Quantum money from knots

Quantum money from knots

Edward Farhi, David Gosset, Avinatan Hassidim,
Andrew Lutomirski, and Peter Shor

April 28, 2010

Abstract

Quantum money is a cryptographic protocol in which a mint can produce a quantum state, no one else can copy the state, and anyone (with a quantum computer) can verify that the state came from the mint. We present a concrete quantum money scheme based on superpositions of diagrams that encode oriented links with the same Alexander polynomial. We expect our scheme to be secure against computationally bounded adversaries.

1 Introduction

In this paper, we present quantum money based on knots (see Figure 1). The purported security of our quantum money scheme is based on the assumption that given two different looking but equivalent knots, it is difficult to explicitly find a transformation that takes one to the other.

Figure 1: Quantum money from knots

One of the problems in classical security is that information can be copied: passwords can be stolen, songs can be pirated, and when you email an attachment, you still have the original. One implication is that E-commerce requires communicating with a server (e.g. the credit card company or PayPal) whenever one makes a transaction. One could hope that the no-cloning theorem would help circumvent this and enable a physical quantum state to function like money. Such money could be used in transactions both in person and on a future "Quantum Internet", not requiring contact with a central authority.

As we will see, quantum money is harder to forge than the paper bills in our wallets. In order for a quantum state to function as money, we require that:

1. The mint can produce it.

2. Anyone can verify it. That is, there is an efficient measurement that anyone (with a quantum device) can perform on a quantum money state that accepts genuine money with high probability and without significantly damaging the money.

3. No one can forge it. That is, no one but the mint can efficiently produce states that are accepted by the verifier with better than exponentially small probability. In particular, it should not be possible to copy a bill.

A quantum money scheme has two components: pieces of quantum money and an algorithm M that verifies quantum money. A piece of quantum money consists of a classical serial number p (authenticated as coming from the mint) along with an associated quantum state $|\$_p\rangle$ on n qubits. The verification algorithm M takes as input a quantum state $|\phi\rangle$ and a serial number q and then decides whether or not the pair $(q, |\phi\rangle)$ is a piece of quantum money. If the outcome is "good money" then the verifier also returns the state $|\phi\rangle$ undamaged so it can be used again. We now formalize each of the above requirements:

1. There is a polynomial-time algorithm that produces both a quantum money state $|\$_p\rangle$ and an associated serial number p.

2. Running the verification algorithm M with inputs p and $|\$_p\rangle$ returns "good money" and does not damage $|\$_p\rangle$. Furthermore, anyone with access to a quantum computer (for example a merchant) can run the verification algorithm M.

3. Given one piece of quantum money $(p, |\$_p\rangle)$, it is hard to generate a quantum state $|\psi\rangle$ on $2n$ qubits such that each part of $|\psi\rangle$ (along with the original serial number p) passes the verification algorithm.

What stops anyone other than the mint, using the same algorithm as the mint, from producing counterfeit money states? This is why the serial number needs to be authenticated. When the mint does a production run it produces a set of pairs $(p, |\$_p\rangle)$. In our quantum money scheme, the mint does not in advance choose the serial numbers, but rather they are produced by a random process. A rogue mint running the same algorithm as the mint can produce a new set of money pairs, but (with high probability) none of the serial numbers will match those that the mint originally produced. A simple way to ensure security is for the mint to publish the list of valid serial numbers and for the merchant to check against this list to see if the serial number from the tendered money is authentic.

An alternative to publishing a list of serial numbers is for the money state to come with a digital signature of the serial number. There are known classical

digital signature protocols that allow anyone who knows the mint's "public key" to verify that a serial number is one that the mint did indeed produce. Still, the authenticated description of the serial number can easily be copied and this is where the quantum security comes into play: knowledge of the serial number p does not allow one to copy the associated quantum state $|\$_p\rangle$. In the remainder of this paper, we assume that anyone verifying quantum money will also check the authentication of the serial number.

We can imagine two different ways to use quantum money for commerce. If we had the technology to print a quantum state onto a piece of paper, then we could use quantum money protocols to enhance the security of paper money against forgery. Alternatively, people could use quantum money states on their personal quantum computers to conduct business either in person or over a quantum Internet. If small portable quantum computers were available (imagine quantum smart phones or quantum debit cards), then it would be easy to buy things with quantum money instead of paper money.

For these uses, the "quantum money" seen by an end-user would either be a file on a quantum computer or a physical piece of money with a quantum state somehow attached.

1.1 Prior work

The idea of quantum money was introduced by Wiesner in 1969, where he proposed using the no cloning theorem to generate bills which can not be copied [4, 14]. However, in Wiesner's original definition the only entity that could verify the quantum money state was the mint that produced it.

There is recent interest in designing quantum money which can be verified by anyone who has a quantum computer (also called "public-key quantum money"). Such money cannot be information-theoretically secure; instead, it must rely on computational assumptions. Aaronson showed that public-key quantum money exists relative to a quantum oracle and proposed a concrete scheme without an oracle [1]. Aaronson's concrete scheme was broken in [8].

A different approach to quantum money was taken by Mosca and Stebila [11, 9, 10, 12], which is somewhat based on ideas of Tokunaga, Okamoto, and Imoto [13]. This line of research adds the additional requirement that all quantum money states be identical and all share the same verification procedure. This additional requirement is an abstraction of coins (which are identical) as opposed to bills (which can have different serial numbers). In their works, Mosca and Stebila give a way to generate quantum coins using a specialized random quantum oracle. They also show how to generate quantum coins using blind quantum computation. This requires the mint to be involved in every transaction.

The main open question is how to design any secure quantum money (whether of the quantum coin variety or otherwise) that does not require an oracle and in which the mint's participation is not needed to use the money.

A natural approach would be for the mint to choose some classical secret and, using the secret, generate the money state and the serial number. It should

be hard to copy the quantum money state without knowing the secret. One such proposal for a quantum money scheme is based on product states. The mint first chooses a random product state $|\$\rangle$ on n qubits and then constructs a set of local projectors $|\psi_i\rangle\langle\psi_i|$ for $i \in \{1, ...m\}$ such that $|\$\rangle$ is a zero eigenvector of each projector (and m is large enough that $|\$\rangle$ is the only state which is in the common zero eigenspace). The serial number associated with the state $|\$\rangle$ is a bit string which describes each of the projectors. To verify a state $|\phi\rangle$, one measures each projector $|\psi_i\rangle\langle\psi_i|$ and accepts the state if and only if the outcomes are all 0. Unfortunately this money scheme is insecure. It was shown in [5] that given a piece of product state quantum money $|\$\rangle$ and the associated serial number it is possible to learn the classical secret (the description of the product state). A generalized version of this attack was also given in [5].

Due to the possibility of this type of attack, it was proposed in [8] to use quantum money which is not based on a classical secret, but rather on the hardness of generating a known superposition. However, [8] did not present a proposal for a quantum money scheme, only a blueprint.

1.2 A blueprint for quantum money

We present a scheme for quantum money which is not based on quantum oracles and does not require communicating with the mint for verification. The overall structure of the scheme is based on ideas from [8].

The mint begins by generating a uniform superposition

$$|\text{initial}\rangle = \frac{1}{\sqrt{|\mathcal{B}|}} \sum_{e \in \mathcal{B}} |e\rangle|0\rangle$$

where $\mathcal{B}$ is a big set. It then applies some function f to e, obtaining the state

$$\frac{1}{\sqrt{|\mathcal{B}|}} \sum_{e \in \mathcal{B}} |e\rangle|f(e)\rangle.$$

Finally, the mint measures the value of f. If the result of the measurement is v, then the residual state is

$$|\$_v\rangle|v\rangle = \frac{1}{\sqrt{N_v}} \sum_{\substack{e \in \mathcal{B} \\ f(e)=v}} |e\rangle|v\rangle.$$

The quantum money is the classical serial number v and the quantum state $|\$_v\rangle$.

To verify the money state, the verifier first measures the value of f on the state she is given to confirm that it has the value v. Then she verifies that the state is in a uniform superposition of states with the same value v for f. It is not obvious in general how to verify this uniform superposition and in our scheme we will see that this is one of the main challenges.

For such a scheme to work, the following properties are required:

- Measuring f twice on two unentangled copies of the initial state should give different values with probability exponentially close to 1. Otherwise, the forger can just repeat the procedure performed by the mint, and obtain a money state with non negligible probability.
- For "most" values v, there should be exponentially many initial states e which have $f(e) = v$. Moreover, given one state $|e\rangle$ with $f(e) = v$ it should be hard to generate the uniform superposition over all such states.
- It should be possible for the verifier to verify that he has a uniform superposition over states with the same value of f.

1.3 Quantum money from knots

To implement this kind of quantum money, we use ideas from knot theory. The initial state generated by the mint is a uniform superposition over planar grid diagrams, which are compact representations of oriented links (an oriented knot is a knot with a preferred direction around it, and an oriented link is a collection of possibly intertwined oriented knots). The function f which the mint measures is the Alexander polynomial of the oriented link represented by a given planar grid diagram. Finally, the verification procedure is based on Reidemeister moves, which do not change the value of the polynomial. (Reidemeister moves turn a knot into another equivalent knot, and the Alexander polynomials for equivalent knots are the same.)

2 Knots, links, and grid diagrams

In this section we briefly review the standard concepts of knots and links and how they are represented using diagrams. The same knot (or link) can be represented as a diagram in many different ways; the Reidemeister moves are a set of local changes to a diagram that do not change the underlying knot or link. We will review how to compute the Alexander polynomial for a given oriented link diagram in polynomial time. The Alexander polynomial is a link invariant in the sense that if we compute its value on diagrams which represent the same oriented link we get the same polynomial. Finally, we review planar grid diagrams and grid moves, which implement Reidemeister moves on grid diagrams.

2.1 Knots and links

We can think of a knot as a loop of string in 3 dimensions, that is, a map from S^1 into $\mathbb{R}^3$. Since it is hard to draw knots in 3 dimensions, usually a knot is represented by a projection of the 3 dimensional object into 2 dimensions where, at each crossing, it is indicated which strand passes above and which below. This is called a knot diagram. In this paper we will be interested in links, which correspond to one or more loops of string (called the components

of the link). An oriented link is a link which has a direction associated with each component. An example of an oriented link diagram is shown in figure 2.

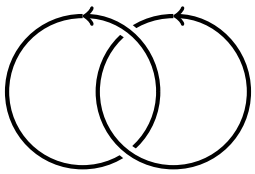

Figure 2: An oriented link diagram.

Two links (or knots) are equivalent if one can be smoothly morphed into the other without cutting the string. If unoriented links K_1 and K_2 are equivalent and they are represented by diagrams D_1 and D_2 respectively, then diagram D_1 can be transformed into D_2 (and vice versa) using the Reidemeister moves pictured in figure 3. (For oriented links the Reidemeister moves can be drawn with the same pictures as in figure 3 but with orientations along the edges which have to be consistent before and after applying the move). Looking at these moves, one sees that if two diagrams can be brought into the same form by using the moves then the diagrams represent equivalent links. The converse of this statement is a theorem.

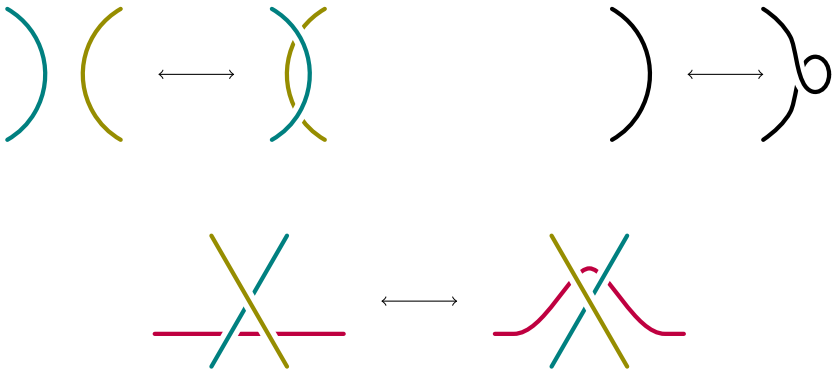

Figure 3: The three Reidemeister moves for unoriented link diagrams.

2.2 The Alexander polynomial of an oriented link

The Alexander polynomial is a polynomial $\Delta(x)$ which can be computed from a given oriented link diagram and which is invariant under the Reidemeister moves. In this section, following Alexander [3], we describe how to compute this polynomial. It will be clear from our discussion that the computation of $\Delta(x)$ can be done in polynomial time in the number of crossings of the diagram.

Suppose we are given a diagram of an oriented link L. If the diagram is disconnected, apply the first Reidemeister move in Figure 3 to connect the diagram. Let us write a for the number of crosses in the diagram. The curve of the diagram then divides the two dimensional plane into $a+2$ regions including one infinite region (this follows from Euler's formula). The following recipe can be used to calculate $\Delta(x)$:

1. For each region $i \in \{1, ..., a+2\}$, associate a variable r_i.

2. For each of the a crossings, write down an equation

$$xr_j - xr_k + r_l - r_m = 0,$$

   where $\{r_j, r_k, r_l, r_m\}$ are the variables associated with the regions adjacent to the crossing, in the order pictured in figure 4.

3. Write the set of equations as a matrix equation

$$\mathcal{M} \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_{m+2} \end{pmatrix} = 0.$$

   This defines the matrix $\mathcal{M}$ which has a rows and $a+2$ columns. The entries of $\mathcal{M}$ are elements of the set $\{\pm 1, \pm x, 1+x, 1-x, -1+x, -1-x\}$.

4. Delete two columns of $\mathcal{M}$ which correspond to adjacent regions in the link diagram. This gives an $a \times a$ matrix $\mathcal{M}_0$.

5. Take the determinant of $\mathcal{M}_0$. This is a polynomial in x. Divide this polynomial by the factor $\pm x^q$ chosen to make the lowest degree term a positive constant. The resulting polynomial is the Alexander polynomial $\Delta(x)$ for the given link.

When we use the Alexander polynomial in this paper, we are referring to the list of coefficients, not the value of the polynomial evaluated at some x.

Figure 4: An equation $xr_j - xr_k + r_l - r_m = 0$ is associated with each crossing, where the adjacent regions $\{r_j, r_k, r_l, r_m\}$ are labeled as shown. Note that the labeling of the adjacent regions depends on the which line crosses on top.

2.3 Grid diagrams

It is convenient to represent knots as *planar grid diagrams*. A planar grid diagram is a $d \times d$ grid on which we draw d X's and d O's. There must be exactly one X and one O in each row and in each column, and there may never be an X and an O in the same cell. We draw a horizontal line in each row between the O and the X in that row and a vertical line in each column between the X and the O in that column. Where horizontal lines and vertical lines intersect, the vertical line always crosses above the horizontal line.

Knots (or links) in a grid diagram carry an implicit orientation: each vertical edge goes from an X to an O, and each horizontal edge goes from an O to an X. Figure 5 shows an example of a $d = 4$ planar grid diagram.

A planar grid diagram G can be specified by two disjoint permutations $\pi_X, \pi_O \in S_d$, in which case the X's have coordinates $\{(i, \pi_X(i))\}$ and the O's have coordinates $\{(i, \pi_O(i))\}$ for $i \in \{1, ..., d\}$. Two permutations are said to be disjoint if for all i, $\pi_X(i) \neq \pi_O(i)$. Any two disjoint permutations $\pi_X, \pi_O \in S_d$ thus define a planar grid diagram $G = (\pi_X, \pi_O)$. Every link can be represented by many different grid diagrams.

We can define grid moves, which are three types of moves (that is, transformations) on planar grid diagrams which, like the Reidemeister moves for link diagrams, are sufficient to generate all planar grid diagrams of the same oriented link.

- The first type of move is a cyclic permutation of either the rows or the columns. Figure 6 shows an example of this type of move on columns. We can think of these moves as grabbing both markers in the rightmost column, pulling them behind of the page, and putting them back down on the left. These moves are always legal. There are equivalent moves on rows.

- The second type of move is transposition of two adjacent rows or columns. This can be done only when no strand would be broken. In Figure 7 we show examples of when this move is allowed. The legality of this move depends only on the position of the markers in the rows or columns being transposed.[1]

- The third type of move adds one row and one column (this is called stabilization) or deletes one row and one column (this is called destabilization), as shown in Figure 8. Destabilization selects three markers forming an "L" shape with sides of length 1 and collapses them into one. That is, it deletes one row and one column such that all three markers are removed. The inverse move selects any marker, adds a row and a column adjacent to that marker, and replaces that marker with three new markers. Any X or O can always be legally stabilized and any three markers forming an "L"

[1] For the case of columns i and $i+1$, the precise condition is that either the two intervals $[\min(x_i, o_i), \max(x_i, o_i)]$ and $[\min(x_{i+1}, o_{i+1}), \max(x_{i+1}, o_{i+1})]$ do not overlap or one of the intervals contains the other.

shape with sides of length 1 can be destabilized unless they form a box (i.e. a 2×2 square with a marker in all four positions).

In the remainder of this paper we will represent links exclusively with planar grid diagrams.

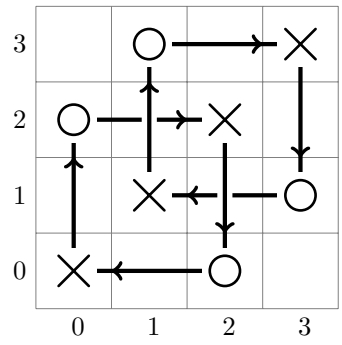

Figure 5: A simple planar grid diagram

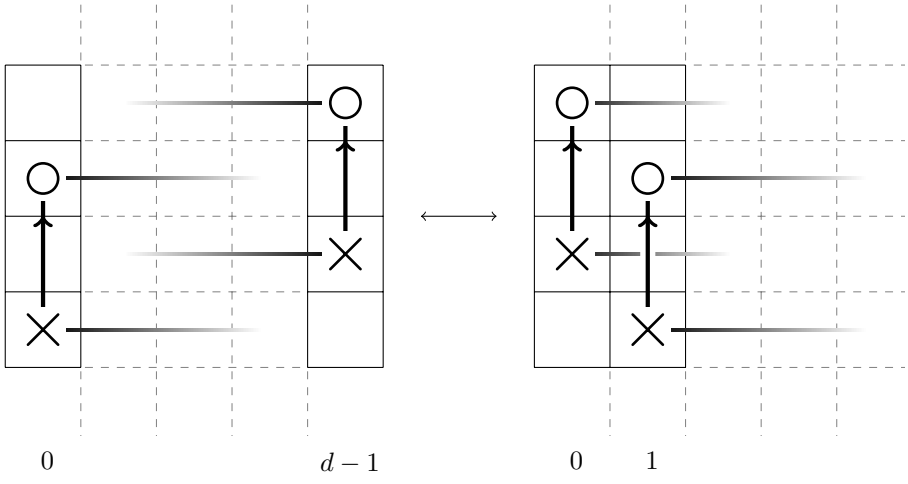

Figure 6: Cyclic permutation of columns. This move is always legal.

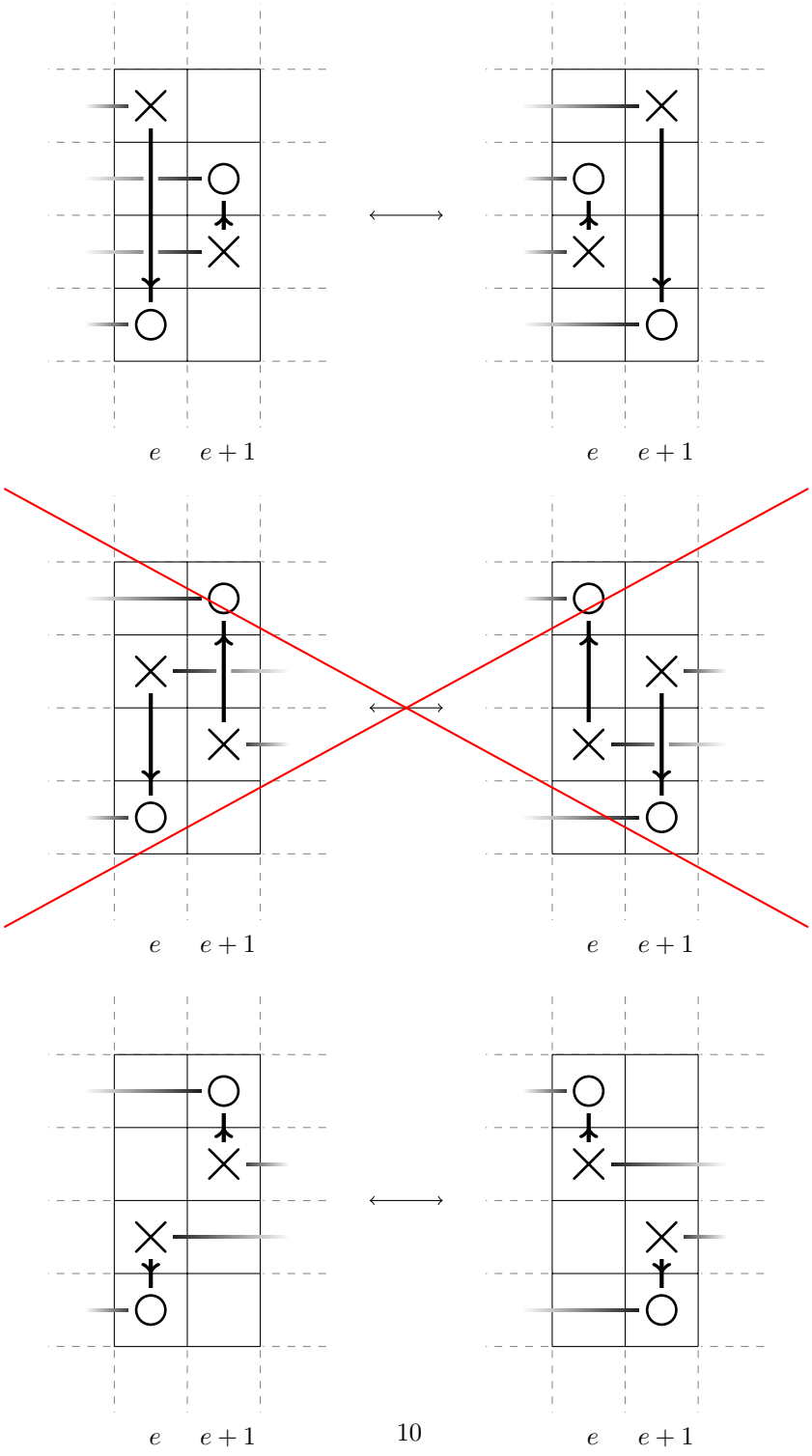

Figure 7: Transposition of adjacent columns e and $e+1$. The legality of these moves depends only on the positions of markers in the columns being transposed. The positions of markers in other columns are irrelevant. The middle move is not allowed.

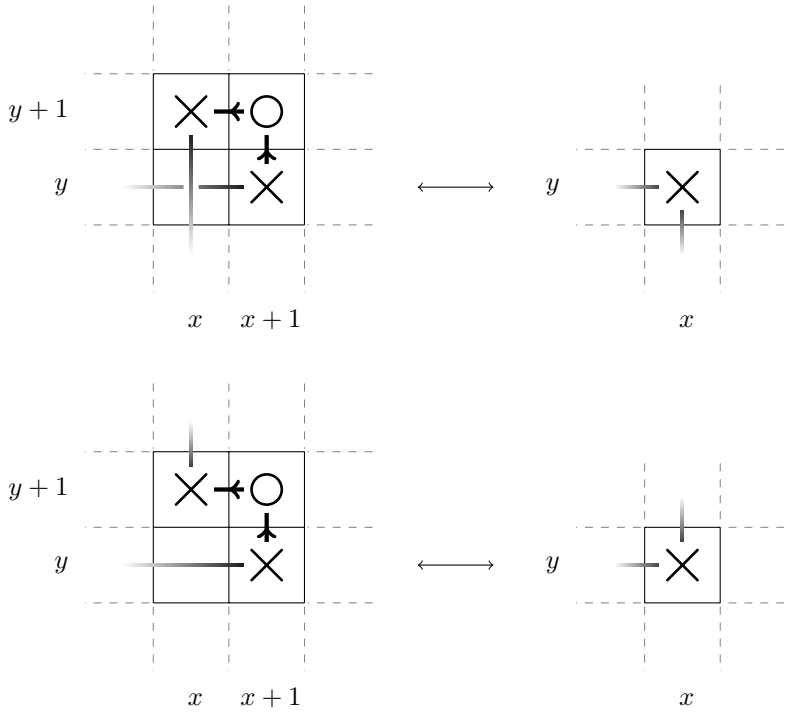

Figure 8: Stabilization (right to left) and destabilization (left to right). This move is legal when rotated as well as when the markers are switched between X and O. The position of other markers does not matter.

3 Quantum money

In this section we describe our quantum money scheme in full detail. In section 3.1 we describe how the mint can make quantum money efficiently. In section 3.2 we describe the verification procedure which can be used by anyone (that is, anyone who owns a quantum computer) to ensure that a given quantum bill is legitimate. In section 3.3 we discuss the security of the verification procedure. In section 3.4 we contrast our quantum money scheme based on knots with a similar but insecure scheme based on graphs.

3.1 Minting quantum money

The basis vectors of our Hilbert space are grid diagrams $|G\rangle = |\pi_X, \pi_O\rangle$, where each grid diagram G is encoded by the disjoint permutations π_X, π_O. The

dimension $d(G)$ is the number of elements in the permutations.[2] The number of disjoint pairs of permutations on d elements is equal to $d!$ times the number of permutations that have no fixed points. The latter quantity is called the number of derangements on d elements, and it is given by $\left[\frac{d!}{e}\right]$ where the brackets indicate the nearest integer function.

To generate quantum money, the mint first chooses a security parameter $\bar{D}$ and defines an unnormalized distribution

$$y(d) = \begin{cases} \frac{1}{d!\left[\frac{d!}{e}\right]} \exp \frac{-\left(d-\bar{D}\right)^2}{2\bar{D}} & \text{if } 2 \leq d \leq 2\bar{D} \\ 0 & \text{otherwise.} \end{cases}$$

Define an integer valued function

$$q(d) = \lceil \frac{y(d)}{y_{min}} \rceil,$$

where $\lceil \cdot \rceil$ means round up and where y_{min} is the minimum value of $y(d)$ for $d \in 2, ..., 2\bar{D}$ (we need an integer valued distribution in what follows). The mint then uses the algorithm in [6] to prepare the state (up to normalization)

$$\sum_{d=2}^{2\bar{D}} d! \sqrt{q(d)} |d\rangle.$$

Using a straightforward unitary transformation acting on this state and an additional register, the mint produces

$$\sum_{d=2}^{2\bar{D}} d! \sqrt{q\left(d\right)} |d\rangle \left(\frac{1}{d!} \sum_{\pi_X, \pi_O \in S_d} |\pi_X, \pi_O\rangle \right),$$

and then measures whether or not the two permutations π_X and π_O are disjoint. (They are disjoint with probability very close to $1/e$.) If the mint obtains the measurement outcome "disjoint", it uncomputes the dimension d in the first register to obtain the state $|\text{initial}\rangle$ on the last two registers, where

$$|\text{initial}\rangle = \frac{1}{\sqrt{N}} \sum_{\text{grid diagrams } G} \sqrt{q\left(d\left(G\right)\right)} |G\rangle \tag{1}$$

and N is a normalizing constant. If the measurement says "not distinct", the mint starts over.

The distribution $q\left(d\right)$ is chosen so that if one were to measure $d(G)$ on $|\text{initial}\rangle$, then the distribution of results would be extremely close to Gaussian with mean $\bar{D}$ restricted to integers in the interval $\left[2, 2\bar{D}\right]$. As $\bar{D}$ becomes large, the missing weight in the tails falls rapidly to zero.

[2] In practice, we will encode d, π_X and π_O as bit strings. Any such encoding will also contain extraneous bit strings that do not describe diagrams; the verification algorithm will ensure that these do not occur.

From the state $|\text{initial}\rangle$, the mint computes the Alexander polynomial $A(G)$ into another register and then measures this register, obtaining the polynomial p. The resulting state is $|\$_p\rangle$, the weighted superposition of all grid diagrams (up to size $2\bar{D}$) with Alexander polynomial p:

$$|\$_p\rangle = \frac{1}{\sqrt{N}} \sum_{G:A(G)=p} \sqrt{q(d(G))}|G\rangle, \tag{2}$$

where N takes care of the normalization. The quantum money consists of the state $|\$_p\rangle$, and the serial number is the polynomial p, represented by its list of coefficients. If the polynomial p is zero, the mint should reject that state and start over. Splittable links have Alexander polynomial equal to zero, and we wish to avoid this case.

3.2 Verifying quantum money

Suppose you are a merchant. Someone hands you a quantum state $|\phi\rangle$ and a serial number which corresponds to a polynomial p and claims that this is good quantum money. To check that indeed this is the case, you would use the following algorithm.

Algorithm 1 Verifying Quantum Money

0. Verify that $|\phi\rangle$ is a superposition of basis vectors that validly encode grid diagrams. If this is the case then move on to step 1, otherwise reject.

1. Measure the Alexander polynomial on the state $|\phi\rangle$. If this is measured to be p then continue on to step 2. Otherwise, reject.

2. Measure the projector onto grid diagrams with dimensions in the range $[\frac{\bar{D}}{2}, \frac{3\bar{D}}{2}]$. If you obtain $+1$ then continue on to step 3. Otherwise, reject. For valid money, you will obtain the outcome $+1$ with high probability and cause little damage to the state. (We discuss the rationale for this step in section 3.3.)

3. Apply the Markov chain verification algorithm described in section 3.2.2. If the state passes this step, accept the state. Otherwise, reject.

If the money passes steps 0 and 1 then $|\phi\rangle$ is a superposition of grid diagrams with the correct Alexander polynomial p. Now passing steps 2 and 3 will (approximately) confirm that $|\phi\rangle$ is the *correct* superposition of grid diagrams. This procedure will accept genuine quantum money states with high probability, but, as we discuss below, there will be other states that also pass verification. We believe that these states are hard to manufacture.

We now discuss the quantum verification scheme used in step 3. We begin by defining a classical Markov chain.

3.2.1 A classical markov chain

The Markov chain is chosen to have uniform limiting distribution over pairs (G, i) where G is equivalent to (and reachable without exceeding grid dimension $2\bar{D}$ from) the starting configuration, and where $i \in \{1, ..., q(d(G))\}$. Therefore, in the limiting distribution (starting from a grid diagram $\tilde{G}$) the probability of finding a grid diagram G (which is equivalent to $\tilde{G}$) is proportional to $q(d(G))$. We use the extra label i in our implementation of the verifier.

There are two types of update rules for the Markov chain. The first type is an update that changes i while leaving G unchanged (the new value of i is chosen uniformly). The second type is an update that changes G to a new diagram G' while leaving i alone (this type of update is only allowed if $i \leq q(d(G'))$). For the moves which leave i unchanged, our Markov chain selects a grid move at random and proposes to update the current grid diagram by applying that move. The move $G \to G'$ is accepted if the value $i \leq q(d(G'))$.

Recall (from section 2.3) that there is a set of grid moves on planar grid diagrams which can be used to transition from one grid diagram to another. These moves only allow transitions between grid diagrams which represent equivalent links, and some of these moves change the dimension of the grid diagram. In general, any two grid diagrams representing equivalent links can be connected by a sequence of these moves (although finding this sequence is not easy). However, this sequence of moves may also pass through grid diagrams which have dimension greater than $2\bar{D}$. In this case, the two equivalent diagrams will not mix, due to the cutoff we defined.

In the appendix we define this Markov chain in detail. This Markov chain does not mix between nonequivalent grid diagrams. As such, it has many stationary distributions. In the next section, we use this Markov chain to produce a quantum verification procedure.

3.2.2 A quantum verification procedure based on a classical Markov chain

Let B denote the Markov matrix. As shown in the appendix, B is a doubly stochastic (the sum of the elements in each row and each column is 1) matrix which can be written as

$$B = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} P_s, \tag{3}$$

where each P_s is a permutation on the configuration space of pairs (G, i) such that $i \leq q(d(G))$, and $\mathcal{S}$ is a set of moves that the Markov chain can make at each iteration. We can view the same matrices as quantum operators acting on states in a Hilbert space where basis vectors are of the form $|G\rangle|i\rangle$. For clarity we will refer to the quantum operators as $\hat{B}$ and $\hat{P}_s$.

Our quantum money states $|\$_p\rangle$ live in a Hilbert space where basis vectors are grid diagrams $|G\rangle$. To enable our verification procedure, we define a related state $|\$'_p\rangle$ on two registers, where the second register holds integers $i \in \{0, ..., q_{max}\}$ (here q_{max} is the maximum that $q(d)$ can reach). $|\$'_p\rangle$ is unitarily related to $|\$_p\rangle$,

so verifying the former is equivalent to verifying the latter. Define a unitary U which acts on basis vectors in the expanded Hilbert space as

$$U\left(|G\rangle|0\rangle\right) = |G\rangle \frac{1}{\sqrt{q(d(G))}} \sum_{i=1}^{q(d(G))} |i\rangle.$$

To obtain $|\$'_p\rangle$, take the quantum money state $|\$_p\rangle$ and adjoin the ancilla register initialized in the state $|0\rangle$. Then apply U to both registers to produce the state

$$\begin{aligned} |\$'_p\rangle &= U\left(|\$_p\rangle|0\rangle\right) \\ &= \frac{1}{\sqrt{N}} \sum_{G:A(G)=p} \sum_{i=1}^{q(d(G))} |G\rangle|i\rangle. \end{aligned}$$

Note that whereas $|\$_p\rangle$ is a weighted superposition in the original Hilbert space (see equation 2), $|\$'_p\rangle$ is a uniform superposition in the expanded Hilbert space. However both $|\$_p\rangle$ and $|\$'_p\rangle$ give rise to the same probability distribution over grid diagrams G.

The subroutine in step 3 of the verification procedure of Section 3.2 starts with the input state $|\phi\rangle$ and applies the following algorithm.[3]

[3] Using a construction from [2], it is possible to associate a Hamiltonian with a Markov chain such as ours. It may also be possible to construct a verifier using phase estimation on this Hamiltonian.

Algorithm 2 Markov chain verification algorithm

1. Take the input state $|\phi\rangle$, append an ancilla register which holds integers $i \in \{0, ..., q_{max}\}$ and prepare the state

$$|\phi'\rangle = U\left(|\phi\rangle|0\rangle\right)$$

using the unitary U defined above.

2. Adjoin an ancilla register which holds integers $s \in \{1, ...|\mathcal{S}|\}$ and initialize the ancilla register to the state $\sum_s \frac{1}{\sqrt{|\mathcal{S}|}}|s\rangle$. This produces the state

$$|\phi'\rangle \sum_s \frac{1}{\sqrt{|\mathcal{S}|}}|s\rangle$$

on three registers (one which holds grid diagrams G, one which holds integers $i \in \{0, ..., q_{max}\}$ and one which holds integers $s \in \{1, ..., |\mathcal{S}|\}$).

3. Repeat $r = \text{poly}(\bar{D})$ times:

   (a) Apply the unitary V, where

$$V = \sum_s \hat{P}_s \otimes |s\rangle\langle s|.$$

   This operator applies the permutation $\hat{P}_s$ to the first two registers, conditioned on the value s in the third register.

   (b) Measure the projector

$$Q = I \otimes I \otimes \left(\sum_{s,s'} \frac{1}{|\mathcal{S}|}|s\rangle\langle s'| \right).$$

4. If you obtained the outcome 1 in each of the r repetitions of step 3, then accept the money. In this case apply $U^\dagger$ and then remove the second and third registers. The final state of the first register is the output quantum money state. If you did not obtain the outcome 1 in each of the r iterations in step 3, then reject the money.

For a quantum money state $|\$_p\rangle$ the associated state $|\$'_p\rangle \sum_s \frac{1}{\sqrt{|\mathcal{S}|}}|s\rangle$ prepared in steps 1 and 2 is unchanged by applying the unitary V in step 3a. Measuring the projector Q in step 3b on this state gives $+1$ with certainty. So for good quantum money, all the measurement outcomes obtained in step 4 are $+1$. We can see that good quantum money passes verification.

Now let us consider the result of applying the above procedure to a general

state $|\phi\rangle$. The first step of the algorithm is to prepare $|\phi'\rangle = U(|\phi\rangle|0\rangle)$. If a single iteration of the loop in step 3 results in the outcome 1 then the final state of the first two registers is

$$\frac{\frac{1}{|\mathcal{S}|}\sum_s \hat{P}_s|\phi'\rangle}{\left\|\frac{1}{|\mathcal{S}|}\sum_{s,t} \hat{P}_s|\phi'\rangle\right\|}.$$

This occurs with probability

$$\left\|\frac{1}{|\mathcal{S}|}\sum_s \hat{P}_s|\phi'\rangle\right\|^2. \tag{4}$$

The entire procedure repeats the loop r times and the probability of obtaining all outcomes 1 is

$$\left\|\left(\frac{1}{|\mathcal{S}|}\sum_s \hat{P}_s\right)^r |\phi'\rangle\right\|^2,$$

in which case the final state (on the first two registers) is

$$\frac{\left(\frac{1}{|\mathcal{S}|}\sum_s \hat{P}_s\right)^r |\phi'\rangle}{\left\|\left(\frac{1}{|\mathcal{S}|}\sum_s \hat{P}_s\right)^r |\phi'\rangle\right\|}.$$

To get some intuition about what states might pass even the first iteration of this test with reasonable probability (4), note that the state $\frac{1}{|\mathcal{S}|}\sum_s \hat{P}_s|\phi'\rangle$ can only have norm close to 1 if most of its terms add coherently. In other words, most of the $\hat{P}_s|\phi'\rangle$ must be nearly the same (they are all exactly the same for good quantum money).

Since $\frac{1}{|\mathcal{S}|}\sum_s \hat{P}_s = \hat{B}$ is our Markov matrix, the set of states that pass all of the rounds is directly related to the mixing properties of the Markov chain—these states correspond to eigenvectors of $\hat{B}$ with eigenvalues close to 1.

3.3 Security of the money scheme

We have shown that the quantum money states $|\$_p\rangle$ can be efficiently generated and pass verification with high probability. In this section we discuss why we believe this quantum money is hard to forge. We consider four possible types of attack against our quantum money.

First, an attacker might measure a valid quantum money state $|\$_p\rangle$ to learn a grid diagram with Alexander polynomial p and then generate a superposition containing that diagram that passes verification. One such state is the (correctly weighted) superposition over grid diagrams equivalent to the measured diagram. If an attacker could do this, the attacker's algorithm could be used to solve grid diagram equivalence, i. e. the problem of telling whether or not two grid diagrams represent the same link. This is believed to be a hard problem on average, even

for quantum computers. In fact, even deciding whether or not a grid diagram represents the unknot is conjectured to be hard.

Second, there are likely to exist grid diagrams of dimension $2\bar{D}$ (our maximum grid dimension) where no allowed grid move reduces the dimension (this follows from the fact that every link has a minimum dimension of grid diagrams that represent it). Because we disallow moves which increase the dimension above $2\bar{D}$, the Markov chain starting from one of these grid diagrams with dimension $2\bar{D}$ will only mix over a small set of diagrams with the same dimension. Uniform superpositions over these sets will pass step 3 of the verification procedure. Step 2 of the verification procedure is designed to reject such superpositions.

Third, if the Markov chain does not mix well, then there will be eigenstates of the matrix $\hat{B}$ with eigenvalues near $+1$. For such eigenstates there may be counterfeit quantum money states which will pass verification with nonnegligible probability. We do not know if these states exist, and even if they do we do not know how to make them. In fact even the simpler question, of finding two equivalent diagrams which require a super-polynomial number of grid moves to go from one to the other (or proving such diagrams do not exist) seems hard.[4]

Fourth, the attacker could use a valid money state $|\$_p\rangle$ and attempt to generate $|\$_p\rangle \otimes |\$_p\rangle$ (or some entangled state on two registers where each register would pass verification). Such an attack worked to forge product state quantum money using quantum state restoration [5]. However, in that case the attack works because the product state money has an embedded classical secret that the attacker can learn to forge the state. In the present work, there is no obvious secret hidden in the money or in the verification algorithm that an attacker could use.

The list above comprises all the lines of attack we were able to think of.

3.4 Why not quantum money from graphs?

We now discuss an alternative quantum money scheme that is similar to the one presented in this paper, but which is based on graphs rather than knots. In practice, it is easy to find the isomorphism relating most pairs of isomorphic graphs; this is enough to render graph based quantum money unusable. The knot based quantum money is an analog of this scheme (translating various concepts from graphs to knots) but we believe it is secure since knot equivalence is believed to be difficult on average.

In the graph based quantum money scheme, the Hilbert space has basis vectors which encode adjacency matrices of graphs on n vertices. The mint generates the state

$$\sum_{\text{Adjacency matrices } A} |A\rangle|0\rangle.$$

[4] Hass and Nowik [7] recently gave the first example of a family of knots that are equivalent to the unknot and require a quadratic number of Reidemeister moves to untangle (previous lower bounds were only linear).

The mint then computes the eigenvalues of the adjacency matrix into the second register and measures the second register to obtain a particular spectrum $R = \{\lambda_1, ...\lambda_n\}$. The resulting state is

$$|\$_R\rangle|R\rangle = \sum_{A \text{ with spectrum } R} |A\rangle|R\rangle.$$

The quantum money is the state $|\$_R\rangle$ and the serial number encodes the spectrum R. The verification procedure is based on a classical Markov chain that, starting from a given adjacency matrix A, mixes to the uniform distribution over adjacency matrices A' that represent graphs isomorphic to A.

Given two adjacency matrices A_0 and $\sigma A_0 \sigma^{-1}$ that are related by the permutation $\sigma \in S_n$, the forger can usually efficiently find the permutation σ (we assume for simplicity that A_0 has no automorphisms so this permutation is unique). We now show how this allows a counterfeiter to forge the graph based quantum money. The forger first measures the state $|\$_R\rangle$ in the computational basis, obtaining a particular adjacency matrix A with the spectrum R. Starting from this state, the forger adjoins two additional registers (one which holds permutations and one which holds adjacency matrices) and, applying a unitary transformation, the forger can produce the state

$$\sum_{\pi \in S_n} |A\rangle|\pi\rangle|\pi A \pi^{-1}\rangle.$$

Now the forger can use the procedure which solves graph isomorphism to uncompute the permutation which is held in the second register, producing the state

$$|A\rangle|0\rangle \sum_{\pi \in S_n} |\pi A \pi^{-1}\rangle.$$

The state of the third register will pass verification. The forger can repeat this procedure using the same adjacency matrix A to produce many copies of this state which pass the verification as quantum money.

Return to the knot based quantum money. Given two grid diagrams G_1 and G_2 which represent the same link, we believe it is hard (on average) to find a sequence of grid moves which transform G_1 into G_2. Even given an oracle for the decision problem of determining whether two links are equivalent, we do not know how to find a sequence of Reidemeister moves relating the links. We hope this discussion has motivated some of the choices we have made for our knot based quantum money.

4 Conclusions

Forge it if you can.

5 Acknowledgments

We thank Louis Kauffman, Haynes Miller and Tom Mrowka for valuable discussions on knot theory. This work was supported in part by funds provided by the U.S. Department of Energy under cooperative research agreement DE-FG02-94ER40818, the W. M. Keck Foundation Center for Extreme Quantum Information Theory, the U.S. Army Research Laboratory's Army Research Office through grant number W911NF-09-1-0438, the National Science Foundation through grant number CCF-0829421, the DoD through the NDSEG Fellowship Program, the Natural Sciences and Engineering Research Council of Canada, and Microsoft Research.

References

[1] S. Aaronson. Quantum copy-protection and quantum money. In *Computational Complexity, Annual IEEE Conference on*, pages 229–242, 2009.

[2] Dorit Aharonov and Amnon Ta-Shma. Adiabatic quantum state generation and statistical zero knowledge, 2003.

[3] J. W. Alexander. Topological invariants of knots and links. *Transactions of the American Mathematical Society*, 30(2):275–306, 1928.

[4] C.H. Bennett, G. Brassard, S. Breidbart, and S. Wiesner. Quantum cryptography, or unforgeable subway tokens. In *Advances in Cryptology–Proceedings of Crypto*, volume 82, pages 267–275, 1983.

[5] Edward Farhi, David Gosset, Avinatan Hassidim, Andrew Lutomirski, Daniel Nagaj, and Peter Shor. Quantum state restoration and single-copy tomography. 2009.

[6] Lov Grover and Terry Rudolph. Creating superpositions that correspond to efficiently integrable probability distributions, 2002.

[7] J. Hass and T. Nowik. Unknot diagrams requiring a quadratic number of Reidemeister moves to untangle. *Discrete and Computational Geometry*, pages 1–5.

[8] Andrew Lutomirski, Scott Aaronson, Edward Farhi, David Gosset, Avinatan Hassidim, Jon Kelner, and Peter Shor. Breaking and making quantum money: toward a new quantum cryptographic protocol, 2010.

[9] Michele Mosca and Douglas Stebila. Uncloneable quantum money, 2006.

[10] Michele Mosca and Douglas Stebila. A framework for quantum money, 2007.

[11] Michele Mosca and Douglas Stebila. Quantum coins, 2009.

[12] Douglas Stebila. Classical authenticated key exchange and quantum cryptography, 2009.

[13] Yuuki Tokunaga, Taisuaki Okamoto, and Nobuyuki Imoto. Anonymous quantum cash, 2003.

[14] Stephen Wiesner. Conjugate coding. *SIGACT News*, 15(1):78–88, 1983.

A Details of the Markov chain on planar grid diagrams

The Markov chain in section 3.2.1 acts on the configuration space of pairs (G, i) where $1 \leq i \leq q(d(G))$. Here we describe an algorithm to implement this Markov chain. For each update of the state of the Markov chain, we first choose several uniformly random parameters:

$$\begin{aligned} j: &\quad \text{an integer from 1 to 8} \\ w: &\quad \text{an integer from 0 to } q_{max}^2 \\ x, y: &\quad \text{integers from 0 to } 2\bar{D} - 1 \\ k: &\quad \text{an integer from 0 to 3} \end{aligned}$$

where q_{max} is the maximum value attained by the function $q(d)$ for $d \in \{2, ..., 2\bar{D}\}$. We then update the configuration (G, i) in a manner that depends on the values of these variables, where j picks the type of the update being performed, and the other variables are used as parameters:

- If $j = 1$, set
$$i \leftarrow (i + w) \mod q(d(G))$$
Leave G unchanged. This is the only move which changes i, and it is used to mix between different labels.

- If $j = 2$, cyclically permute the columns of G by moving each column to the right by one. Leave i unchanged.

- If $j = 3$, cyclically permute columns of G by moving each column to the left by one. Leave i unchanged.

- If $j = 4$, cyclically permute the rows of G by moving each row up by one. Leave i unchanged.

- If $j = 5$, cyclically permute the rows of G by moving each row down by one. Leave i unchanged.

- If $j = 6$ and $x+1 < d\,(G)$, then check whether transposing columns x and $x + 1$ is a legal move (as defined in section 2.3). If so, transpose them; otherwise, do nothing. Leave i unchanged.

- If $j = 7$ and $y + 1 < d\,(G)$, then check whether transposing rows y and $y + 1$ is a legal move (as defined in section 2.3). If so, transpose them; otherwise, do nothing. Leave i unchanged.

- If $j = 8$, $k = 0$, $x, y < d\,(G)$, and there is a marker (X or O) at position (x, y), then consider stabilizing by adding markers forming an L to the upper right of position (x, y). In this case, construct G' from G by adding a new column to the right of x and a new row above y, deleting the original marker, adding markers of the same type at $(x + 1, y)$ and $(x, y + 1)$, and adding a marker of the opposite type at $(x + 1, y + 1)$. Then if $i \leq q(d(G'))$, set $(G, i) \leftarrow (G', i)$. If not, leave the configuration unchanged.

- If $j = 8$, $k = 0$, $x, y < d\,(G) - 1$, there is no marker at (x, y), and there are markers at positions $(x + 1, y + 1)$, $(x + 1, y)$, $(x, y + 1)$, then consider destabilizing by removing markers forming an L to the upper right of position (x, y). In this case, construct G' from G by removing column $x + 1$ and row $y + 1$ (thus removing those three markers) and adding a new marker of the appropriate type at (x, y) (measured after deletion of the row and column). If $i \leq q(d(G'))$, set $(G, i) \leftarrow (G', i)$. If not, leave the configuration unchanged.

- If $j = 8$, and $k \in 1, 2, 3$, rotate the grid diagram by $90k$ degrees counterclockwise and then apply the update rule corresponding to $j = 8, k = 0$ with the same (x, y). After applying the update, rotate the diagram back by the same amount.

The parameter j determines the type of move. The first move ($j = 1$) is used to permute between different labels of the same graph. It's the only move which changes the labels. Moves 2–5 are cyclic permutations of rows and columns, while moves 6 and 7 are transpositions. Finally, $j = 8$ stabilizes or destabilizes an L in a direction ($\urcorner\,\lrcorner\,\llcorner$ or $\ulcorner$) depending on k (0, 1, 2, or 3 respectively). In all the moves x is treated as column index (which describes where to apply the move), and y is treated as a row index.

For fixed (j, w, x, y, k), the update of the state is an easily computable, invertible function (in other words it is a permutation of the configuration space). This is easy to see for any of the moves with $j \in \{1, ..., 7\}$. One can check that each move with $j = 8$ is its own inverse. This justifies our assertion that the Markov matrix B can be written in the form of equation 3.

In the notation of Section 3.2, $s = (j, w, x, y, k)$ and

$$\mathcal{S} = \{1, \ldots, 8\} \times \{0, \ldots, q_{\max}^2\} \times \{0, \ldots, 2\bar{D} - 1\} \times \{0, \ldots, 2\bar{D} - 1\} \times \{0, 1, 2, 3\}.$$