On the query complexity of counterfeiting quantum money

Andrew Lutomirski

December 14, 2010

Abstract

Quantum money is a quantum cryptographic protocol in which a mint can produce a state (called a quantum bill) which anyone can verify but no one can copy. The only published protocol (quantum money from knots) implements a protocol for collision-free money that is defined in terms of a classical oracle. I give a reduction from a hard classical problem to a restricted type of attack against the general collision-free money protocol.

1 Introduction

Quantum money is a quantum cryptographic protocol in which a mint can produce a state (called a quantum bill) which anyone can verify but no one can copy. A quantum money protocol is called collision-free if each quantum bill has an associated serial number such that no one, not even the mint, can efficiently produce two quantum bills with the same serial number. Collision-free money can be used as quantum bills: the mint produces a large number of collision-free money states and publishes a list of all the serial numbers. One benefit of collision-free money is that no secret is needed: the only unique power that the mint has is the ability to publish the list of valid serial numbers.

In [2], Lutomirski et al. give a construction for collision-free money using classical oracles. The computational basis states correspond to a large set S. The first oracle is a *labeling function* L that assigns a label to each element of S. The second oracle is a classical function M that, given an element of S, finds another random element with the same label.

The labels correspond to serial numbers of the money states. For a serial number ℓ, the money state is

$$|\$_\ell\rangle = \sum_{\substack{s \in S \\ L(s) = \ell}} |s\rangle.$$

The state $|\$_\ell\rangle$ can be made for a *random* ℓ by making the uniform superposition over all elements of S and measuring L. The verification algorithm measures L

and then uses M to verify that it was given the correct superposition. (Details can be found in [2] or [1].)

For security, the labeling function should be far from one-to-one: if a uniformly random $s \in S$ is chosen and its label $\ell = L(s)$ is computed, no value of ℓ should occur with more than negligible probability, but there should be exponentially many other elements of S with the same label. This means that the probability of generating $|\$_\ell\rangle$ twice for the same ℓ is negligible and that the money states are large superpositions and thus hard to prepare directly.

The only published quantum money protocol which is not yet broken is an implementation of collision-free money using knot diagrams for S, Reidemeister moves for M, and the Alexander polynomial for L. [1]

There is, so far, no security proof for this type of money. Ideally, preparing any $+1$ eigenstate of the projector

$$\sum_\ell |\$_\ell\rangle\langle\$_\ell| \otimes |\$_\ell\rangle\langle\$_\ell|$$

(that is, any pair of money states with the same serial number) would require exponentially many queries to L and M. In this paper, I give progress toward a weaker result: that given $s \in S$, it is hard to prepare $|\$_{L(s)}\rangle$. This would imply that any attempt to counterfeit collision-free money by first measuring a valid money state in the computational basis would fail.

We give a reduction from the SAME COMPONENT PROBLEM (determining whether two elements of S have the same label *without access to* L) to the simplified counterfeiting problem.

2 Formalism

Definition 1. A PARTIAL MIXING ORACLE M on a partition $\{S_1, \ldots, S_c\}$ of a set S is a function mapping $\mathbb{Z}_r \times S \to S$ for some $r = \Omega\left(|S|^2\right)$. For notational convenience, we will write $M(i, s)$ as $M_i(s)$. Each M_i is invertible, and, in general, we assume that any algorithm given access to M is also given access to each M_i^{-1}. M has the property that it mixes over the components $S_1, \ldots S_k$: each M_i maps each component S_j to itself but otherwise behaves like a random permutation on each S_j.

Definition 2. The SIMPLE COUNTERFEITING problem is: Given an integer n, a set S, a partial mixing oracle M on a partition $\{S_1, \ldots, S_{2^n}\}$ of S (some elements of which may be empty), a labeling oracle $L : S \to 2^n$ such that $L(x) = i$ if $x \in S_i$, and an element $s \in S_i$ for some i, prepare the state $|\$_i\rangle = \sum_{x \in S_i} |x\rangle$.

Definition 3. The SAME COMPONENT problem is: Given a set T and a partial mixing oracle M on an unknown partition $\{T_1, \ldots, T_c\}$ of T, test whether two elements s, t lie in the same component.

Graph isomorphism and group membership are special cases of the SAME COMPONENT problem. Graph connectivity would also be a special case if the

mixing properties of M were relaxed. I conjecture that there is no worst-case polynomial time quantum algorithm for SAME COMPONENT.

The SAME COMPONENT problem has a related state preparation problem.

Definition 4. The COMPONENT SUPERPOSITION problem is: Given an integer n, a set T with $|T| \leq 2^n$, a partial mixing oracle M on an unknown partition $\{T_1, \ldots, T_c\}$ of T, and an element $t \in T$, prepare the state $\sum_{x \in T_j} |x\rangle$ where T_j is the component containing t.

The SAME COMPONENT problem reduces to COMPONENT SUPERPOSITION by a swap test. I give a reduction from COMPONENT SUPERPOSITION to SIMPLE COUNTERFEITING relative to random permutation oracles.

3 The reduction

We are given an instance of the COMPONENT SUPERPOSITION problem on the set T and partial mixing oracle M. WLOG, assume that our starting element t is in T_1. We assume that T can be easily encoded in binary—without any further loss of generality, we take $T \subseteq \mathbb{Z}_{2^n}$. We further assume that we have access to random permutations.

We take as given a quantum algorithm A that solves the SIMPLE COUNTERFEITING problem on the set $S = \mathbb{Z}_{2^{2n}}$ using $O(\text{poly}(n))$ queries to the oracles. A takes as input the number n, a starting state s, and two oracles. As output, it either reports "failed" (and produces no output) or reports "success" and outputs the state $|\$_i\rangle$ (or something exponentially close to $|\$_i\rangle$), where i is the label of s.

Let $\bar{T} = \mathbb{Z}_{2^n} \setminus T$. Then

$$P = \left\{T_1, \ldots, T_c, \bar{T}, \{2^n + 1\}, \ldots, \left\{2^{2n}\right\}\right\}$$

is a partition of S.

Let R be any partial mixing oracle on P that is consistent with M (in the sense that $R_i(x) = M_i(x)$ whenever $x \in T$). A query to R can be implemented with at most one query of M (when the input is in T) and at most one query of a random permutation (when the input is in $\bar{T}$).

Let $L : S \to \mathbb{Z}_{2^{2n}}$ be the labeling function

$$L(x) = \begin{cases} 0 & \text{if } x \in T \\ 1 & \text{if } x \in \bar{T} \\ x & \text{if } x > 2^n \end{cases};$$

L is straightforward to implement with a single query to the binary encoding of T . (For simplicity, the domain of L is unnecessarily large.)

The function L is in general inconsistent with R, as it takes the same value on the entire set T. We address this with another partition P' and second set of

oracles R' and L'. Let P' be the partition of S that contains T_1 and partitions everything else into sets of size 1. Let

$$L'(x) = \begin{cases} 0 & \text{if } x \in T_1 \\ x & \text{otherwise} \end{cases}.$$

Finally, let R' be the partial mixing oracle that does the same thing as M on T_1 and leaves all other elements of S unchanged.

Note that the unprimed oracles L and R are inconsistent but can be efficiently implemented, whereas the primed oracles L' and R' are consistent but cannot be implemented without being able to test membership in T_1.

The primed and unprimed oracles are the same on the vast majority of their inputs. It is easy to tell them apart, though—just query both on a random element in T. If we randomly permute the oracles, however, intuitively it should be impossible to distinguish them and the SIMPLE COUNTERFEITING algorithm A should work just as well with the unprimed oracles as with the primed oracles. We can formalize this notion by embedding a Grover instance in the difference between the primed and unprimed oracles.

The outcome of A on any problem instance is a mixed state on two registers: one bit indicating success and a register containing the output state if the algorithm succeeds.

Claim. If we run A on instance $I = \left(n, \sigma(t), \pi \circ L \circ \sigma^{-1}, \sigma \circ R \circ \sigma^{-1}\right)$ or on $I' = \left(n, \sigma(t), \pi \circ L' \circ \sigma^{-1}, \sigma \circ R' \circ \sigma^{-1}\right)$, the mixed state outcomes ρ and ρ' have negligible trace distance $|\rho - \rho'|_{\text{tr}}$, with high probability in (π, σ).

Proof. Assume the contrary. Let $f : \mathbb{Z}_{2^n} \to \{0, 1\}$ be any unknown binary oracle of Hamming weight zero or one. That is, f is an instance of the Grover problem.

To solve the Grover problem using A, we will perform some preprocessing. Choose 2^n disjoint subsets $U_1, \ldots, U_{2^n} \subseteq \mathbb{Z}_{2^{2n}} \setminus T_1$, each of size $2^n - |T_1|$. For each i, we will choose an embedding of $T \setminus T_1$ into U_i. To do this, we choose a bijection τ_i from $\mathbb{Z}_{2^n} \setminus T_1$ to U_i. We define a partial mixing oracle R_i'' that acts on T_1 identically to M, on U_i as $\tau_i \circ M \circ \tau_i^{-1}$, and as the identity everywhere else. We define a matching label function L_i'' that evaluates to 0 on T_1 and $\tau_i(T \setminus T_1)$, to 1 on $\tau_i\left(\bar{T}\right)$, and maps every other $x \in \mathbb{Z}_{2^{2n}}$ to itself.

Now we create a composite labeling function

$$L''(x) = \begin{cases} L_i''(x) & \text{if } x \in U_i \text{ and } f(i) = 1 \text{ for some } i \\ L'(x) & \text{otherwise} \end{cases}$$

and the mixing function

$$R''(j, x) = \begin{cases} R_i''(j, x) & \text{if } x \in U_i \text{ and } f(i) = 1 \text{ for some } i \\ R'(x) & \text{otherwise} \end{cases}.$$

Note that, if f is all zeros, then L'' and R'' look like L' and R' up to a permutation and, if f has a single nonzero entry then L'' and R'' look like L and R

up to a permutation. Furthermore, L'' and R'' can each be implemented with a single query to f (although they may require an exponentially large number of queries to the original oracle M).

Generate random permutations π'' and σ'' and run A on

$$I'' = \left(n, \sigma''(t), \pi'' \circ L'' \circ \sigma''^{-1}, \sigma'' \circ R'' \circ \sigma''^{-1}\right).$$

Marginalized over $(\pi, \pi'', \sigma, \sigma'')$, I'' is distributed identically to either I and I', depending on the Hamming weight of f. Since, with nonnegligible probability in (π, σ), A produces output states that are non-negligibly different in trace distance in the two cases, the output of A can be used to decide the Hamming weight of f in a polynomial number of tries. This is a violation of the BBBV theorem.

Therefore the output of A on I and I' is, with high probability in (π, σ), negligibly different in trace distance. □

This means that, averaged over (π, σ), A will succeed on input I with probability at least $\frac{1}{\text{poly}(n)}$ *even though* I *is not an instance of* SIMPLE COUNTERFEITING. We can condition on success, in which case the output state must be close to $\sum_{x \in T_1} |\sigma(x)\rangle$. By applying σ^{-1}, we recover $\sum_{x \in T_1} |x\rangle$, which is what we wanted.

4 Conclusions

I have shown that, in the worst case, a simplified approach to counterfeiting collision-free quantum money is at least as hard as the SAME COMPONENT PROBLEM, an unstructured search problem that contains both graph isomorphism and group membership as special cases. Graph isomorphism may or may not be hard on average, but group membership on black box groups is at least hard for classical computers [3]. I suspect that the SAME COMPONENT PROBLEM is outside of BQP in a black-box setting, but I have not yet been able to prove this.

The more general problem of creating quantum money collisions without solving SIMPLE COUNTERFEITING remains open.

References

[1] Edward Farhi, David Gosset, Avinatan Hassidim, Andrew Lutomirski, and Peter Shor. Quantum money from knots. 2010.

[2] 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. In *Innovations in Computer Science*, 2010.

[3] J. Watrous. Succinct quantum proofs for properties of finite groups. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:537, 2000.