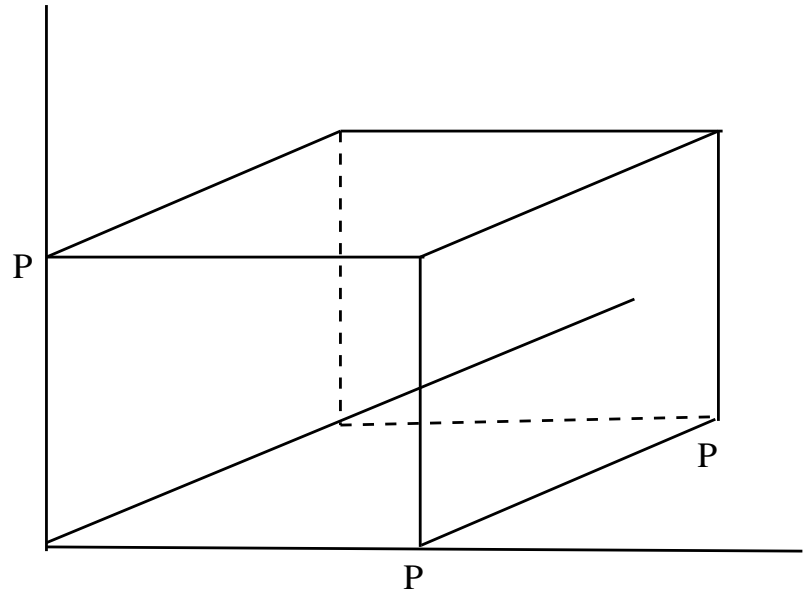
The dual lattice is the set of all vectors which are perpendicular to all vectors in a lattice:

$$\mathcal{L}^{\perp} = \{x | x \cdot v \in P\mathbb{Z} \ \forall v \in \mathcal{L}\}.$$



The dual lattice that has one vector in each hyperplane. (So P lattice vectors total.)
In these lattices, each lattice vector is a multiple of a generating vector.

Hardness



If the short vector problem is hard in arbitrary lattices, it is still hard in these lattices, even if $P \approx \exp(\mathrm{poly}(n))$ (Eldar and Shor)

Quantum Fourier transform on $\mathbb{Z}_P^n$ (Eldar and Shor)

We can define a quantum Fourier transform that takes vectors in the lattice $\mathcal{L}$ to a superposition of vectors in $\mathcal{L}$.
The equation for this transform is

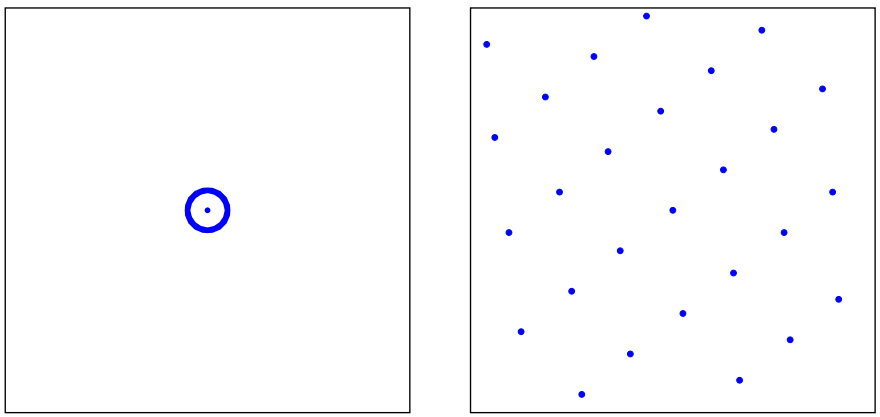
$$|x\rangle \rightarrow \frac{1}{P^{(n-1)/2}} \sum_{y \in \mathcal{L}} \exp\left(-2\pi i \frac{x \cdot y}{P}\right) |y\rangle$$

Properties of Quantum Fourier transform on $\mathbb{Z}_P^n$

The Quantum Fourier transform takes a Gaussian superposition of lattice points of L around 0 to a Gaussian superposition of lattice points of L around each of the points of the dual lattice $L^{\perp}$.

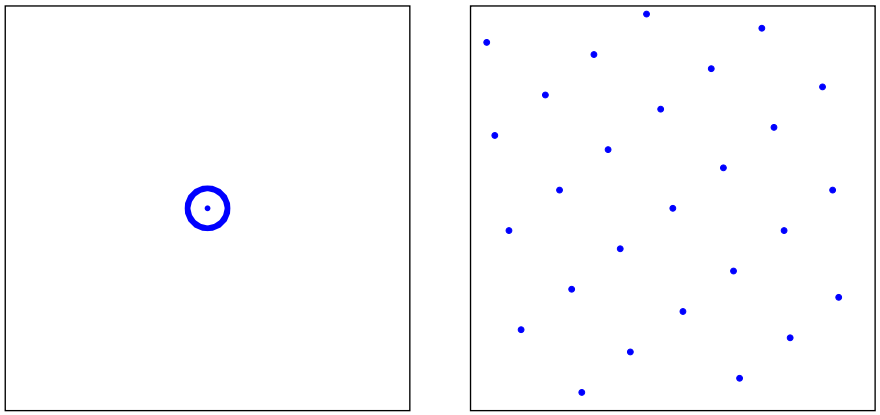
Properties of Quantum Fourier transform on $\mathbb{Z}_P^n$

The Quantum Fourier transform takes a Gaussian superposition of lattice points of L around 0 to a Gaussian superposition of lattice points of L around each of the points of the dual lattice $L^{\perp}$.



Properties of Quantum Fourier transform on $\mathbb{Z}_P^n$

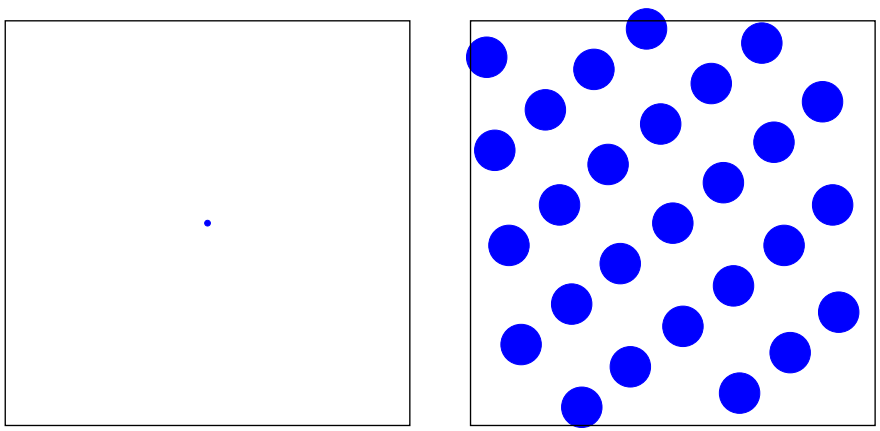
The Quantum Fourier transform takes a Gaussian superposition of lattice points of L around 0 to a Gaussian superposition of lattice points of L around each of the points of the dual lattice $L^{\perp}$.



If the original Gaussian superposition is large, the Gaussian superpositions around each point of the dual lattice are small.

Properties of Quantum Fourier transform on $\mathbb{Z}_P^n$

The Quantum Fourier transform takes a Gaussian superposition of lattice points of L around 0 to a Gaussian superposition of lattice points of L around each of the points of the dual lattice $L^{\perp}$.



If the original Gaussian superposition is small, the Gaussian superpositions around each point of the dual lattice are large.

Properties of Quantum Fourier transform on $\mathbb{Z}_P^n$

If you start with a Gaussian ball centered at a dual lattice vector $v \neq 0$, you still get Gaussian balls around each dual lattice vector.

But now, the Gaussian ball around dual lattice vector w has phase

$$\exp\left(-2\pi i \frac{v \cdot w}{P}\right).$$

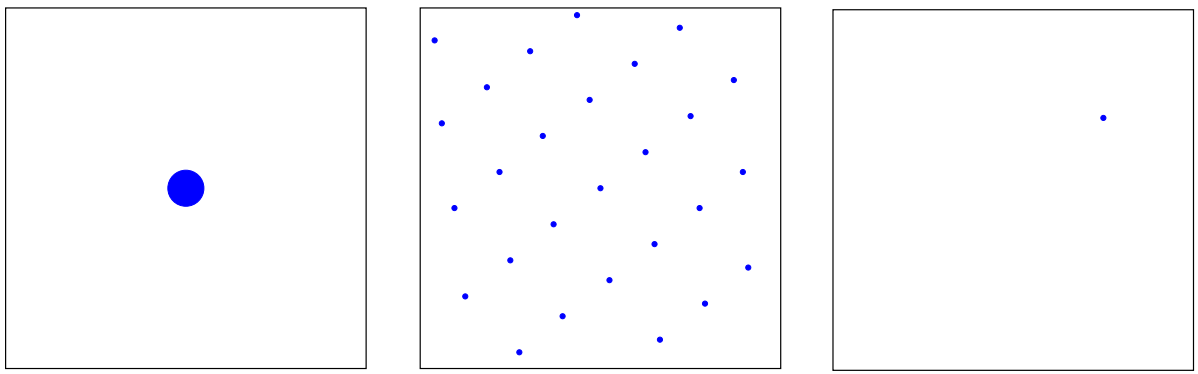
Simple Algorithm (does not work)

The quantum money state is a Gaussian superposition of lattice points in a small ball around a dual latitce vector w.

Simple Algorithm (does not work)

The quantum money state is a Gaussian superposition of lattice points in a small ball around a dual latitce vector w.

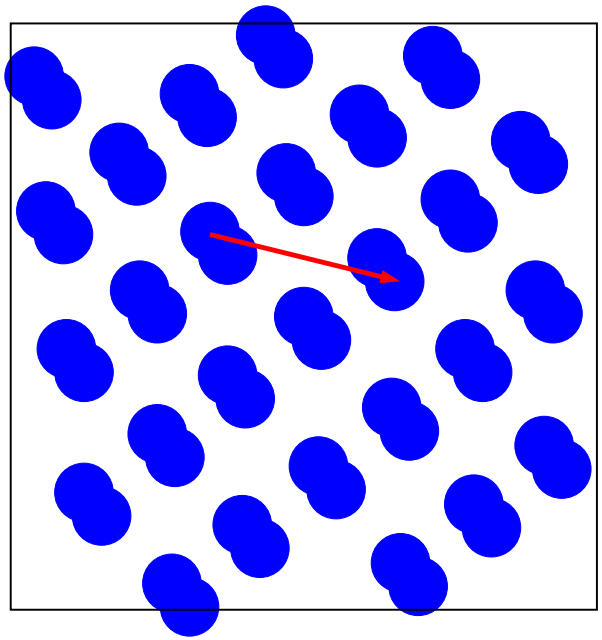
To create the quantum money state:



Create large Gaussian ball at 0. Take the Fourier transform. Measure nearest dual lattice vector to get small Gaussian ball around one dual vector w. This is your quantum state.

Simple Algorithm (does not work), continued

To verify the quantum money state:



Check that it is a superposition of all points near dual lattice vector w.

Take Fourier transform to get big Gaussian balls around all dual lattice points.

Shift lattice balls and measure overlap using SWAP test. (You can predict the exact overlap.)

Possible Objection

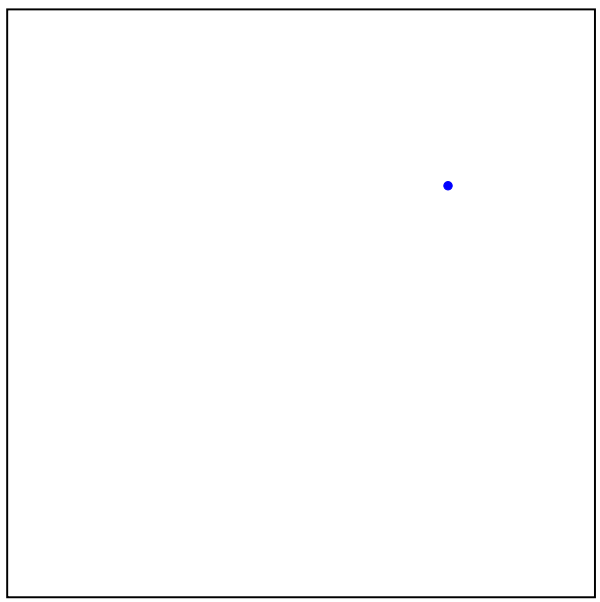
Because we don't pass the verification test with probability $1 - \epsilon$, verification destroys the quantum money state.

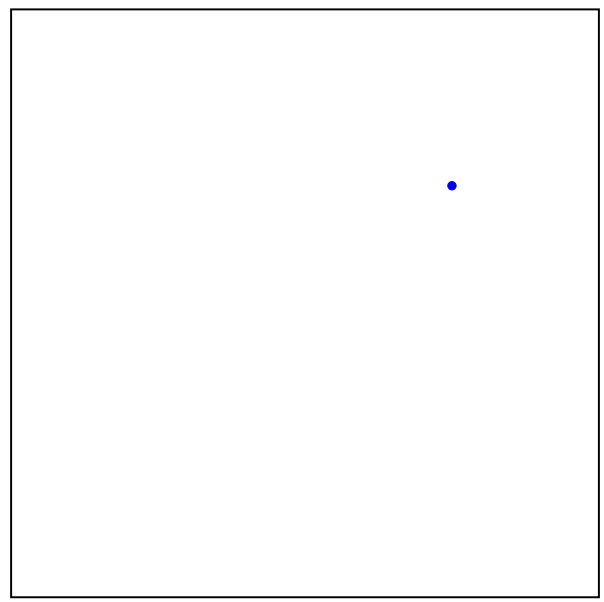
Solution: Use statistics to enhance the probability of the verification test so it is close to 1.

This means that the money will be many copies of small Gaussian balls (each centered at a different lattice point).

Why shouldn't you be able to copy?

Suppose you could copy. You could sample from each of the original and the copy, and get two lattice vectors which are both close to w. Their difference is close to 0, thus a short vector.





Why doesn't this protocol work?

We don't know how to distinguish between having one lattice vector near w and having a Gaussian superposition of lattice vectors near w.

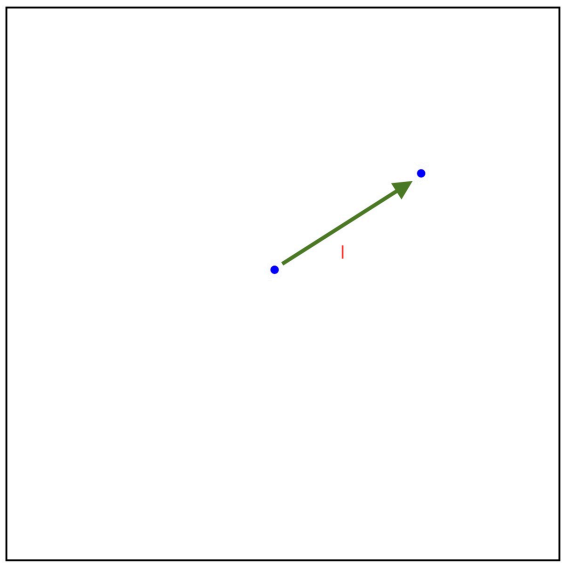
Why doesn't this protocol work?

We don't know how to distinguish between having one lattice vector near w and having a Gaussian superposition of lattice vectors near w.

So somebody who wanted to counterfeit this money could simply measure one lattice vector from the Gaussian ball and make arbitrarily many copies of that.

How to fix it?

Use a superposition of two copies of the Gaussian superposition around dual lattice vectors w_1 and w_2. Let ℓ be the vector between them: $\ell = w_1 - w_2$.

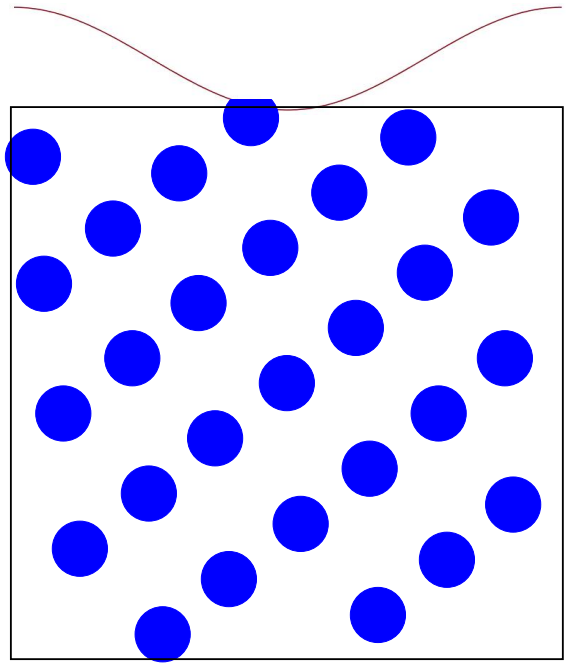


When we take the Fourier transform, we get interference; the large Gaussian balls coming from the small Gaussian ball around w_1 interfere with those coming from those around w_2, because they have different phases.

What happens after the Fourier transform?

Why does this work?

When we take the Fourier transform, the big balls coming from each of the small balls have different phases on them. So some have larger amplitudes and some have smaller amplitudes. This makes the probability that we see a point with x in the ith coordinate proportional to $\cos^2(2\pi x/P)$. We can observe this interference because it's not washed out by the width of the Gaussian balls.



How do we choose ℓ?

We want a vector ℓ in the dual lattice such that

$$\ell \cdot v = v(1) \ \forall v \in \mathcal{L}^{\perp}$$

where $v(1)$ is the first coordinate of v.

Easy to find: choose any vector $w \in \mathcal{L}^{\perp}$ and let

$$\ell = w(1)(w \cdot w)^{-1} w.$$

How do we choose ℓ?

We want a vector ℓ in the dual lattice such that

$$\ell \cdot v = v(1) \ \forall v \in \mathcal{L}^{\perp}$$

where $v(1)$ is the first coordinate of v.

Easy to find: choose any vector $w \in \mathcal{L}^{\perp}$ and let

$$\ell = w(1)(w \cdot w)^{-1} w.$$

Then

$$\ell \cdot w = w(1)(w \cdot w)^{-1}(w \cdot w) = w(1).$$

Why doesn't this algorithm work?

It doesn't work for the same reason that the first algorithm doesn't work. We can find a vector ℓ' in $\mathcal{L}$ very close to ℓ.

Easy to check that $\ell' = \ell - e_1$ is orthogonal to all vectors in $\mathcal{L}^{\perp}$.

Why doesn't this algorithm work?

It doesn't work for the same reason that the first algorithm doesn't work. We can find a vector ℓ' in $\mathcal{L}$ very close to ℓ.

Easy to check that $\ell' = \ell - e_1$ is orthogonal to all vectors in $\mathcal{L}^\perp$.

Now, a counterfeiter can just choose two vectors u_1 and u_2 in $\mathcal{L}$ which are close to w_1 and w_2, and use

$$\frac{1}{\sqrt{2}}\left(|\, u_1\rangle + |\, u_2\rangle \right)$$

as our counterfeit money.

Why doesn't this algorithm work?

It doesn't work for the same reason that the first algorithm doesn't work. We can find a vector ℓ' in $\mathcal{L}$ very close to ℓ.

Easy to check that $\ell' = \ell - e_1$ is orthogonal to all vectors in $\mathcal{L}^\perp$.

Now, a counterfeiter can just choose two vectors u_1 and u_2 in $\mathcal{L}$ which are close to w_1 and w_2, and use

$$\frac{1}{\sqrt{2}}\big(\,|\,u_1\rangle + |\,u_2\rangle\,\big)$$

as our counterfeit money.

Since we know u_1 and u_2, we can make as many copies as we like.

How do we fix this?

We change ℓ.

Let us choose ℓ so that

$$\ell \cdot v = \alpha v(1) \ \forall v \in \mathcal{L}^{\perp}.$$

When we find ℓ', we have that the vectors ℓ and ℓ' differ by an amount comparable to the radius of a small Gaussian ball.

How do we fix this?

We change ℓ.

Let us choose ℓ so that

$$\ell \cdot v = \alpha v(1) \; \forall v \in \mathcal{L}^{\perp}.$$

When we find ℓ', we have that the vectors ℓ and ℓ' differ by an amount comparable to the radius of a small Gaussian ball.

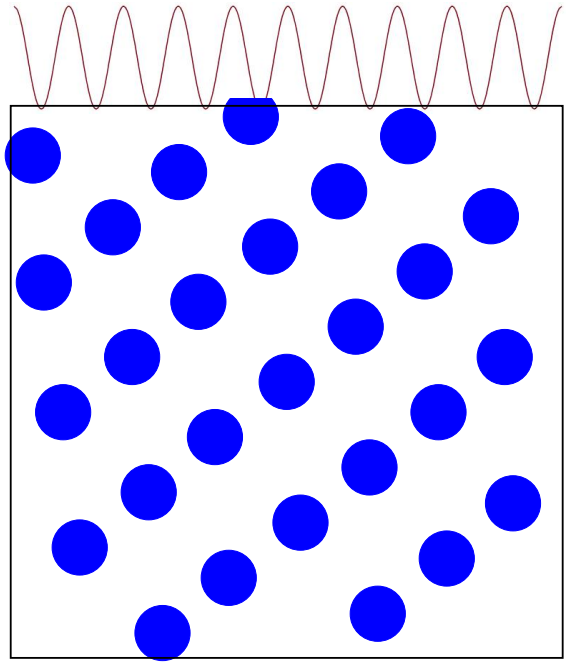
So now, the average of a sample of points cannot be at the center of the Gaussian balls, w_1 or w_2, but will be significantly off-center.

We can test for this.

What happens after the Fourier transform?

Now, there are α periods of the cosine function in the interference pattern.

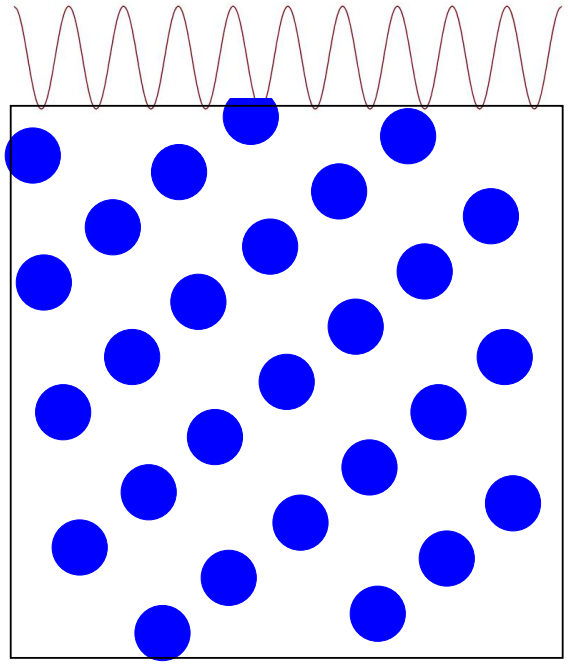
We choose ℓ so that the period of the cosine function is comparable to the width of the Gaussian balls. Now, the interference isn't washed out completely (it would be if the period was much small than the width of the Gaussian balls), so the verification still works.



What happens after the Fourier transform?

Now, there are α periods of the cosine function in the interference pattern.

We choose ℓ so that the period of the cosine function is comparable to the width of the Gaussian balls. Now, the interference isn't washed out completely (it would be if the period was much small than the width of the Gaussian balls), so the verification still works.



There is a balance between the period of the cosine function and how close ℓ' is to a dual lattice vector, but we can choose α so that our quantum money protocol works.

Challenges

- Improve this quantum money scheme (we haven't worried about the time and space needed, beyond making sure they are polynomial).

Challenges

- Improve this quantum money scheme (we haven't worried about the time and space needed, beyond making sure they are polynomial).
- Come up with other quantum money schemes.

Challenges

- Improve this quantum money scheme (we haven't worried about the time and space needed, beyond making sure they are polynomial).
- Come up with other quantum money schemes.
- Are there any other cryptographic protocols which are impossible classically, but which can be done on a quantum computer?

Thank You