Breaking and making quantum money

Scott Aaronson[1] Edward Farhi[2] David Gosset[3] Jonathan Kelner[4] Avinatan Hassidim[5] Andrew Lutomirski[6] Peter Shor[7]

[1−7]Massachusetts Institute of Technology, Cambridge, MA 02139

aaronson@csail.mit.edu farhi@mit.edu dgosset@mit.edu kelner@mit.edu avinatanh@gmail.com luto@mit.edu shor@math.mit.edu

Abstract: Quantum money is a cryptographic protocol in which a bank can create quantum states which anyone can verify but no one except possibly the bank can clone or forge. Aaronson [1] proposed a scheme for this, which we show to be insecure.

On the positive side, we propose a new scheme, and conjecture that it is secure. Our scheme is inherently quantum, in the sense that there are no classical secrets involved, and the only randomness comes from measurements, whose results are made public. This new scheme has the additional property that not even the bank can make two instances of quantum money with the same "serial number."

Keywords: quantum money; cryptography; random matrices; and markov chains

1 Introduction

Ever since there's been money, there have been people trying to counterfeit it. Preventing counterfeiting is one of the oldest "security problems" facing human civilization. The problem attracted the attention of no less than Isaac Newton, who, in his role as Master of the Mint, redesigned English coins by adding milled edges that were difficult for counterfeiters to reproduce. (Newton also personally oversaw the hangings of many counterfeiters.) Today, paper money comes equipped with an arsenal of anti-counterfeiting measures, including holograms, embedded strips, "microprinting," and special inks that look different depending on the angle.

Yet it should surprise no one that, as mints have become more technically sophisticated, so have counterfeiters—leading to an arms race with no obvious winner. From a cryptographer's perspective, the problem is that unforgeable cash seems theoretically impossible—a criminal organization could in principle copy whatever printing technology a government can build. Furthermore, if we abstract away the printing details, we can see a bill as just a string of 1's and 0's, and we know that any information that can be read and copied an unlimited number of times. Of course, credit card companies (and services such as PayPal) get around this problem by having a trusted bank authorize every transaction. But secure digital cash without such a "middleman"—that is, with the same flexibility and convenience as ordinary cash—would seem to be a fundamental impossibility.

In 1969, Wiesner [2] wrote a remarkable paper, pointing out that the No-Cloning Theorem raises the possibility of uncopyable cash: bills whose authenticity would be guaranteed by quantum physics.[1] Here's how Wiesner's scheme works: besides an ordinary serial number, each bill would contain (say) a few hundred photons, which the central bank "polarized" in random directions when it issued the note. The bank, in a massive database, remembers the polarization of every photon on every bill ever issued. If you want to verify that a bill is genuine, you take it to the bank, and the bank uses its knowledge of the polarizations to measure the photons. On the other hand, the No-Cloning Theorem ensures that someone who *doesn't* know the polarization of a photon can't produce more photons with the same polarizations. Indeed, copying a bill can succeed with probability at most $(5/6)^n$, where n is the number

[1]This is the same paper that introduced the idea of quantum cryptography. Unfortunately, Wiesner's paper was not published until the 1980s; the field of quantum computing and information (to which it naturally belonged) had not yet been invented.

of photons per bill.

Wiesner's scheme has a serious drawback, namely that the bank is the only one who can verify that bill as genuine. Ideally, *printing* bills ought to be the exclusive prerogative of the bank, but the *checking* process ought to be open to anyone. But is it possible, even in principle, to have quantum money satisfying all three requirements:

(1) the bank can print it,

(2) anyone can verify it, and

(3) no one (except possibly the bank) can copy it.

We call such a scheme a *public-key quantum money scheme*, by analogy with public-key cryptography. Perhaps surprisingly, the question of whether public-key quantum money schemes are possible has remained open for forty years, from Wiesner's time till today.

Recently Aaronson [1] revisited the question, and proposed an explicit candidate scheme (henceforth - Stabilizer Money) for public-key quantum money. The scheme has quite a few parameters, but its security is based on the assumption that given a set of m stabilizer measurements which is composed of a large set of random stabilizer measurements and a small set of stabilizer measurements which stabilize the same state (and thus pairwise commute), it is hard to identify the small set.

To make the success probability of the verifier large, a quantum bill consists of ℓ independent sets of measurements, and the bank has ℓ secrets, namely which measurements form the small commuting subset of each one of the ℓ sets. The verifier then does ℓ independent tests, and a Chernoff bound shows that the verification works with high probability.

We prove that the money is insecure, by showing that one of two cases hold, for each one of the ℓ sets

1. If the small commuting set is "too small" (less than $\sqrt{m}/16$) then the test performed by the verifier on this set is too weak, and one can find another money state which would pass the test.

2. If the set of the small commuting set is "too big" (more than $\sqrt{m}/100$) then one can find it.

The hard part of the attack is the second regime, and the tools we provide there may be of independent interest. Consider the graph, whose vertices are the m measurements, and two vertices share an edge iff the measurements commute. The small subset would correspond to a clique, and the large set of measurements is pseudorandom. Unfortunately, one can not apply the clique finding algorithm of Alon, Krivelevich, and Sudakov [3], for finding large planted cliques in random graphs - as the graph is far from random[2]. The [3] algorithm works by finding the second largest eigenvector of the adjacency matrix of the graph. They require a lemma from [5], which asserts that for random graphs the second eigenvector is well behaved. We prove a weaker variant of this lemma for graphs which have logarithmic independence, and show how to use the weaker lemma in Alon et al.'s proof and get a slightly weaker result. We then argue that with high probability the measurement graph has the required independence properties, and thus the clique can be found.

The main contribution of this paper is a new quantum money scheme, which, unlike previous schemes [1, 2, 6], is not based on any secret. In our new scheme, the bank prepares quantum money states by

(1) preparing a uniform superposition over all $n + mr$-bit strings (here r is logarithmic in m),

(2) measuring a collection of randomly-chosen hash functions $h_1, \dots, h_m$ with $m \approx \sqrt{n}$—which collapses the state from a uniform superposition over all $x \in \{0,1\}^{n+mr}$ to a uniform superposition over all x such that $h_1(x), \dots, h_m(x)$ take on their measured values—and then

(3) distributing the partially-collapsed state $|\psi\rangle$ that remains, along with the m-bit string $h_1(x), \dots, h_m(x)$ and a digital signature thereof.

Note that it is easy to generate a classical state which satisfies all the hash functions. However, we conjecture that it is hard to generate a uniform superposition over such states. To verify the

[2]One can not use [4] as well.

money, one first checks that the hash functions give the right values (and thus it is a superposition of legal states). To verify that it's the uniform superposition, one applies a Markov process which has the uniform distribution over the legal states as its stationary distribution. One then verifies that the state one is holding didn't change by the process. The main technical challenge here is to prove that one can design a rapidly mixing Markov chain which has the correct stationary distribution (otherwise other superpositions may also pass the test). We design such chain, and show how to perform a polynomial-time quantum measurement that projects onto the stationary distribution.

Let us now point out a conceptually novel feature of our money scheme: *the bank never needed any classical secret (that is, any private random bits) to generate the state $|\psi\rangle$!* So in particular, any counterfeiting strategy based on "learning the secret of how $|\psi\rangle$ was generated"is doomed to failure here. Indeed, not even the bank itself knows how to generate a second copy of $|\psi\rangle$ efficiently. In other words, if we think of the measurement outcomes $h_1(x), \ldots, h_m(x)$ as the classical "serial number" of a bill, then the bank itself doesn't know the serial number of the bill it's about to generate, nor can it generate two bills with the same serial number (even if it wants to).

This feature of our money scheme has no clear analogue in the classical world—depending as it does on the fact that there is no way to "pull the randomness out of a quantum algorithm," in the same way that one can pull the randomness out of a classical randomized algorithm (and view it as a deterministic algorithm with an auxiliary input). Thus, our new scheme underscores just how different the quantum money problem is from classical cryptographic problems, despite superficial similarities.

Related to that point, it remains a major challenge to base the security of a public-key quantum money scheme on any previously-studied (or at least "standard-looking") cryptographic assumption, for example, that some public-key cryptosystem is secure against quantum attack. That challenge was not met for any previous money scheme, nor do we meet it for our new one. Much as we wish it were otherwise, it seems possible that public-key quantum money intrinsically requires a new mathematical "leap of faith," just as public-key cryptography required a new leap of faith when it was first introduced in the 1970s. However, by evading all currently-known classes of attacks, our new scheme increases our confidence that public-key quantum money is possible at all.

2 Brute-force attack against Stabilizer Money for $\epsilon \leq \frac{1}{16\sqrt{m}}$

The brute-force attack against the Stabilizer Money does not clone the money but rather attacks the verification algorithm directly. To understand the attack, it will be helpful to review the verification algorithm.

2.1 Verification of the Stabilizer Money

The Stabilizer money is parameterized by integers n, m and l and by a real number $\varepsilon \in [0,1]$. These are required to satisfy $\frac{n}{\epsilon} \ll m \ll \frac{1}{\epsilon^2} \ll l$. We assume that $m = poly(n)$.

Any instance of quantum money is verified using a classical certificate, which consists of an $m \times l$ table of n qubit Pauli group operators. The (i,j)th element of the table is an operator

$$E_{ij} = (-1)^{b_{ij}} A_1^{ij} \otimes A_2^{ij} ... \otimes A_n^{ij}$$

where each $A_k^{ij} \in \{1, \sigma_x, \sigma_y, \sigma_z\}$ and $b_{ij} \in \{0,1\}$.

We will use one important property of the algorithm that generates the table of Pauli operators: with the exception of the fact that $-I^{\otimes n}$ cannot occur in the table, the distribution of the tables is symmetric under negation of all of the operators.

The verification algorithm works by choosing, for each i, a random $j(i) \in [m]$. The verifier then measures

$$M = \frac{1}{l} \sum_i I^{\otimes i-1} \otimes E_{i,j(i)} \otimes I^{\otimes m-i}. \qquad (1)$$

The algorithm accepts iff the outcome is greater than or equal to $\frac{\epsilon}{2}$. Note that measuring the operator M is equivalent to measuring the operator $E_{i,j(i)}$ for each register $i \in [l]$ and then averaging the results, since the measurements on different registers commute.

To better understand the statistics of the operator M, we consider measuring an operator $E_{ij(i)}$ on a state ρ_i, where $j(i) \in [m]$ is chosen uniformly at random. The total probability $p_1(\rho_i)$ of measuring a $+1$ is given by

$$\begin{aligned} p_1(\rho_i) &= \frac{1}{m}\sum_{j=1}^{m} Tr\left[\left(\frac{1+E_{ij(i)}}{2}\right)\rho_i\right] \\ &= \frac{1+\mathrm{Tr}\left[H^{(i)}\rho_i\right]}{2}, \end{aligned}$$

where (for each $i \in [l]$) we have defined the Hamiltonian

$$H^{(i)} = \frac{1}{m}\sum_{j=1}^{m} E_{ij}.$$

2.2 Attacking the verifier

For $\epsilon \le \frac{1}{16\sqrt{m}}$ and with high probability in the table of Pauli operators, we can efficiently generate a state that passes verification with high probability.

We assume we are given a table of operators E_{ij} generated as in Aaronson's scheme, and our goal will be to produce an nl qubit mixed state ρ which is accepted by Aaronson's verifier with all but exponentially small (in n) probability. For some choices of E_{ij} our attack may fail but we will show that the probability that such a table of operators is selected in Aaronson's scheme is exponentially small.

The idea of the attack is simple, and we now describe how it works in the overwhelmingly likely event that the table of operators E_{ij} does not cause it to fail. We use an algorithm, described below, to independently generate an n qubit mixed state ρ_i on each register $i \in [l]$. At least $1/4$ of these states ρ_i will have the property that

$$\mathrm{Tr}[H^{(i)}\rho_i] \ge \frac{1}{4\sqrt{m}} + O(\frac{1}{m^2}), \tag{2}$$

and the rest have

$$p_1(\rho_i) \ge \frac{1}{2} + O(\frac{1}{m}), \tag{3}$$

which implies that,

$$\mathop{\mathbb{E}}_i p_1(\rho_i) \ge \frac{1}{2} + \frac{1}{8\sqrt{m}} + O(\frac{1}{m^2}).$$

We use the state

$$\rho = \rho_1 \otimes \rho_2 \otimes ... \otimes \rho_l$$

as our forged quantum money. The mean value of the operator M (from equation 1) in this state is at least $\frac{1}{4}(\frac{1}{4\sqrt{m}} + O(\frac{1}{m^2})) + \frac{3}{4}O(\frac{1}{m})$, and the probability of measuring an energy less than $\frac{1}{32\sqrt{m}}$ (which is the case where the state is not accepted) is exponentially small for m sufficiently large. Therefore the forged money state ρ is accepted by Aaronson's verifier with probability that is exponentially close to 1 if $\epsilon \le \frac{1}{16\sqrt{m}}$.

Before describing our algorithm to generate the states ρ_i, we must understand the statistics (in particular, we consider the first two moments) of each $H^{(i)}$ on the fully mixed state $\frac{I}{2^n}$. We will assume that, for $j \neq k$, $E_{i,j} \neq E_{i,k}$. We also assume that the operators $\pm I \otimes I \otimes I ... \otimes I$ do not appear in the list. Both of these assumptions are satisfied with overwhelming probability. We then have for the first and second moments of $H^{(i)}$

$$\mathrm{Tr}\left[H^{(i)}\frac{I}{2^n}\right] = 0,$$

and

$$\begin{aligned} &\mathrm{Tr}\left[\left(H^{(i)}\right)^2\frac{I}{2^n}\right] \qquad (4) \\ =\;& 2^{-n}\,\mathrm{Tr}\left[\frac{1}{m^2}\sum_j (E_{i,j})^2 + \frac{1}{m^2}\sum_{j\neq k} E_{i,j}E_{i,k}\right] \\ =\;& \frac{1}{m}. \qquad (5) \end{aligned}$$

Now let us define f_i to be the fraction (out of 2^n) of the eigenstates of $H^{(i)}$ which have eigenvalues in the set $[\frac{1}{2\sqrt{m}}, 1] \cup [-1, -\frac{1}{2\sqrt{m}}]$. Since the eigenvalues of $H^{(i)}$ are bounded between -1 and 1, we have

$$\mathrm{Tr}\left[\left(H^{(i)}\right)^2\frac{I}{2^n}\right] \le f_i + (1-f_i)\frac{1}{4m}.$$

Plugging in equation 5 and rearranging we obtain

$$f_i \ge \frac{3}{4m-1}.$$

We also define g_i to be the fraction of eigenstates of $H^{(i)}$ that have eigenvalues in the set $[\frac{1}{2\sqrt{m}}, 1]$. The distribution (for any fixed i) of $E_{i,j}$ as generated by the bank is symmetric under negation of all the $E_{i,j}$, so with probability at least $1/2$ over the choice of the operators in the row labeled by i, the fraction g_i satisfies

$$g_i \geq \frac{3}{8m-2}. \quad (6)$$

We assume this last inequality is satisfied for at least $1/4$ of the indices $i \in [l]$, for the particular table E_{ij} that we are given. The probability that this is not the case is exponentially small in l.

Ideally, we would generate the states ρ_i by preparing the fully mixed state, measuring $H^{(i)}$, keeping the result if the eigenvalue is at least $\frac{1}{2\sqrt{m}}$, and otherwise trying again, up to some appropriate maximum number of tries. After enough failures, we would simply return the fully mixed state. It is easy to see that outputs of this algorithm would satisfy eq. 2.2 with high probability.

Unfortunately, we cannot efficiently measure the exact eigenvalue of an arbitrary Hermitian operator, but we can use phase estimation, which gives polynomial error using polynomial resources. In appendix A we review the phase estimation algorithm which is central to our procedure for generating the states ρ_i. In section 2.3, we describe an efficient algorithm to generate ρ_i using phase estimation, and, in appendix B, we show that the resulting states, even in the presence of errors due to polynomial-time phase estimation, are accepted by the verifier with high probability, assuming that the table E_{ij} has the appropriate properties.

2.3 Procedure to Generate ρ_i

We now fix a particular $H^{(i)}$ and define $H = \frac{1}{4}H^{(i)}$ so that all the eigenvalues of H lie in the interval $[-\frac{1}{4}, \frac{1}{4}]$. We denote the eigenvectors of H by $\{|\psi_j\rangle\}$ and write

$$e^{2\pi i H}|\psi_j\rangle = e^{2\pi i \phi_j}|\psi_j\rangle,$$

where $\phi_j \in [0, \frac{1}{4}] \cup [\frac{3}{4}, 1]$.

Our attack which can be used to forge Aaronson's quantum money is to do the following procedure for each register $i \in [l]$, producing mixed state outputs $\{\rho_i\}$. We will analyze this procedure in appendix B.

1. Set $k = 1$.
2. Prepare the completely mixed state $\frac{I}{2^n}$.
3. Use the phase estimation circuit to measure the operator $e^{2\pi i H}$. Here the phase estimation circuit (see appendix A) acts on the original n qubits in addition to $q = r + \lceil log(2 + \frac{2}{\delta})\rceil$ ancilla qubits, where we choose
$$r = \lceil log(20m)\rceil \qquad \delta = \frac{1}{m^3}.$$
4. Accept the resulting state (of the n qubit register) if the measured phase $\phi' \doteq \frac{z}{2^q}$ is in the interval $[\frac{1}{8\sqrt{m}} - \frac{1}{20m}, \frac{1}{2}]$. In this case stop and output the state of the first register. Otherwise set $k = k + 1$.
5. If $k = m^2 + 1$ then stop and output the fully mixed state. Otherwise go to step 2.

3 Insecurity of the Stabilizer Money for $\epsilon \geq \frac{c}{\sqrt{m}}$

In this section, we will describe how to forge the Stabilizer Money when the number of commuting measurements is at least $c\sqrt{m}$ for any constant $c > 0$. We will consider each column of the table separately. For the i^{th} column, let $M = M_i$ be the list of possible measurements for $\psi = \psi_i$, and let $K = K_i$ denote the set of commuting measurements that stabilize ψ. Set $k = |K|$ and $m = |M|$. We will first consider the case $k > 100\sqrt{m}$, and we will then show how to reduce the case $k > c\sqrt{m}$ to this case for any constant $c > 0$. The algorithm we present has success probability $4/5$ over the choice of the random measurements. We have not attempted to optimize this probability, and it could be improved with a more careful analysis.

We begin by casting our question as a graph problem. Let G be a graph whose vertices correspond to the m measurements, and connect vertices i and j if and only if the corresponding measurements commute. The set K now forms a clique, and we aim to find it.

In general, it is intractable to find the largest clique in a graph. In fact, it is NP-hard even to approximate the size of the largest clique within $n^{1-\epsilon}$, for any $\epsilon > 0$ [7]. However, if the graph is obtained by planting a clique of size $\epsilon\sqrt{m}$ in an (Erdös-Rényi) random graph drawn from $G(m, 1/2)$, Alon, Krivelevich, and Sudakov showed that one can find the clique in polynomial time with high probability [3]. Unfortunately, the measurement graph G is not drawn from $G(m, 1/2)$, so we cannot directly apply their result. However, we shall show that G is sufficiently random that a modified version of their approach can be made to go through.

3.1 Properties of the Measurement Graph

To analyze G, it will be convenient to use a linear algebraic description of its vertices and edges. Recall that any stabilizer measurement on n qubits can be described as a vector in $\mathbb{F}_2^{2n}$ as follows:

- for $j \leq n$, set the j^{th} coordinate to 1 if and only if an X is applied to the j^{th} coordinate, and
- for $n < j \leq 2n$, set the j^{th} coordinate to 1 if and only if a Z is applied to the $(j-n)^{\text{th}}$ coordinate.

For $v, w \in \mathbb{F}_2^{2n}$, let

$$\langle v, w \rangle = v^T \begin{pmatrix} \mathbf{0}_n & I_n \\ I_n & \mathbf{0}_n \end{pmatrix} w,$$

where I_n and $\mathbf{0}_n$ are the $n \times n$ identity and all-zeros matrices, respectively. It is easy to check that the measurements corresponding to v and w commute if and only if $\langle v, w \rangle = 0$ (over $\mathbb{F}_2$).

With this notation, we can associate a vector s_u to each vertex u, and there is an edge between vertices u and v in G if and only if $\langle s_u, s_v \rangle = 0$. From this description, it follows that the edges of G are dependent, so it is not drawn from $G(m, 1/2)$. To see this, suppose that there is some set of vertices any new measurement v is completely determined by the set of measurements in $v_1, \ldots, v_{2n}$ that commute with v. In particular, for any other fixed measurement u, the existence of an edge from u to v is determined once we know the set of edges between v and the v_is. However, the following lemma will allow us to bound the effect of these dependencies when the number of variables being considered is not too large.

Lemma 1. *Let $v_1, \ldots v_t, u$ be measurements such that $s_{v_1}, \ldots s_{v_t}, s_u$ are linearly independent, and let $x_1, \ldots, x_t \in \{0, 1\}$ be arbitrary. Let v be a random stabilizer measurement such that $\langle s_v, s_{v_i} \rangle = x_i$ for every i and the vectors $s_{v_1}, \ldots, s_{v_t}, s_u, s_v$ are linearly independent. Then*

$$\Pr(\langle s_v, s_u \rangle = 0) = 1/2 \pm O\left(\frac{1}{2^{2(n-t)}}\right).$$

Proof. The vector $s_v \in \{0, 1\}^{2n}$ is chosen uniformly at random from the set of vectors satisfying the following constraints:

1. For every i, we have $\langle s_v, s_{v_i} \rangle = x_i$.
2. The vectors $s_{v_1}, \ldots s_{v_t}, s_u, s_v$ are linearly independent.

Let S_0 denote the set of vectors that satisfy these constraints and have $\langle s_v, s_u \rangle = 0$, and let S_1 be the set of vectors that satisfy these constraints and have $\langle s_v, s_u \rangle = 1$. We have

$$\Pr(\langle s_v, s_u \rangle = 0) = \frac{|S_0|}{|S_0 + S_1|}.$$

The vectors $s_{v_1}, \ldots s_{v_t}, s_u$ are linearly independent, so there are 2^{2n-t-1} solutions to the set of equations $\langle s_v, s_u \rangle = 1$ and $\langle s_v, s_{v_i} \rangle = x_i$ for all i. This implies that $|S_1| \leq 2^{2n-t-1}$.

Constraint 2 rules out precisely the set of vectors in the span of $s_{v_1}, \ldots, s_{v_t}, s_u$. This is a $(t+1)$-dimensional subspace, so it contains 2^{t+1} points, and thus $|S_0| \geq 2^{2n-t-1} - 2^{t+1}$. It follows that

$$\begin{aligned} \Pr(\langle s_v, s_u \rangle = 0) &\geq \frac{2^{2n-t-1} - 2^{t+1}}{2^{2n-t} - 2^{t+1}} \\ &= \frac{1}{2} - \frac{1}{2^{2n-2t} - 1} \\ &= \frac{1}{2} - O\left(\frac{1}{2^{2(n-t)}}\right). \end{aligned}$$

Repeating this argument gives the same bound for $\Pr(\langle s_v, s_u \rangle = 1)$, from which the desired result follows. □

3.2 Finding Planted Cliques in Random Graphs

Our algorithm for finding the clique K will be identical to that of Alon, Krivelevich, and Sudakov [3], but we will need to modify the proof of correctness to show that it still works in our setting. In this section, we shall give a high level description of [3] and explain the modifications necessary to apply it to G. The fundamental difference is that Alon et al. rely on results from random matrix theory that use the complete independence of the matrix entries to bound mixed moments of arbitrarily high degree, but we only have guarantees about moments of degree $O(\log m)$. As such, we must adapt the proof to use only these lower order moments.

Let $G(m, 1/2, k)$ be a random graph from $G(m, 1/2)$ augmented with a planted clique of size k, and let A be its adjacency matrix. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m$ be the eigenvalues of A, and let $v_1, \ldots, v_m$ be the corresponding eigenvectors. To find the clique, Alon et al. find the set W of vertices with the k largest coordinates in v_2. They then prove that, with high probability, the set of vertices that have at least $3k/4$ neighbors in W precisely comprise the planted clique.

The analysis of their algorithm proceeds by analyzing the largest eigenvalues of A. They begin by proving that the following two bounds hold with high probability:

- $\lambda_1 \geq \left(\frac{1}{2} + o(1)\right) m$, and
- $\lambda_i \leq (1 + o(1)) \sqrt{m}$ for all $i \geq 3$.

The second of these bounds relies heavily on a result by Füredi and Komlós about the eigenvalues of matrices with independent entries. The independence assumption will not apply in our setting, and thus we will need to reprove this bound for our graph G. This is the main modification that we will require to the analysis of [3].

They then introduce a vector z that has $z_i = (m - k)$ when vertex i belongs to the planted clique, and has $z_i = -k$ otherwise. Using the above bounds, they prove that, when one expands z in the eigenbasis of A, the coefficients of $v_1, v_3, \ldots, v_m$ are all small compared to $||z||$, so z has most of its norm coming from its projection onto v_2. This means that v_2 has most of its weight on the planted clique, which enables them to prove the correctness of their algorithm.

Other than the bound on $\lambda_3, \ldots, \lambda_m$, the proof goes through with only minor changes. The bound on $\lambda_1 = (1 + o(1))m/2$, follows from a simple analysis of the average degree, which holds for the measurement graph as well. The rest of their proof does not make heavy use of the structure of the graph. The only change necessary is to replace various tail bounds on the binomial distribution and Chebyschev bounds with Markov bounds. These weaker bounds result in a constant failure probability and weaker constants, but they otherwise do not affect the proof. (For brevity, we omit the details.) As such, our remaining task is to bound λ_i for $i \geq 3$.

3.3 Bounding $\lambda_3, \ldots, \lambda_m$

To bound the higher eigenvalues of the adjacency matrix, Alon et al. apply the following theorem of Füredi and Komlós [5]:

Lemma 2. *Let R be a random symmetric $m \times m$ matrix in which $R_{i,i} = 0$ for all i, and the other entries are independently set to ± 1 with $\Pr(R_{i,j} = 1) = \Pr(R_{i,j} = -1) = \frac{1}{2}$. The largest eigenvalue of R is at most $m + O(m^{1/3} \log m)$ with high probability.*

We will prove a slightly weaker variant of this lemma for random measurement graphs. Let B be a matrix that is generated by picking m random stabilizer measurements $M_1, \ldots, M_m$ and setting $B_{i,i} = 0$, $B_{i,j} = 1$ if M_i commutes with M_j, and $B_{i,j} = -1$ if M_i anticommutes with M_j. The main technical result of this section will be the following:

Theorem 3. *With high probability, the largest eigenvalue of B is at most $10\sqrt{m}$.*

Alon et al.[3] show how to transform a bound on the eigenvalues of R into a bound on the third largest eigenvalue of A. This reduction does not depend on the properties of G, and it works in our case when applied to B. This gives a bound of $10\sqrt{m}$ on the third largest eigenvalue of the adjacency matrix of G.

The proof of Theorem 3 will rely on the following lemma, which shows that the entries of small powers of the matrix B have expectations quite close to those of R.

Lemma 4. *For $t \leq O(\log m)$,*

$$\mathbb{E}\left[(B^t)_{i,j}\right] = \mathbb{E}\left[(R^t)_{i,j}\right] \pm \frac{1}{2^{\Omega(n-t)}}.$$

Proof of Lemma 4. With high probability, for every subset of vertices U such that $|U| < t \leq O(\log m)$, we have that the set $\{s_u \mid u \in U\}$ is linearly independent over $\mathbb{F}_2$. We condition the rest of our analysis on this high probability event.

We begin by expanding the quantity we aim to bound:

$$\mathbb{E}\left[(B^t)_{i,j}\right] = \mathbb{E}\left[\sum_{\ell_2,\ldots \ell_t}\prod_{\alpha=1}^{t+1} B_{\ell_\alpha,\ell_{\alpha+1}}\right]$$
$$= \sum_{\ell_2,\ldots \ell_t}\mathbb{E}\left[\prod_{\alpha=1}^{t+1} B_{\ell_\alpha,\ell_{\alpha+1}}\right], \qquad (7)$$

where we take set $\ell_1 = i$ and $\ell_{t+1} = j$, and we sum over all possible values of the indices $\ell_2, \ldots, \ell_t$.

We break the nonzero terms in this summation into two types of monomials: those in which every matrix element appears an even number of times, and those in which at least one element appears an odd number of times. In the former case, the monomial is the square of a ± 1-valued random variable, so we have

$$\mathbb{E}\left[\prod_\alpha B_{\ell_\alpha,\ell_{\alpha+1}}\right] = \mathbb{E}\left[\prod_\alpha R_{\ell_\alpha,\ell_{\alpha+1}}\right] = 1,$$

and it suffices to focus on the latter case. By the same reasoning, we can drop any even number of occurrences of an element, so it suffices to estimate the expectations of monomials of degree at most t in which all of the variables are distinct.

Any such monomial in the $R_{i,j}$ has expectation zero by symmetry, so we need to provide an upper bound on terms of the form $\prod_{\alpha=1}^q B_{\ell_\alpha,\ell_{\alpha+1}}$, where $q \le t \le r$ and each matrix element appears at most once.

Consider the probability that $B_{q-1,q} = 1$, where we take the probability over the choice of the $2n$ bit string s_q, given that for any $\alpha \le q$, we have $B_{\alpha,\alpha+1} = x_\alpha$ for some value x_α. We are computing this expectation conditioned on the the s_u being linearly independent, so we can apply Lemma 1. This gives us that

$$\mathbb{E}\prod_{\alpha=1}^{q} B_{\ell_\alpha,\ell_{\alpha+1}}$$
$$= \sum_{x_1,\ldots x_{q-1}} \Pr(\langle s_{\ell_\alpha}, s_{\ell_{\alpha+1}}\rangle = x_\alpha)$$
$$\times \Big\{ \Pr(\langle s_{q-1}, s_q\rangle = 1 | x_1, \ldots x_{q-1})$$
$$- \Pr(\langle s_{q-1}, s_q\rangle = -1 | x_1, \ldots x_{q-1})\Big\}$$
$$\le O\left(\frac{1}{2^{2(n-t)}}\right)\cdot \sum_{x_1,\ldots x_{q-1}} \Pr(\langle s_{\ell_\alpha}, s_{\ell_{\alpha+1}}\rangle = x_\alpha)$$
$$= O\left(\frac{1}{2^{2(n-t)}}\right).$$

There are $n^{O(\log m)}$ terms in the summation of (7), and we have shown that each term is at most $O\left(1/2^{2(n-t)}\right)$, so we obtain

$$\mathbb{E}\left[(B^t)_{i,j}\right] \le O\left(\frac{n^{O(\log m)}}{2^{2(n-t)}}\right) = \frac{1}{2^{\Omega(n)}},$$

as desired. □

We can now use this lemma to prove Theorem 3.

Proof of Theorem 3. Consider a random matrix R, with $R_{i,i} = 0$ and each other cell distributed independently at random according to $\Pr(R_{i,j} = 1) = \Pr(R_{i,j} = -1) = \frac{1}{2}$. Lemma 3.2 of [5] shows that, for $t < m^{1/3}$,

$$\operatorname{Tr}(\mathbb{E}(R^t)) = m^{t/2+1}4^t.$$

For $t \ge 10\log m$, Lemma 4 implies that

$$\operatorname{Tr}(\mathbb{E}(B^t)) = \operatorname{Tr}(\mathbb{E}(R^t)) \pm \frac{1}{2^{\Omega(n-t)}}$$
$$= m^{t/2+1}4^t \pm \frac{1}{2^{\Omega(n-t)}}.$$

Let $\lambda_1 \ge \cdots \ge \lambda_n$ be the eigenvalues of B. For any even t, one has that

$$\operatorname{Tr} B^t = \sum_i \lambda_i^t \ge \lambda_1^t.$$

Applying this relation with $t = 10\log m$ gives:

$$\Pr(\lambda_1 \ge 10\sqrt{m}) = \Pr\left(\lambda_1^t \ge (10\sqrt{m})^t\right)$$
$$\le (10\sqrt{m})^{-t}\mathbb{E}\lambda_1^t \le (10\sqrt{m})^{-t} m^{t/2+1}4^t$$
$$= m\left(\frac{4}{10}\right)^t < 1/m^4. \qquad \square$$

Plugging the bound from Theorem 3 into the argument from the section 3.2 and computing the correct constants yields that the algorithm finds a planted clique in G of size at least $100\sqrt{m}$ with probability $4/5$.

3.4 Finding Cliques of Size $c\sqrt{m}$

To fully break Aaronson's quantum money scheme, we need to find cliques of size $c\sqrt{m}$ for any $c > 0$. In [3], Alon et al. show how to bootstrap the above scheme to work for any c.

The procedure used by Alon et al. is to iterate over all sets of vertices of size $\log(100/c)$, and, for each such set S, to try to find a clique in the graph G_S of the vertices that are connected to all of the vertices in S.

When S is in the planted clique, G_S also contains the clique. However, $|G_S| \approx c|G|/100$, as most of the vertices that are outside the clique are removed. As G_S behaves like a random graph with a planted clique of size $100\sqrt{|G_S|}$, one can find it using the second largest eigenvector.

To use the same algorithm in our case, we apply Lemma 4 with parameter $k + \log 100/c$. This shows that, up to a small additive error, the expected value of the k^{th} power of the adjacency matrix of G_S behaves like the expected value of the k^{th} power of the adjacency matrix of a random graph, which was all that we used in the proof.

4 Quantum money based on random constraint satisfaction

4.1 Definitions and generation of quantum money

We propose a new quantum money protocol. Our quantum money consists of a classical description of a random instance of a constraint satisfaction problem (CSP), a digital signature of the description of the instance, and a quantum state which is the uniform superposition of the solutions of the CSP. In our CSP, some of the bits are covered by 10 clauses and the rest are covered by only one clause; this structure enables us to verify the money. While the CSP we use can be easily solved—in fact, we can even efficiently sample its solutions—we conjecture that generating the uniform superposition of the solutions is hard. Nonetheless, an efficient algorithm can project onto the uniform superposition of solutions, allowing us to efficiently verify the state and ensure that so other state passes verification with better than exponentially small probability.

Our quantum money scheme has several global parameters. These are: a key pair for a digital signature protocol (the bank publishes the public key); a security parameter n; related parameters m and r; a graph G which describes the clause structure of the CSP; and a sequence $(h_1, \ldots, h_m)$ of random well-balanced (in the sense of appendix C.1) binary-valued functions. We suggest using cryptographic hashes for the h_j and setting $m = \lceil 10\sqrt{n} \rceil$ and $r = \lceil 2\log_2(m+r) \rceil$, which ensures that random functions are well-balanced whp. We generate a random 10-regular hypergraph G on with n vertices and m edges. (We suggest that each vertex be incident to 10 hyperedges selected uniformly at random without repetition.) We say that the hyperedge e_i is incident to the vertices with indices $\left\{e_{i,(1)}, \ldots e_{i,(|e_i|)}\right\}$. All of these parameters can be shared between multiple instances of quantum money without weakening the security of the protocol.

An instance of quantum money is a classical vector $y \in \mathbb{Z}_2^m$, a quantum state $|\psi_y\rangle$ on $n + mr$ qubits, and the bank's digital signature $\text{Sig}(y)$. Using the graph G and the functions $h_1, \ldots, h_m$, we define a (classical) constraint satisfaction problem on $n + mr$ bits arranged into a vector $v \in \mathbb{Z}_2^n$ and a matrix $w \in \mathbb{Z}_2^{m \times r}$. The bits in v (the *inner* bits) correspond to the vertices of G and the bits in w (the *outer* bits) do not. The constraints are $\forall j . h_j\left(v_{e_{j,(1)}}, \ldots, v_{e_{j,(|e_i|)}}, w_{j,1}, \ldots, w_{j,r}\right) = y_j$. This is a random CSP where each clause allows approximately half of the possible assignments of its bits. We refer to the set of satisfying assignments of the CSP as S. We use the classical states of (v, w) as our computational basis, and we define

$$|\psi_y\rangle = \frac{1}{\sqrt{|S|}} \sum_{(v,w) \in S} |(v, w)\rangle.$$

The bank can efficiently produce tuples $(y, \text{Sig}(y), |\psi_y\rangle)$ by generating the uniform superposition $|+\rangle^{\otimes(n+mr)}$ over *all* states, measuring $y_i = h_i(\cdots)$ for each i, and then signing the resulting vector y. (Note that any adversary can use the same procedure to generate pairs $(y, |\psi_y\rangle)$ without the signature, so the bank does not weaken the money by reusing the graph and random functions.)

To verify the quantum money, anyone can project onto $|\psi_y\rangle$ using an approximation with exponentially small error, described below.

The quantum state $|\psi_y\rangle$ is the uniform superposition of a random CSP. Although the CSP can be solved easily, we conjecture that no computationally bounded algorithm can generate the state $|\psi_y\rangle \otimes |\psi_y\rangle$ for any y; this means that no one can clone the quantum money and, in fact, that not even the bank can generate two identical-looking pieces of quantum money. We further conjecture that replacing the random functions h_j with (different) cryptographic hashes does not weaken the protocol's security. The generic approach to creating such a uniform superposition is a problem known as Q-sampling, which is SZK-hard and therefore conjectured to require exponential time, even on a quantum computer [8].

4.2 A rapidly mixing Markov chain over CSP solutions

The core of our verification algorithm is a rapidly mixing Markov chain over CSP solutions with a uniform stationary state. Each state is a solution to the CSP, and the transition rule is:

1. With probability $\frac{1}{4}$, choose an inner bit i uniformly at random. If the current state with v_i negated is a solution to the CSP, then negate that bit; otherwise, no nothing. (The bit i is covered by 10 clauses, so the new trial state will be a solution to the CSP with probability approximately $\frac{1}{1024}$.)

2. With probability $\frac{1}{4}$, choose a clause j at random and a bit string $\left(w'_{j,1},\ldots,w'_{j,r}\right)$ uniformly at random. If the current state with the jth row of w set to $\left(w'_{j,1},\ldots,w'_{j,r}\right)$ is a solution to the CSP, then make that change; otherwise, do nothing. (The new trial state will be a solution with probability approximately $\frac{1}{2}$.)

3. With probability $\frac{1}{2}$, do nothing. This rule ensures that the Markov chain is lazy.

Intuitively, the presence of the auxiliary bits w ensures that the rule that flips bits in v does not get stuck. In appendix C.2, we prove that this Markov chain mixes rapidly in the sense that it has a spectral gap of at least $1/\left(\mathrm{poly}(n,m,r)\right)$.

4.3 From a rapidly mixing Markov chain to a projector

The first step of projecting onto $|\psi_y\rangle$ is to measure all of the hash functions—this projects onto the span of all states of our Markov chain. It remains to project onto the stationary state $|\psi_y\rangle$.

Our Markov chain's Markov operator M has a large spectral gap, so all but one eigenvalue is bounded in $[-1+\epsilon,1-\epsilon]$ for some known xϵ. We would like to implement a measurement that projects onto the $+1$ eigenvector.

We can easily represent the Markov update rule as a uniform selection over permutations in the computational basis. We do this by selecting, uniformly at random, an integer α from 1 to 4, an integer i from 1 to n, an integer j from 1 to m, and an r-bit string z. If $\alpha = 1$ and the current state with bit i flipped satisfies the CSP, then we flip bit i. If $\alpha = 2$ and the current state with the jth row of w xored by z satisfies the CSP, xor the jth row of w by z. If $\alpha \geq 3$, do nothing. For any $s = (\alpha, i, j, z)$, this operation is its own inverse, so we can efficiently implement it unitarily on a quantum computer. We refer to this unitary operation as U_s, and we note that $M = \frac{1}{N}\sum_{s=1}^{N} U_s$, where N is the size of the domain of s. We define a new unitary operator on two registers $U = \sum_s (I\otimes|s\rangle)U_s(I\otimes\langle s|)$, which is just an update of the first register controlled by the second register.

Given some initial state, we can add an ancilla in a uniform superposition over all s, update the register, and project the ancilla back into the uniform superposition, aborting if the ancilla is found to be orthogonal to the uniform superposition. This implements the operator

$$\begin{aligned}&\left(I\otimes\frac{1}{\sqrt{N}}\sum_s\langle s|\right)U\left(I\otimes\frac{1}{\sqrt{N}}\sum_s|s\rangle\right)\\ =\;&\frac{1}{N}\sum_s U_s = M\end{aligned}$$

This operation can be implemented with one call to controlled-U_s and overhead logarithmic in N. If we repeat this process t/ϵ times, we obtain $M^{t/\epsilon} = |\pi\rangle\langle\pi| + O(e^{-t})$, where $|\pi\rangle$ is the stationary state of the Markov operator, which is what we wanted.

This construction has the caveat that, if the projection fails (i.e. the input $|\psi\rangle$ is orthogonal to $|\pi\rangle$), the final state is not $(1-|\pi\rangle\langle\pi|)|\psi\rangle$. This can be corrected by copying the answer and uncomputing the measurement, but, as we do not care about the final state of bad quantum money, we do not need this correction.

We have $\epsilon = \left[3\cdot 2^{18}\left(n+2mn+2\right)m\left(n+1\right)\right]^{-1}$, so we can verify quantum money with error $O\left(e^{-t}\right)$ using $t\left[3\cdot 2^{18}\left(n+2mn+2\right)m\left(n+1\right)\right]$ calls to controlled-U_s, which can be implemented in time polynomial in all of the parameters of interest.

References

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

[2] S. Wiesner. Conjugate coding. *SIGACT News*, 15(1):78–88, 1983. Original manuscript written circa 1970.

[3] N. Alon, M. Krivelevich, and B. Sudakov. Finding a large hidden clique in a random graph. In *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, pages 594–598. Society for Industrial and Applied Mathematics Philadelphia, PA, USA, 1998.

[4] Uriel Feige and Robert Krauthgamer. Finding and certifying a large hidden clique in a semirandom graph. *Random Struct. Algorithms*, 16(2):195–208, 2000.

[5] Z. Füredi and J. Komlos. The eigenvalues of random symmetric matrices. *Combinatorica*, 1(3):233–241, 1981.

[6] 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.

[7] David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(1):103–128, 2007.

[8] D. Aharonov and A. Ta-Shma. Adiabatic Quantum State Generation. *SIAM Journal on Computing*, 37:47, 2007.

[9] M.A. Nielsen and I.L. Chuang. Quantum computation and quantum information. 2000.

[10] R. Matwani and P. Raghavan. Randomized Algorithms, 1995.

[11] P. Diaconis and D. Stroock. Geometric bounds for eigenvalues of Markov chains. *The Annals of Applied Probability*, pages 36–61, 1991.

A Review of the Phase Estimation Algorithm

In this section we review the phase estimation algorithm as described in [9]. We are interested in using this algorithm to measure the eigenvalues of the operator $e^{2\pi i H}$. The phase estimation circuit takes as input an integer r and a parameter δ and uses

$$q = r + \lceil log(2 + \frac{2}{\delta})\rceil$$

ancilla qubits. When used to measure the operator $e^{2\pi i H}$, phase estimation requires as a subroutine a circuit which implements the unitary operator $e^{2\pi i H t}$ for $t \leq 2^r$, which can be approximated efficiently if $2^r = poly(n)$. This approximation of the Hamiltonian time evolution incurs an error which can be made polynomially small in n using polynomial resources (see for example [9]). We therefore neglect this error in the remainder of the discussion. The phase estimation circuit, when applied to an eigenstate $|\psi_j\rangle$ of H such that

$$e^{2\pi i H}|\psi_j\rangle = e^{2\pi i \phi_j}|\psi_j\rangle,$$

and with the q ancillas initialized in the state $|0\rangle^{\otimes q}$, outputs a state

$$|\psi_j\rangle \otimes |a_j\rangle$$

where $|a_j\rangle$ is a state of the ancillas. If this ancilla register is then measured in the computational basis, the resulting q bit string z will be an approximation to ϕ_j which is accurate to r bits , with probability of failure $\leq \delta$ in the sense that

$$Pr\left(|\phi_j - \frac{z}{2^q}| > \frac{1}{2^r}\right) \leq \delta. \qquad (8)$$

In order for this algorithm to be efficient in this case we must choose r and δ so that $2^r = poly(n)$ and $\delta = \frac{1}{poly(n)}$.

B Analysis of the Procedure to Generate ρ_i

In section 2.2 we considered two cases depending on whether or not the inequality 6 is satisfied for a given register i. Our analysis of the procedure outlined in section 2.3 considers these two cases separately. The first case is when, choosing ϕ_p uniformly

$$\Pr\left(\frac{1}{4} \geq \phi_p \geq \frac{1}{8\sqrt{m}}\right) \geq \frac{3}{8m-2} \qquad (9)$$

The second case is when this is not true. This inequality is satisfied for at least 1/4 of the indices $i \in [l]$ with all but exponential probability.

We label each eigenstate of H as either "good" or "bad" according to its energy. We say an eigenstate $|\psi_j\rangle$ is good if $\phi_j \in [\frac{1}{16\sqrt{m}}, \frac{1}{4}]$. Otherwise we say it is bad (which corresponds to the case where $\phi_j \in [0, \frac{1}{16\sqrt{m}}] \cup [\frac{3}{4}, 1]$). In our analysis of the procedure described in the previous section we will use the fact that starting with the completely mixed state $\frac{I}{2^n}$ is completely equivalent to starting with an equal probabilistic mixture of all of the eigenstates of H. We will imagine that in the step 2 of the procedure one selects an eigenstate $|\psi_p\rangle$ uniformly at random since the density matrix of the output will be the same.

Case 1: Register i does not satisfy inequality 6

On one iteration of steps 2,3, and 4, the probability p_b that you pick a bad state and then accept it is

$$\begin{aligned}
& \Pr\left(|\psi_p\rangle \text{ is bad and you accept }\right) \\
\doteq\ & p_b \\
\leq\ & Pr\left(\text{accept}|\ |\psi_p\rangle \text{ was bad }\right) \\
\leq\ & Pr\left(|\phi_p - \phi'| > \frac{1}{16\sqrt{m}} - \frac{1}{20m}\right) \\
\leq\ & Pr\left(|\phi_p - \phi'| > \frac{1}{20m}\right) \\
\leq\ & \delta \\
=\ & \frac{1}{m^3},
\end{aligned}$$

by equation 8. So the total probability of accepting a bad state at any point is

$$\Pr\left(\text{accept a bad state ever}\right) \leq \sum_{k=1}^{m^2} \delta = \frac{1}{m}. \quad (10)$$

So if register i does not satisfy inequality 6 then the state ρ_i which is the output of the above procedure will satisfy

$$\begin{aligned}
& \mathrm{Tr}[H^{(i)}\rho_i] \\
\geq\ & Tr\left[H^{(i)}\frac{1}{2^n}\right] - Pr\left(\text{accept a bad state ever}\right) \\
=\ & 0 - \frac{1}{m}.
\end{aligned}$$

Case 2: Register i satisfies inequality 6

The probability p_g that you pick a good state (in one iteration of steps 2, 3, and 4) and then accept it is at least

$$\begin{aligned}
& \Pr\left(|\psi_p\rangle \text{ is good and you accept}\right) \\
\doteq\ & p_g \\
\geq\ & Pr\left(\frac{1}{4} \geq \phi_p \geq \frac{1}{8\sqrt{m}} \text{ and you accept}\right) \\
=\ & Pr\left(\frac{1}{4} \geq \phi_p \geq \frac{1}{8\sqrt{m}}\right) Pr\left(\text{ accept }|\ \frac{1}{4} \geq \phi_p \geq \frac{1}{8\sqrt{m}}\right) \\
\geq\ & Pr\left(\frac{1}{4} \geq \phi_p \geq \frac{1}{8\sqrt{m}}\right)(1-\delta) \\
\geq\ & \frac{3}{8m-2}\left(1 - \frac{1}{m^3}\right) \\
\geq\ & \frac{1}{4m} \text{, for m sufficiently large.}
\end{aligned}$$

The total probability of outputting a good state is

$$\begin{aligned}
& Pr(\text{ output a good state}) && (11) \\
=\ & \sum_{k=1}^{m^2} p_g(1-p_g-p_b)^{k-1} \\
=\ & \frac{p_g}{p_g+p_b}\left(1-(1-p_g-p_b)^{m^2}\right) \\
\geq\ & \frac{p_g}{p_g+p_b}\left(1-(1-p_g)^{m^2}\right) \\
\geq\ & \frac{p_g}{p_g+\delta}\left(1-(1-p_g)^{m^2}\right). \\
\geq\ & \frac{p_g}{p_g+\delta}\left(1-e^{-p_g m^2}\right) && (12) \\
\geq\ & \frac{1}{1+\frac{4}{m^2}}\left(1-e^{-p_g m^2}\right) \text{ for m sufficiently large.} \\
=\ & 1 - O(\frac{1}{m^2})
\end{aligned}$$

So in the case that register i satisfies inequality 6 then the state ρ_i will satisfy

$$\begin{aligned}
& \mathrm{Tr}\left[H^{(i)}\rho_i\right] \\
\geq\ & \Pr\left(\text{output a good state}\right)\frac{1}{4\sqrt{m}} \\
& \quad -\left(1 - Pr\left(\text{output a good state}\right)\right) \\
=\ & \frac{1}{4\sqrt{m}} + O(\frac{1}{m^2}).
\end{aligned}$$

We have thus shown that equation 2 holds for all indices i which satisfy inequality 6 and that equation 3 holds for the rest of the indices. As discussed in section 2.2, this guarantees (assuming at least $1/4$ of the indices i satisfy inequality 6) that our state $\rho = \rho_1 \otimes \rho_2 \otimes ... \otimes \rho_l$ is accepted by Aaronson's verifier with high probability, if $\epsilon \leq \frac{1}{16\sqrt{m}}$.

C Rapid mixing of the Markov chain over CSP solutions

C.1 Random functions are well-balanced

Our algorithm for verifying quantum money assumes that the hash functions are well-balanced in the sense that the number of satisfying assignments subject to certain types of constants is within a factor of two of the expected number. Letting $W(v) = \{w | (v,w) \in S\}$, we demand that four properties hold:

1. For any v, we expect half of the 2^r values of each row of w to satisfy the CSP, for a total of $2^{m(r-1)}$, so we demand that
$$\forall i, v. |W(v)| \leq 2 \cdot 2^{m(r-1)}. \tag{13}$$
2. For any pair of values of v differing in exactly one bit, we expect $2^{m(r-1)}$ values of w to satisfy the constraints for one of the values of v. Each of the 10 constraints on the bit that differs will exclude around half of those values of w when v is changed, so we expect $2^{m(r-1)-10}$ values of w to satisfy the constraints for both values of v. We demand that
$$\begin{aligned}&\forall v, v' \text{ with Hamming distance } 1.\\ &|W(v) \cap W(v')| \geq 2^{m(r-1)-11}.\end{aligned} \tag{14}$$
3. We refer to the number of assignments to the jth row of w that satisfy the constraints for some value of v and y_j as $a_{v,y_j,j}$. That is,
$$a_{v,y_j,j} = |\{(w_{j,1}, \ldots, w_{j,r}) | h_j(\cdots) = y_j\}|. \tag{15}$$
We expect half of the 2^r possible assignments to satisfy the constraints, so we demand that
$$\forall v, j, y_j. a_{v,y_j,j} \geq 2^{r-2}. \tag{16}$$
4. We expect a 2^{-m} fraction of assignments (v,w) to satisfy the CSP. We demand that
$$|S| \geq 2^{mr+n-m-1}. \tag{17}$$

These properties hold with high probability in the choice of random functions $\{h_i\}$ for $2^r \geq 2\log_2(m+r)$. We prove them using union bounds.

For properties (13) and (16), we can write $a_{v,y_j,j}$ as a sum of random variables indicating whether any particular value of the jth row of w satisfies the constraints:
$$a_{v,y_j,j} = \sum_{w_{j,1},\ldots,w_{j,r}} \mathbb{1}_{h_j(\cdots)=y_j}.$$

The total number $|W(v)|$ of assignments to all rows of w that satisfy the constraints is thus $\prod_j a_{v,y_j,j}$. Under uniform selection of h_j and for any y_j, each $a_{v,y_j,j}$ is i.i.d. $\mathrm{Binom}(2^r, \frac{1}{2})$. By a Chernoff bound, with probability at least $1 - \exp\left(-\frac{1}{4}\left(1+\frac{\log 2}{m}\right)^2 2^r\right)$,
$$a_{v,y_j,j} \leq \left(1 + \frac{\log 2}{m}\right) 2^{r-1}. \tag{18}$$

Taking a union bound, (18) holds for all v and j w.p. at least $1 - m2^n \exp\left(-\frac{1}{4}\left(1+\frac{\log 2}{m}\right)^2 2^r\right)$, giving property (16). The same bound gives $\prod_j \sigma_{v,j} \leq 2 \cdot 2^{m(r-1)}$, which proves property (13).

The proof of properties (17) and (14) follow similar arguments. The latter picks up a factor of 2^{-10} because each element of $W(v)$ is in $W(v')$ with probability 2^{-10}—it must satisfy one additional constraint for each clause covering the bit in which v and v' differ.

Because a collection of random functions is well-balanced with high probability, any good collection of cryptographic hashes will also be well-balanced.

C.2 For any well-balanced set h_j, the Markov chain mixes rapidly

We refer to the (directed, symmetric) graph on which this Markov chain walks as $\tilde{G}$, and we refer to the non-self-loop edges resulting from the first rule as *inner* edges (because they flip inner bits) and to the non-self-loop edges resulting from the second rule as *outer* edges. Our Markov chain is symmetric; hence the uniform distribution over all CSP solutions is a stationary solution.

We use the method of canonical paths to prove that our Markov chain mixes rapidly. For each pair of states $\left(\left(v^{(\text{start})}, w^{(\text{start})}\right), \left(v^{(\text{end})}, w^{(\text{end})}\right)\right)$, we define a canonical path on the graph $\tilde{G}$. Each such path traverses a number of inner edges exactly equal to the Hamming distance between $v^{(\text{start})}$ and $v^{(\text{end})}$. The first inner edge sets the first bit in which the $v^{(\text{start})}$ and $v^{(\text{end})}$ differ, the second inner edge sets the second bit, and so on. For each pair (v, v') that differ in exactly one bit, a total of at most $2^{2m(r-1)+n}$ canonical paths traverses any inner edge of the form $(v, w) \to (v', w)$—there are 2^{n-2} pairs $\left(v^{(\text{start})}, v^{(\text{end})}\right)$ between which canonical paths traverse an edge changing the inner bits from v to v', and, by (13), each pair $\left(v^{(\text{start})}, v^{(\text{end})}\right)$ corresponds to at most $\left(2 \cdot 2^{m(r-1)}\right)^2$ paths. For each inner edge from v to v' on a canonical path, we are free to choose any $w \in W(v) \cap W(v')$, of which, by (14), there are at least $2^{m(r-1)-11}$. We evenly distribute these inner edges among such w, giving at most

$$2^{m(r-1)+n+11} + 1 \tag{19}$$

canonical paths traversing each inner edge.

We now address how we route the portions of the canonical paths that use outer edges. We refer to a maximal subpath of a canonical path that uses only outer edges as an *outer subpath* (so each canonical path contains at most $n + 1$ outer subpaths). The subgraphs O_v induced by all vertices with a given v contain all outer edges in $\tilde{G}$ of the form $(v, w) \to (v, w')$ where w and w' differ in exactly one row, and each outer subpath traverses only edges in O_v for one value of v. Counting outer subpaths that start at a state (v, w), there are, by (13) at most $2^{m(r-1)+n+1}$ canonical paths that begin at that state and at most $n \cdot 2^{m(r-1)+n+11} + n$ that arrive by an inner edge, so at most $\left(n + 2^{-10}\right) 2^{m(r-1)+n+11} + n + n \leq (n+1)\, 2^{m(r-1)+n+11}$ outer subpaths begin at any state. By the same counting at most $(n+1)\, 2^{m(r-1)+n+11}$ outer subpaths end at any state.

We define an outer subgraph O_v as a subgraph induced by all states sharing a given value of v. We route outer subpaths on each outer subgraph by choosing a good outcome of a randomized algorithm. First, add dummy outer subpaths such that *exactly* $(n+1)\, 2^{m(r-1)+n+11}$ subpaths begin and the same number end at each state. Allowing m self-loops per state (one for each row of w) that we will remove later, each outer subpath from $\left(v, w^{(\text{init})}\right)$ to $\left(v, w^{(\text{final})}\right)$ traverses exactly $2m$ outer edges: for the first pass, sequentially set the first through last rows of w to random values that satisfy the CSP, and then, for the second pass, sequentially set the first through last rows to their desired values. We wish to compute the expected number μ_{v,w_1,w_2} of outer subpaths containing the edge $(v, w_1) \to (v, w_2)$. By linearity of expectation, μ_{v,w_1,w_2} is the total number of outer subpaths (including dummies) times the expected number of times that an outer subpath, selected uniformly at random, traverses any given edge. But *every* outer subpath traverses exactly two edges affecting a given but of w, and, on both passes, the marginal distribution of which such edge that path takes is uniform. So a $2a^{-1}_{v,y_j,j}$ fraction of outer subpaths traverses any given edge, so $\mu_{v,w_1,w_2} = 2a^{-1}_{v,y_j,j}\,(n+1)\, 2^{m(r-1)+n+11}$. By (16), $a_{v,y_j,j} \geq 2^{r-2}$, giving

$$\mu_{v,w_1,w_2} \leq= (n+1)\, 2^{mr-m-r+n+14}.$$

On the other hand, the number of outer subpaths (after pruning dummy subpaths) containing any given edge is the sum of independent random binary variables, each of which indicates whether a particular outer subpath contains that edge. This allows us to apply a Chernoff bound: for any outer edge and for any $x > 2e \cdot (n+1)\, 2^{mr-m-r+n+14}$, $\Pr\left[\text{paths containing that edge} \geq x\right] \leq 2^{-x}$ (see, for example, ex. 4.1 in [10]). Choosing, for concreteness, $x = 6 \cdot (n+1)\, 2^{mr-m-r+n+14}$, and applying a union bound over all edges in O_v, there is a nonzero probability that none of the edges are on more than

$$6 \cdot (n+1)\, 2^{mr-m-r+n+14} \tag{20}$$

canonical paths. This implies that at least one routing of the outer subpaths satisfies has this property, so we can derandomize our algorithm by choosing any such routing.

All that remains is to bound the spectral gap of our Markov chain. For consistency with the literature, we let $\pi(x)$ be the stationary probability on a state x. In our case, $\pi(x) = \left|\tilde{G}\right|^{-1}$ for all x. Since all of our canonical paths can be routed as above, our Markov chain is irreducible. Inner edges have weight $P(e) = \frac{1}{4n}$ and outer edges have weight

$P(e) = \frac{1}{4m2^r}$. A canonical path γ_{xy} from x to y traverses at most n inner edges and $2(n+1)m$ outer edges, so length $|\gamma_{xy}|$ of any path is bounded above by $n + 2(n+1)m$. We define a measure of congestion K:

$$\begin{aligned} K &= \max_e P(e)^{-1} \left|\tilde{G}\right| \sum_{\gamma_{xy} \ni e} |\gamma_{xy}| \pi(x) \pi(y) \\ &= \left|\tilde{G}\right|^{-1} \max_e P(e)^{-1} \sum_{\gamma_{xy} \ni e} |\gamma_{xy}| \\ &\leq \left|\tilde{G}\right|^{-1} (n + 2(n+1)m) \max_e g(e). \end{aligned}$$

where $g(e) = P(e)^{-1} |\{\gamma_{xy} | e \in \gamma_{xy}\}|$. By (19), inner edges have $g(e) \leq 2^{13}(n+1)2^{mr+n-m}$ and by (20), outer edges have $g(e) \leq 3 \cdot 2^{17} m (n+1) 2^{mr+n-m}$. By (17), we know that $\left|\tilde{G}\right| \geq 2^{mr+n-m-1}$. The bound on inner edges is strictly stronger than on outer edges, so we have

$$K \leq 3 \cdot 2^{18} (n + 2mn + 2) m (n+1).$$

By a standard theorem about congestion of canonical paths [11], the spectral gap of our Markov chain is at least $\frac{1}{K} = \left[3 \cdot 2^{18} (n + 2mn + 2) m (n+1)\right]^{-1}$.