Guide to Implementing a Quantum Coin Scheme

Based on work by Jerry Hogan, Hazel Murray and Joao F. Santos
Supervised by Harun Siljak

Hazel Murray

November 2019

Abstract

In this article we describe Gavinsky's scheme [3] for creating quantum coins. We contribute by setting out the methods for implementing this scheme computationally. We convert the algebraic steps into computing steps for both the quantum creation of the coins and the quantum verification.

Goal Create quantum coins which can be traded and their authenticity verified.

Definition 1 (Quantum Coin)**.** A quantum coin is a unique object that can be created by a trusted mint[a] and then circulated among untrusted holders.

[a] In our notation the mint is both the creator and the verifier of the coin. In Gavinsky's work the term bank is used for the two roles.

1 Steps in Gavinsky's protocol

Coins with unique identification numbers have been created in the mint. A holder Bob has one such coin and wishes to verify its authenticity.

1. The holder sends the identification number on the coin they hold to the mint.
2. The mint chooses uniformly at random a set $\mathcal{L}_{mt} \subset [k]$ of size t, and sends it to the coin holder.
3. The holder consults with P and chooses uniformly at random a set $\mathcal{L}_{hl} \subset \mathcal{L}_{mt}$ consisting of $2t/3$ yet unmarked positions. He sends $\mathcal{L}_{hl}$ to the mint and marks in P all the elements of $\mathcal{L}_{hl}$ as used.
4. The mint chooses at random $2t/3$ values $m_i \in \{0,1\}$, one for each $i \in \mathcal{L}_{hl}$, and sends them to the coin holder.
5. The holder measures the quantum registers corresponding to the elements of $\mathcal{L}_{hl}$ in order to produce $2t/3$ pairs $(a_i, b_i) \in HMP_4$ for all $i \in \mathcal{L}_{hl}$. The holder sends the list of (a_i, b_i)'s to the mint.
6. The mint checks whether $(x_i, m_i, a_i, b_i) \in HMP_4$ for all $i \in \mathcal{L}_{hl}$, in which case it confirms validity of the coin. Otherwise the coin is declared to be a counterfeit.

We will expand on the steps in the above protocol. We will provide the algebraic steps given in Gavinsky's paper and our translation of these into computing protocol.

2 Creation of $\mathcal{Q}$ coins

First let us describe Gavinsky's definition of a coin, named $\mathcal{Q}$-coin.

Definition 2 ($\mathcal{Q}$-coins)**.** Let $3|t$. A secret record consists of k entries $x_1, \ldots, x_k$, $x_i \in \{0,1\}^4$ (i.e., the secret record contains $4k$ classical bits).

A "fresh" $\mathcal{Q}$-coin corresponding to the record $(x_1, \ldots, x_k)$ consists of

- k quantum registers consisting of 2 qubits each, where the i'th register contains $|\alpha(x_i)\rangle$;
- a k-bit classical register P, that is initially set to 0^k;
- a unique identification number.

A mint produces *fresh* $\mathcal{Q}$-coins; as a $\mathcal{Q}$-coin goes through more and more verification protocols, its quantum registers lose their original content. **Reason:** when a quantum wavelength is measured it collapses. For each quantum verification we measure $2t/3$ quantum registers. So we collapse $2t/3$ registers every time we verify the coin's identity. Depending on the level of trust we require and how long we want the coin to last we can choose the value t.

2.1 Implementation

Creation of the coins requires the conversion of the 4 classical bits to two quantum bits.

Definition 3 (*HMP$_4$-states*)**.** Let $x \in \{0,1\}^4$. The corresponding *HMP$_4$-state* is

$$|\alpha(x)\rangle \stackrel{\text{def}}{=} \frac{1}{\sqrt{4}} \sum_{1 \le i \le 4} (-1)^{x_i} |i\rangle.$$

Algebraically this formula perfectly represents the conversion of 4 bits to 2 qubits. For example, the string of 4 classical bits $[0110]^T$ is converted to qubits as

$$\begin{aligned} |\alpha([0110]^T)\rangle &= \frac{1}{\sqrt{4}}((-1)^0|1\rangle + (-1)^1|2\rangle + (-1)^1|3\rangle + (-1)^0|4\rangle) \\ &= \frac{1}{2}(|1\rangle - |2\rangle - |3\rangle + |4\rangle) \end{aligned}$$

We wish to create a pair of quantum bits corresponding to each of the 16 possible arrangements of the 4 classical bits. To effect the same conversion computationally we need to manipulate a starting pair of qubits.
No entanglement

- Given any two quantum bits. Let $q_1 = \alpha|0\rangle + \beta|1\rangle$ and $q_2 = \gamma|0\rangle + \delta|1\rangle$.
- The state space[1] of these two bits is

$$\begin{aligned} q_1 \otimes q_2 = (\alpha|0\rangle + \beta|1\rangle)(\gamma|0\rangle + \delta|1\rangle) &= \alpha\gamma|00\rangle + \alpha\delta|01\rangle + \beta\gamma|10\rangle + \beta\delta|11\rangle & (1) \\ &= \alpha\gamma|1\rangle + \alpha\delta|2\rangle + \beta\gamma|3\rangle + \beta\gamma|4\rangle & (2) \\ &\equiv \begin{bmatrix} \alpha\gamma & \alpha\delta & \beta\gamma & \beta\gamma \end{bmatrix}^T & (3) \end{aligned}$$

- α, β, γ and δ can each take either -1 or +1. Thus we we can create:

$$\begin{array}{lll} \alpha = -1 & \beta,\gamma,\delta = 1 & \rightarrow \begin{bmatrix} -1 & -1 & 1 & 1 \end{bmatrix}^T = |Q_1\rangle \\ \beta = -1 & \alpha,\gamma,\delta = 1 & \rightarrow \begin{bmatrix} 1 & 1 & -1 & -1 \end{bmatrix}^T = |Q_2\rangle \\ \gamma = -1 & \alpha,\beta,\delta = 1 & \rightarrow \begin{bmatrix} -1 & 1 & -1 & 1 \end{bmatrix}^T = |Q_3\rangle \\ \delta = -1 & \alpha,\beta,\gamma = 1 & \rightarrow \begin{bmatrix} 1 & -1 & 1 & -1 \end{bmatrix}^T = |Q_4\rangle \\ \alpha,\beta = -1 & \gamma,\delta = 1 & \rightarrow \begin{bmatrix} -1 & -1 & -1 & -1 \end{bmatrix}^T = |Q_5\rangle \\ \alpha,\gamma = -1 & \beta,\delta = 1 & \rightarrow \begin{bmatrix} 1 & -1 & -1 & 1 \end{bmatrix}^T = |Q_6\rangle \\ \alpha,\delta = -1 & \beta,\gamma = 1 & \rightarrow \begin{bmatrix} -1 & 1 & 1 & -1 \end{bmatrix}^T = |Q_7\rangle \\ \alpha,\beta,\gamma,\delta = -1 & & \rightarrow \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^T = |Q_8\rangle \end{array}$$

- All other combinations give repetitions of these 8 quantum states.

With entanglement

- First let us create entangled qubits.

Definition 4 (Bell States)**.** Bell states are four specific maximally entangled quantum states of two qubits. They are in a superposition of 0 and 1. They form a maximally entangled basis, known as the Bell basis, of the four-dimensional Hilbert space for two qubits:

$$\begin{aligned} \left|\Phi^+\right\rangle &= \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |0\rangle_B + |1\rangle_A \otimes |1\rangle_B) = \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix}^T & (4) \\ \left|\Phi^-\right\rangle &= \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |0\rangle_B - |1\rangle_A \otimes |1\rangle_B) = \begin{bmatrix} 1 & 0 & 0 & -1 \end{bmatrix}^T & (5) \\ \left|\Psi^+\right\rangle &= \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |1\rangle_B + |1\rangle_A \otimes |0\rangle_B) = \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix}^T & (6) \\ \left|\Psi^-\right\rangle &= \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |1\rangle_B - |1\rangle_A \otimes |0\rangle_B) = \begin{bmatrix} 1 & 0 & -1 & 0 \end{bmatrix}^T & (7) \end{aligned}$$

- These Bell states can each be created by taking a computational basis as the input and putting this through a Hadamard gate and a CNOT gate. As an example, Figure 1 shows a quantum circuit with a Hadamard-CNOT combination. Given input $|00\rangle$ it produces the Bell state $\left|\Phi^+\right\rangle$.
- Using the now entangled qubits we can generate the required remaining 8 combinations by putting these through quantum gates. Below we define these gates.

[1] The state space of a composite systems is the tensor product of the state spaces of the component physical systems.

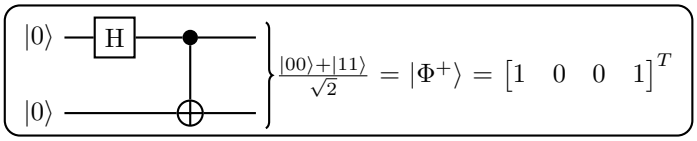

Figure 1: Quantum circuit showing a Hadamard-CNOT combination. Given input $|00\rangle$ it produces the Bell state $|\Phi^+\rangle$

.

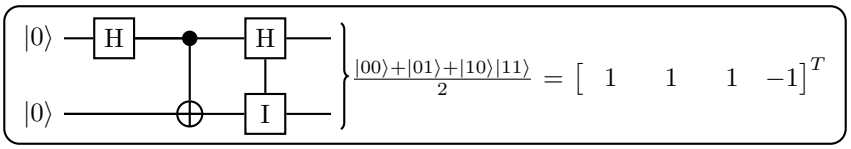

Figure 2: The extended Hadamard gate applied to the entangled Bell state $|\Phi^+\rangle$.

Definition 5 (Hadamard gate)**.** The Hadamard gate acts on a single qubit. It maps the basis state $|0\rangle$ to $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|1\rangle$ to $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$. That is, it creates a superposition where the measurement has equal probability of becoming 0 or 1. It is represented by the Hadamard matrix:

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$$

Definition 6 (Pauli-X gate)**.** The Pauli-X gate acts on a single qubit. It is the quantum equivalent of the NOT gate for classical computers. Due to this nature, it is sometimes called bit-flip. It maps $|0\rangle$ to $|1\rangle$ and visa versa. It is represented by the Pauli X matrix:

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

Definition 7 (Pauli-Z gate)**.** The Pauli-Z gate acts on a single qubit. It equates to a rotation around the Z-axis of the Bloch sphere by π radians. It leaves the basis state $|0\rangle$ unchanged and maps $|1\rangle$ to -$|1\rangle$. It is sometimes called phase-flip. It is represented by the Pauli Z matrix:

$$Z = \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$$

Definition 8. The swap gate swaps two qubits. It is represented by the matrix:

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

An example of these gates acting on the entangles qubits is shown in Figure 2. It shows the entangled Bell state $|\Phi^+\rangle$ passed through an extended Hadamard gate.

- Below we depict the combination of gates used to create the 8 linear combinations. I denotes the identity matrix, $\otimes$ denotes the tensor product (gates wired in parallel) and $\times$ denotes the ordinary matrix cross product (serially wired gates).

$$\begin{aligned}
(H\otimes I)\times|\Phi^+\rangle &\rightarrow \tfrac{1}{2}\begin{bmatrix}1 & 1 & 1 & -1\end{bmatrix}^T = |Q_9\rangle\\
(H\otimes I)\times|\Psi^+\rangle &\rightarrow \tfrac{1}{2}\begin{bmatrix}1 & 1 & -1 & 1\end{bmatrix}^T = |Q_{10}\rangle\\
(H\otimes I)\times|\Phi^-\rangle &\rightarrow \tfrac{1}{2}\begin{bmatrix}1 & -1 & 1 & 1\end{bmatrix}^T = |Q_{11}\rangle\\
(X\otimes I)\times|Q_{10}\rangle &\rightarrow \tfrac{1}{2}\begin{bmatrix}-1 & 1 & 1 & 1\end{bmatrix}^T = |Q_{12}\rangle\\
(Z\otimes I)\times|Q_{11}\rangle &\rightarrow \tfrac{1}{2}\begin{bmatrix}1 & -1 & -1 & -1\end{bmatrix}^T = |Q_{13}\rangle\\
(Z\otimes I)\times|Q_{12}\rangle &\rightarrow \tfrac{1}{2}\begin{bmatrix}-1 & 1 & -1 & -1\end{bmatrix}^T = |Q_{14}\rangle\\
(I\otimes Z)\times|Q_{12}\rangle &\rightarrow \tfrac{1}{2}\begin{bmatrix}-1 & -1 & 1 & -1\end{bmatrix}^T = |Q_{15}\rangle\\
(X\otimes I)\times|Q_{14}\rangle &\rightarrow \tfrac{1}{2}\begin{bmatrix}-1 & -1 & -1 & 1\end{bmatrix}^T = |Q_{16}\rangle
\end{aligned}$$

- For the creation of quantum coin, this mapping of classical bits to quantum bits can be hard-coded into the system. This this work only need be completed once.
- As described in [2] there are alternative circuit configurations which will produce equivalent results.

3 Verification

3.1 Steps 1 – 4

1. The holder sends the identification number on the coin they hold to the mint.
2. The mint chooses uniformly at random a set $\mathcal{L}_{mt} \subset [k]$ of size t, and sends it to the coin holder.
3. The holder consults with P and chooses uniformly at random a set $\mathcal{L}_{hl} \subset \mathcal{L}_{mt}$ consisting of $2t/3$ yet unmarked positions. He sends $\mathcal{L}_{hl}$ to the mint and marks in P all the elements of $\mathcal{L}_{hl}$ as used.
4. The mint chooses at random $2t/3$ values $m_i \in \{0, 1\}$, one for each $i \in \mathcal{L}_{hl}$, and sends them to the coin holder.

These are classical steps. They involve classical correspondence between Bob and the Bank. Below we include the code for both parties. Additional code can be found on github in files `coin.py` and `bob2.py` [4].

Listing 1: Code for Mint - Alice

```
def verify_coin(register):
    #Step 2
    register_c = list(register)
    t=3*(randint(1,math.floor(k/3)))
    m_s = []
    list_of_random_indexes = random.sample(register_c, t)
    with CQCConnection("Alice") as Mint:
        # Step 2 cont; send the list or indexes to the coin holder
        Mint.sendClassical("Bob", list_of_random_indexes)
        # Wait for receiver to send back subset of list
        sleep(1)
        #Receive output from step 3
        index_list = Mint.recvClassical()
        rlist = list(index_list)
        # Step 4; send randomly either 0 or 1 to correspond to each index in Bob's list.
        for c,i in enumerate(rlist):
            register_c.remove(i)
            m_s.append(random.randint(0,1))
        Mint.sendClassical("Bob", m_s)
```

Listing 2: Code for Bob - coin holder

```
def verify_coin():
    print("Verify Coin ID 1")
    with CQCConnection("Bob") as Bob:
        register_c = list(range(8))
        #Receive output from step 2
        list_of_random_indexes = Bob.recvClassical()
        #Step 3; chooses a subset of the index list received of size 2t/3
        t = len(list_of_random_indexes)
        local_selection = random.sample(list_of_random_indexes, 2*t/3)
        #Step 3 cont; send subset to Mint and mark those indexes as used.
        Bob.sendClassical("Alice", local_selection)
        for c,i in enumerate(local_selection):
            register_c.remove(i)
        #Receive output from step 4
        m_s = Bob.recvClassical()
```

3.2 Steps 5 & 6

5. The holder measures the quantum registers corresponding to the elements of $\mathcal{L}_{hl}$ in order to produce $2t/3$ pairs $(a_i, b_i) \in HMP_4$ for all $i \in \mathcal{L}_{hl}$. The holder sends the list of (a_i, b_i)'s to the mint.
6. The mint checks whether $(x_i, m_i, a_i, b_i) \in HMP_4$ for all $i \in \mathcal{L}_{hl}$, in which case it confirms validity of the coin. Otherwise the coin is declared to be a counterfeit.

Definition 9 (*HMP_4-queries*)**.** An *HMP_4-query* is an element $m \in \{0,1\}$. A valid answer to the query w.r.t $x \in \{0,1\}^4$ is a pair $(a,b) \in \{0,1\} \times \{0,1\}$, such that $(x, m, a, b) \in HMP_4$.

An *HMP_4-state* can be used to answer an *HMP_4-query* with certainty: If $m = 0$, let

$$v_1 \stackrel{\text{def}}{=} \frac{|1\rangle + |2\rangle}{\sqrt{2}}, v_2 \stackrel{\text{def}}{=} \frac{|1\rangle - |2\rangle}{\sqrt{2}}, v_3 \stackrel{\text{def}}{=} \frac{|3\rangle + |4\rangle}{\sqrt{2}}, v_4 \stackrel{\text{def}}{=} \frac{|3\rangle - |4\rangle}{\sqrt{2}}; \tag{8}$$

otherwise $(m = 1)$, let

$$v_1 \stackrel{\text{def}}{=} \frac{|1\rangle + |3\rangle}{\sqrt{2}}, v_2 \stackrel{\text{def}}{=} \frac{|1\rangle - |3\rangle}{\sqrt{2}}, v_3 \stackrel{\text{def}}{=} \frac{|2\rangle + |4\rangle}{\sqrt{2}}, v_4 \stackrel{\text{def}}{=} \frac{|2\rangle - |4\rangle}{\sqrt{2}}; \tag{9}$$

Measure $|\alpha(x)\rangle$ in the basis $\{v_1, v_2, v_3, v_4\}$ and let (a,b) be $(0,0)$ if the outcome is v_1; $(0,1)$ in the case of v_2; $(1,0)$ in the case of v_3; $(1,1)$ in the case of v_4. Then $(x, m, a, b) \in HMP_4$ always.

3.3 Step 5

Step 5 involves using the m received from the mint as the *HMP_4-query* and generating an output (a,b) determined by the outcome resulting from measuring with the respective basis. The basis describes the gates the qubits are passed through.

If $m = 0$ we measure the qubits using a basis determined by an expanded Hadamard gate.

$$I \otimes H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

If $m = 1$ we measure the qubits using a basis determined by the expanded Hadamard gate and the SWAP gate.

$$SWAP \times (I \otimes H) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

3.4 Step 6

In the final step the mint receives a value (a,b) corresponding to each index in $\mathcal{L}_{hl}$. The mint knows the classical value $x \in \{0,1\}$ corresponding to each index position and the given $m \in \{0,1\}$. So the bank can verify whether $(x, m, a, b) \in HMP_4$ and thus verify the coin.

Definition 10 (*HMP_4*)**.** $(x, m, a, b) \in HMP_4$ if

$$b = \begin{cases} x_1 \otimes x_{2+m} & \text{if } a = 0 \\ x_{3-m} \otimes x_4 & \text{if } a = 1 \end{cases}$$

4 Conclusion

We have shown the quantum computing steps necessary for the implementation of Gavinsky's quantum coin scheme. [3] We have provided an brief outline of the code necessary for it's implementation in Simulaqron [1] on GitHub [4]. We have also provided additional descriptions for various aspects of the protocol, building on the descriptions provided by Gavinsky.

References

[1] Axel Dahlberg and Stephanie Wehner. Simulaqron—a simulator for developing quantum internet software. *Quantum Science and Technology*, 4(1):015001, 2018.

[2] Juan Carlos Garcia-Escartin and Pedro Chamorro-Posada. Equivalent quantum circuits. *arXiv preprint arXiv:1110.2998*, 2011.

[3] Dmitry Gavinsky. Quantum money with classical verification. In *2012 IEEE 27th Conference on Computational Complexity*, pages 42–52. IEEE, 2012.

[4] Jerry Horgan Joao F. Santos, Hazel Murray. Quantum Coin GitHub Repository. `https://github.com/facocalj/quantumcoin`, November 2019.