

Implementing a Quantum Coin Scheme

Hazel Murray	Jerry Horgan	Joao F. Santos	David Malone	Harun Siljak
<i>Dept. Maths & Stats /</i>	<i>TSSG</i>	<i>CONNECT Centre</i>	<i>Dept. Maths & Stats /</i>	<i>CONNECT Centre</i>
<i>Hamilton Institute</i>	<i>Waterford Inst. of Technology</i>	<i>Trinity College Dublin</i>	<i>Hamilton Institute</i>	<i>Trinity College Dublin</i>
<i>Maynooth University</i>	Ireland	Ireland	<i>Maynooth University</i>	Ireland
Ireland	jhorgan@tssg.org	facocalj@tcd.ie	Ireland	harun.siljak@tcd.ie
hazel.murray@mu.ie			david.malone@mu.ie	

Abstract—Quantum computing has the power to break current cryptographic systems, disrupting online banking, shopping, data storage and communications. Quantum computing also has the power to support stronger more resistant technologies. In this paper, we describe a digital cash scheme created by Dmitry Gavinsky, which utilises the capability of quantum computing. We contribute by setting out the methods for implementing this scheme. For both the creation and verification of quantum coins we convert the algebraic steps into computing steps. As part of this, we describe the methods used to convert information stored on classical bits to information stored on quantum bits.

Index Terms—quantum, coins, banking, gates, qubits

I. INTRODUCTION

Quantum mechanics is the study of the smallest things in nature. At the 1927 Solvay Conference, 29 prominent physicists met to discuss the foundation of today's quantum theory. Amongst the participants were Albert Einstein, Marie Curie, Max Planck, Niels Bohr and Erwin Schrödinger. With their help, an understanding of quantum mechanics has allowed us to develop many modern technologies including MRI scanners, nuclear power, lasers, transistors and semiconductors [1].

Many years later, in 1980, computation using the principles of quantum mechanics was conceived. Benioff [2] showed that a computer could operate under the laws of quantum mechanics by providing a Schrödinger equation description of Turing machines. In 1988, Yamamoto and Igeta proposed the first physical realization of a quantum computer, it included the quantum equivalent of classical gates [3]. In 1991, Artur Ekert invented entanglement-based secure communication [4]. In 1998, a working 2-qubit quantum computer was built by Jones and Mosca at Oxford University [5]. This was the first experimental demonstration of a quantum algorithm. Since then, quantum devices have come a long way. In 2007, Switzerland used quantum technology to secure their voting systems [6]. In Japan, in 2010, a TV conference was secured using quantum key cryptography [7]. China installed a 2000km optical fibre

This publication has emanated from research supported in part by a research grant from Science Foundation Ireland (SFI) and is co-funded under the European Regional Development Fund under Grant 13/RC/2077. Hazel Murray was supported by an Irish Research Council 2017 Government of Ireland Postgraduate Scholarship. Harun Siljak's work was funded by European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 713567.

capable of quantum communication, which is being tested for use in banking and communications [8]. In 2015, a small quantum network was demonstrated by Delft University with plans to build a larger advanced quantum network across the Netherlands [9]. There are over forty multinational companies investing in quantum computing/communication [10]. These include IBM, Google, Microsoft and Intel.

Quantum computing has the theoretical power to break certain modern cryptography [11]. In 1994, Peter Shor developed a quantum algorithm that has the power to break some public key cryptographic systems [12], such as RSA. In 1996, Grover's algorithm was developed, which reduced the effectiveness of symmetric key cryptographic systems [13]. Without cryptography, much of our online banking, shopping and data storage technology would no longer be usable.

Though quantum computing has the power to break some of our current systems, it also holds the key to unlocking solutions that exceed the bounds of our current computational capabilities. Quantum technology has particularly useful qualities for applications to communication systems, privacy and security. In 2019, RIPE NCC [14] ran the first Pan-European Quantum Internet Hackathon. This event connected experts from six different locations and tasked them with solving open problems and developing technical infrastructure to allow the evolution of the Quantum Internet. Among other developments, teams successfully worked on Device-Independent Quantum Key Distribution, a Quantum version of Byzantine Agreement, Quantum Key Distribution in OpenSSL, Quantum-Cheque Protocol, Quantum Anonymous Transmission, Entanglement Routing and Quantum VPN. For more details on these projects see the Github repository [15].

This paper arose from work completed at the Irish node of the Pan-European Quantum Internet Hackathon [16]. Our goal was to develop the implementation steps necessary for a digital cash protocol based on quantum technologies; denoted a Quantum Coin Scheme. In this paper, we introduce quantum mechanics and describe its relevance to applications in banking and communication systems. We describe the mechanisms involved in creating and manipulating quantum bits. Finally, we describe contributions that allow for the implementation of Gavinsky's [17] theoretical quantum coin protocol.

In Sec. II-A, we explain the underlying properties of quantum mechanics that make it valuable for communication and computation technologies. In Sec. II-B, we describe related work and the development of quantum money. Sec. II-C introduces the notation and terminology used in this paper. In Sec. III, we describe one definition of a quantum coin (denoted a \mathcal{Q} -coin) and demonstrate the steps necessary for creating it. This involves the creation of a method for converting classical bits to quantum information. This is used to show how to create quantum coins for use in quantum money transactions. Sec. IV details the processes necessary for using these \mathcal{Q} -coins in the implementing of Gavinsky's quantum coin validation. Sec. V summarises some features of the scheme.

II. BACKGROUND

A. The Power Of Quantum Computing

Quantum mechanics is interesting because it contains properties that are at odds with our general understanding of classical physics. Here we will give a brief overview of the properties we utilise. Many more detailed descriptions are available (e.g. [1]).

Used for information storage, a classical bit can take the value 0 or 1. A qubit is the quantum equivalent to a classical bit. Qubits have three important properties that makes them fascinating as an alternative to our classical view of information: *superposition*, *measurement* and *entanglement*.

The first property, *superposition*, describes the fact that a qubit can take the value of both 0 and 1 at the same time! Imagine we have two classical bits, these can represent 4 states: either both bits are zero: 00, one bit is zero and the other is one: 01 or 10, or both bits are one: 11. If we have 2 qubits, we can still represent these 4 states: 00, 01, 10 and 11. However, because of superposition, the 2 qubits can represent a mix of all 4 states at the same time. This gives quantum computers the capacity to complete computations in parallel and where n classical bits allow n computations, n qubits can allow 2^n computations.

The second property is *measurement*. In classical mechanics, looking at something does not change its state. In quantum mechanics, a qubit can be in a superposition of both 0 and 1 at the same time and when measured it must *collapse* to either 0 or 1. The state of a quantum bit is represented by a wave function, where $|0\rangle$ is the 0 wave function, $|1\rangle$ is the 1 wave function, and $\alpha|0\rangle + \beta|1\rangle$ is a superposition. A wave function that is composed of only $|0\rangle$ or $|1\rangle$ is called an *eigenstate*. If we measure a wave function to see if it is a 0 or a 1, then there is a probability $|\alpha|^2$ of measuring 0 and $|\beta|^2$ of 1. Naturally, we need to normalise so that $|\alpha|^2 + |\beta|^2 = 1$. Each qubit can be represented as a wave function and on measurement of the wave function as 0 or 1 it collapses and becomes $|0\rangle$ or $|1\rangle$. This has implications for security. If we send classical bits from one place to another, we have no way to know whether they were observed by a malicious user. However, if we communicate using qubits, a malicious user who observes the qubits will collapse the wave function and we will know that the message was intercepted.

One interesting thing to note, is that we are collapsing the wave function for the property we are measuring, this is called the *basis* that we are measuring with respect to. Imagine there are two measurements on a qubit, say its position can be A or B and its momentum can be 0 or 1. We measure its position and the wave function collapses to $|A\rangle$. If we continue to consecutively measure with respect to the position basis then we will continue to get A . If we then measure using the momentum basis, the momentum wave function collapses to $|1\rangle$, and the position variable is again probabilistic. So if we remeasure the position it will return either A or B with some probability. It is true for any measurable qubit attributes. This is an example of the famous Heisenberg Uncertainty Principle [18].

The third and, according the Einstein, the 'spooky' property of quantum mechanics is *entanglement*. Take two qubits that are entangled and let us move them to opposite ends of the globe. If we measure one of the qubits then we know that we will get the same measurement for the second, entangled, qubit. Imagine we take the first qubit and measure it using a momentum basis and get 1. Then the other qubit will also measure as 1. This is remarkable since each returned result is a function of probabilities $|\alpha|^2$ and $|\beta|^2$. This relationship gives us the ability to send information via these two entangled qubits (but not faster-than-light, as we might be tempted to attempt [19]).

These properties have applications in our computing and communications infrastructure. We are going to look at the applications of qubits to our online representation of coins that are used to transfer funds between bank accounts.

B. Quantum Money

In classical cryptography the concept of digital cash has been well-explored [20]. Let us briefly describe a classical digital cash scheme.

Every coin issued by the bank is represented by a secret string s . These strings are known to the bank and to the current coin holder (Alice). Suppose Alice wishes to pay Bob, she will want to pass her coin to Bob:

- Alice sends her string s to the Bank and tells the bank she wants to send the coin to Bob,
- The bank checks if the string sent by Alice is valid. If so, the bank erases the string s from the list of valid strings and adds a newly generated secret string s' to the list.
- The bank sends s' to Bob; henceforth, Bob holds the coin.

For classical digital cash schemes, the main concern is the double-spending problem, where a user spends the same digital coin multiple times. One solution, as above, is to include a verification of each token with a bank. However, an intruder who pretends to be the bank can steal a valid coin from its fair holder who wants it to be verified.

Definition 1 (Coin): A coin is a unique object that can be created by a trusted mint (or bank) and then circulated among untrusted holders.

For quantum money we will also need our coins to be non-counterfeitable. Conveniently, qubits have a property described as the *no-cloning theorem* [21] that makes them perfect to be applied to quantum money. The no cloning theorem tells us that it is impossible to create an identical copy of a quantum state. Wiesner argued that this property allows us to create quantum coins that are unforgeable, something that is impossible with our classical physical money. In 1983, Wiesner [22] and Bennett, Brassard, Breidbard, and Wiesner [23] conceived the first quantum money schemes.

In 2003, Tokunaga, Okamoto, and Imoto give a scheme for non-transferable anonymous quantum cash with online verification [24]. In 2010, Mosca and Stebila present a new type of quantum money which they call quantum coins [25]. These coins are transferable, locally verifiable, and unforgeable, and have some anonymity properties. However both these schemes require quantum communication with a bank and are also both susceptible to an adaptive attack conceived by Lutomirski [26].

In 2012, Gavinsky proposed a new quantum coin scheme that allows classical verification of coins. In the version of a quantum internet where quantum and classical computers will work in synchrony this is an ideal scheme. We can leverage the power of quantum bits without the requirement for every user to possess quantum communication technology. Gavinsky's scheme is secure against adaptive adversaries, the coins are exponentially hard to counterfeit, verification can be conducted via insecure communication lines, the bank's database is static and can therefore be decentralized, and the scheme protects against a malicious user masquerading as a bank. The coins are limited to a certain number of verifications, which trade off against the size of the coin (number qubits). However, Gavinsky shows that this dependency is optimal.

In this paper we outline the methods necessary for implementing Gavinsky's quantum coin scheme. We specifically describe the physical gates necessary for the creation and verification of the quantum coins.

C. Notation

This section introduces the notation used in the paper.







1) *Matrix representation of qubits:* In Sec. II-A, we explained that a qubit can be in a superposition of both 0 and 1 and is represented as the vector:

$$q = \alpha|0\rangle + \beta|1\rangle,$$

where states $|0\rangle$ and $|1\rangle$ form a basis for the vector space and α and β are complex numbers that indicate the amplitude of the state. The amplitude squared tells us the probability of the state occurring. The above vector describes one qubit that can be in a superposition of two states. In this paper we are generally working with 2 qubits, which have 4 possible states. We call these states $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$. These states can also be described in matrix form as:

$$|00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \text{ and } |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

TABLE I
OVERVIEW OF SIX QUANTUM GATES

Name	Gate	Matrix	Description
Hadamard		$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$	Maps $ 0\rangle$ to $\frac{ 0\rangle+ 1\rangle}{\sqrt{2}}$ and $ 1\rangle$ to $\frac{ 0\rangle- 1\rangle}{\sqrt{2}}$. It sends a qubit into a superposition.
Pauli-X		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	Maps $ 0\rangle$ to $ 1\rangle$ and vice versa. Equivalent of the classical NOT gate.
Pauli-Z		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	It leaves the basis state $ 0\rangle$ unchanged and maps $ 1\rangle$ to $- 1\rangle$. It is sometimes called phase-flip.
CNOT		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$	Flips the second qubit (the target qubit) and only if the first qubit is $ 1\rangle$. The CNOT gate allows us to entangle two input qubits.
SWAP		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	The swap gate swaps two qubits.
Identity		$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	Leaves the basis states $ 0\rangle$ and $ 1\rangle$ unchanged. It can be used to expand gates so that they can work on multiple qubits.

The four states together form the *basis*. Each state has a certain probability of occurring, determined by the amplitudes α, β, γ and δ of the wave function. The basis matrix times the amplitude vector gives us the wave function for our two qubits:

$$\begin{bmatrix} |00\rangle & |01\rangle & |10\rangle & |11\rangle \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{bmatrix} = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle.$$

For simplicity, given the context of the basis, we can just report the amplitude matrix when describing the qubit pair. We use the subscript A to denote an amplitude matrix: $[\alpha \ \beta \ \gamma \ \delta]_A$.

2) *Quantum gates:* Both classical and quantum logic gates take binary inputs and produce a single binary output. Quantum gates, like classical gates, can be combined into a circuit. One benefit of quantum gates is that, unlike classical gates, they are always reversible. This means that no information is lost when qubits travel through quantum gates.

In this paper we will use six quantum gates. In Tab. I, we define each gate by stating the gate's function, symbol and matrix representation. To learn more about quantum gates see [27].

III. CREATION OF Q COINS

Let us describe Gavinsky's definition of a coin, named Q -coin, then we will describe how it is created.

Definition 2 (Q -coins): For each coin, a bank holds a secret record consisting of k entries x_1, \dots, x_k s.t. $x_i \in \{0, 1\}^4$ (i.e., the secret record contains k strings of 4 classical bits).

A "fresh" Q -coin is then created corresponding to this record (x_1, \dots, x_k) . **The coin consists of:**

- k quantum registers consisting of 2 qubits each, where the i 'th register contains a specific state $|\alpha(x_i)\rangle$;
- a k -bit classical register \mathcal{P} . This consists of k binary markers that indicate whether the i 'th quantum register has been used in previous validation processes. The values of \mathcal{P} are initially set to 0^k ;
- a unique identification number.

Creation of the coins requires the conversion of the four classical bits to two quantum bits. Algebraically, we use the formula below for conversion. Because this conversion satisfies the 4-bit version of the Hidden Matching Problem (HMP) [28], Gavinsky calls these quantum registers HMP_4 -states.

Definition 3 (HMP_4 -states): Let $x \in \{0, 1\}^4$. The corresponding HMP_4 -state is

$$|\alpha(x)\rangle \stackrel{\text{def}}{=} \frac{1}{\sqrt{4}} \sum_{1 \leq i \leq 4} (-1)^{x_i} |(i-1)_2\rangle,$$

where $(\cdot)_2$ denotes writing a number in base 2.

For example, the 4 classical bits $x = 0110$ are converted to the state $|\alpha(0110)\rangle$, which is

$$\begin{aligned} \frac{1}{\sqrt{4}} & ((-1)^0 |00\rangle + (-1)^1 |01\rangle + (-1)^1 |10\rangle + (-1)^0 |11\rangle) \\ & = \frac{1}{2} (|00\rangle - |01\rangle - |10\rangle + |11\rangle). \end{aligned}$$

Up to normalisation, this can be represented by the following amplitude matrix $[1 \ -1 \ -1 \ 1]_A$.

A. Implementation

Q -coins require a quantum representation of 4 bit classical strings. There are 16 possible combinations of 4 bits. Each of these needs to be uniquely represented by a quantum register according to the conversion specified in Def. 3. In this section, we show how to prepare these 16 HMP_4 -states.

1) *No entanglement:* Given any two quantum bits. Let $q_1 = \alpha |0\rangle_1 + \beta |1\rangle_1$ and $q_2 = \gamma |0\rangle_2 + \delta |1\rangle_2$, where the subscript denotes the qubit the state belongs to. The state space of a composite systems is the tensor product of the state spaces of the components, so for our two qubits

$$\begin{aligned} q_1 \otimes q_2 &= (\alpha |0\rangle_1 + \beta |1\rangle_1)(\gamma |0\rangle_2 + \delta |1\rangle_2) \\ &= \alpha\gamma |00\rangle + \alpha\delta |01\rangle + \beta\gamma |10\rangle + \beta\delta |11\rangle \\ &\equiv [\alpha\gamma \ \alpha\delta \ \beta\gamma \ \beta\delta]_A \end{aligned}$$

For simple input states, α, β, γ and δ can each take either -1 or +1. Thus, by manipulating the state of the initial qubits, q_1 and q_2 , we can create the following state spaces:

$$\begin{aligned} \alpha = -1 \quad \beta, \gamma, \delta = 1 &\rightarrow [-1 \ -1 \ 1 \ 1]_A \stackrel{\text{def}}{=} |Q_1\rangle \\ \beta = -1 \quad \alpha, \gamma, \delta = 1 &\rightarrow [1 \ 1 \ -1 \ -1]_A \stackrel{\text{def}}{=} |Q_2\rangle \\ \gamma = -1 \quad \alpha, \beta, \delta = 1 &\rightarrow [-1 \ 1 \ -1 \ 1]_A \stackrel{\text{def}}{=} |Q_3\rangle \\ \delta = -1 \quad \alpha, \beta, \gamma = 1 &\rightarrow [1 \ -1 \ 1 \ -1]_A \stackrel{\text{def}}{=} |Q_4\rangle \\ \alpha, \beta = -1 \quad \gamma, \delta = 1 &\rightarrow [-1 \ -1 \ -1 \ -1]_A \stackrel{\text{def}}{=} |Q_5\rangle \\ \alpha, \gamma = -1 \quad \beta, \delta = 1 &\rightarrow [1 \ -1 \ -1 \ 1]_A \stackrel{\text{def}}{=} |Q_6\rangle \\ \alpha, \delta = -1 \quad \beta, \gamma = 1 &\rightarrow [-1 \ 1 \ 1 \ -1]_A \stackrel{\text{def}}{=} |Q_7\rangle \\ \alpha, \beta, \gamma, \delta = -1 &\rightarrow [1 \ 1 \ 1 \ 1]_A \stackrel{\text{def}}{=} |Q_8\rangle. \end{aligned}$$

All other combinations give repetitions of these 8 HMP_4 -states, so we use entanglement to create the other states.

2) *With entanglement:* To convert the remaining 8 classical strings, we begin by entangling the 2 input qubits, q_1 and q_2 . We then send these entangled qubits through gates to manipulate them to create the 8 required quantum registers.

We create entangled states by taking simple inputs and putting them through a Hadamard and a CNOT gate. The Hadamard gate is applied to the first qubit and sends it into a superposition. Then the CNOT gate is applied to both qubits. This conditional gate creates an entanglement between the two qubits. See Tab. I for the matrix description of both gates.

By specifying four different initial states of the qubits, q_1 and q_2 , we can create four different entangled states. These are called *Bell states*.

As an example, by starting both qubits in the eigenstate $|0\rangle$ we create the first Bell state, called $|\Phi^+\rangle$:

$$\left. \begin{array}{l} |0\rangle \text{---} \boxed{H} \text{---} \bullet \text{---} \\ |0\rangle \text{---} \oplus \end{array} \right\} = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \equiv [1 \ 0 \ 0 \ 1]^T \stackrel{\text{def}}{=} |\Phi^+\rangle$$

Definition 4 (Bell States): The Bell states are four specific maximally entangled quantum states of two qubits given by:

$$\begin{aligned} |\Phi^+\rangle &= \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle) \equiv [1 \ 0 \ 0 \ 1]^T \\ |\Phi^-\rangle &= \frac{1}{\sqrt{2}} (|00\rangle - |11\rangle) \equiv [1 \ 0 \ 0 \ -1]^T \\ |\Psi^+\rangle &= \frac{1}{\sqrt{2}} (|01\rangle + |10\rangle) \equiv [0 \ 1 \ 1 \ 0]^T \\ |\Psi^-\rangle &= \frac{1}{\sqrt{2}} (|01\rangle - |10\rangle) \equiv [0 \ 1 \ -1 \ 0]^T \end{aligned}$$

Using the Bell states we can generate the required remaining 8 combinations by using a sequence of quantum gates.

For example, if we create the Bell state $|\Phi^+\rangle$ and pass it through an extended Hadamard gate, $(H \otimes I)$, we can create a ninth HMP_4 -state; $[1 \ 1 \ 1 \ -1]_A$:

$$\left. \begin{array}{l} |0\rangle \text{---} \boxed{H} \text{---} \bullet \text{---} \boxed{H} \text{---} \\ |0\rangle \text{---} \oplus \text{---} \boxed{I} \end{array} \right\} = \frac{|00\rangle + |01\rangle + |10\rangle - |11\rangle}{2} = [1 \ 1 \ 1 \ -1]_A$$

Below we describe the input Bell state and the combination of gates used to create the last 8 linear combinations. I denotes

the identity matrix, H denotes the Hadamard gate, \otimes denotes the tensor product (gates wired in parallel) and \times denotes the ordinary matrix cross product (serially wired gates).

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

We have shown how to create 16 quantum registers which uniquely represent the 16 classical combinations of 4 bits. For creation of the quantum coin, the mapping of classical bits to quantum bits can be hard-coded and only needs to be completed once. As described in [29], there will be alternative circuit configurations that will produce equivalent results.

IV. VERIFICATION

Let us now introduce Gavinsky's coin verification protocol. The protocol involves the key holder proving that they hold the coin, without needing to reveal its full details to the bank. It is a version of a zero knowledge protocol.

As described in Def. 2, coins with unique identification numbers have been created by the bank. The bank holds a secret record (x_1, \dots, x_k) associated with the identification number for each one of its created coins. A coin holder Bob has one such \mathcal{Q} -coin which contains the following information:

Identification number	
\mathcal{P}_1	$ \alpha(x_1)\rangle$
\mathcal{P}_2	$ \alpha(x_2)\rangle$
\vdots	\vdots
\mathcal{P}_k	$ \alpha(x_k)\rangle$

\mathcal{P}_i marks whether the quantum register, $|\alpha(x_i)\rangle$, at position i has previously been used for verification. The coin holder, Bob, wishes to verify the coin's authenticity.

The verification is based on the communication complexity problem called the Hidden Matching Problem introduced by Bar-Yossef et al. [28]. The Hidden Matching Problem (HMP) is defined as follows:

Definition 5 (HMP_4 condition): For $x \in \{0, 1\}^4$ and $m, a, b \in \{0, 1\}$, we say that $(x, m, a, b) \in HMP_4$ if

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

In the below protocol, Bob will provide values (a_i, b_i) to the bank. The bank holds the values x_i (the classical bit strings) and m_i . If $\forall i (x_i, m_i, a_i, b_i) \in HMP_4$, then the bank can verify that Bob does in fact hold the \mathcal{Q} -coin corresponding to the classical values x_i . We will now describe the specific steps involved and the methods for implementation.

A. Steps in Gavinsky's protocol

Coin Holder	Bank
Step 1: The holder sends the identification number on the coin they hold to the bank.	
$\boxed{\text{ID number}} \rightarrow$	
Step 2: The bank uses the identification number to look up the secret record of k classical strings, (x_1, \dots, x_k) , which were created for this coin.	
The bank chooses t indexes (s.t. $3 t$ & $t \leq k$) at random between 1 and k and sends them to the coin holder:	
	$\mathcal{L}_{bank} \subset [k],$ s.t. $ \mathcal{L}_{bank} = t$ and $3 t$ $\leftarrow \boxed{\mathcal{L}_{bank}}$
Step 3: The holder randomly selects $2t/3$ of the values sent by the bank that have not been used for validation before, i.e. $\mathcal{P}_i = 0$:	
$\mathcal{L}_{holder} \subset \mathcal{L}_{bank},$ s.t. $\forall i \in \mathcal{L}_{holder} \mathcal{P}_i = 0$ and $ \mathcal{L}_{holder} = 2t/3.$	
The holder sends \mathcal{L}_{holder} to the bank and marks those elements as used in the register \mathcal{P} :	
$\mathcal{P}_i = 1 \forall i \in \mathcal{L}_{holder}$ $\boxed{\mathcal{L}_{holder}} \rightarrow$	
Step 4: For each index in \mathcal{L}_{holder} , the bank randomly chooses an m equal to 0 or 1 and sends these back to the coin holder:	
	$\forall i \in \mathcal{L}_{holder}$ $m_i \in \{0, 1\}$ $\leftarrow \boxed{m_i}$
Step 5: The holder measures the quantum registers, $ \alpha(x_i)\rangle, \forall i \in \mathcal{L}_{holder}$. The basis used for the measurement is determined by the value m sent by the bank:	
$\forall i \in \mathcal{L}_{holder}$ measure $ \alpha(x_i)\rangle \Rightarrow (a_i, b_i).$	
The coin holder sends the output values (a_i, b_i) corresponding to each $i \in \mathcal{L}_{holder}$ to the bank:	
$\boxed{(a_i, b_i)} \rightarrow$	
Step 6: The bank checks whether $(x_i, m_i, a_i, b_i) \in HMP_4$ for all $i \in \mathcal{L}_{holder}$ (by Def. 5):	
	if $(x_i, m_i, a_i, b_i) \in HMP_4$ $\forall i \in \mathcal{L}_{holder}$ $\leftarrow \boxed{\text{Coin is Valid}}$

Note that a bank produces *fresh* \mathcal{Q} -coins but as a \mathcal{Q} -coin goes through more and more verification protocols, its quantum registers lose their original content; when a quantum state/register is measured it collapses. For each quantum verification we measure $2t/3$ quantum registers. Hence, 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 .

We will expand on the steps in the above protocol and translate them into computational steps. We begin by describing the implementation of Steps 1 to 4 in Sec. IV-B, the implementation of Step 5 in Sec. IV-C, and finally we describe Step 6 in Sec. IV-D.

B. Implementation of Steps 1–4

Steps 1–4 are classical steps. They involve classical correspondence between Bob and the bank. Below we include the code for both parties. The code is written for SimulaQron [30]. SimulaQron is a free quantum internet simulator. It allows users to program their your own quantum internet applications.

Code for bank — Alice

```
def verify_coin(register,t):
    #Step 2
    register_c = list(register)
    m_s = []
    list_of_random_indexes =
        random.sample(register_c, t)
    with CQCCConnection("Alice") as bank:
        # Step 2 cont; send the list of indexes
        to the coin holder
        bank.sendClassical("Bob",
            list_of_random_indexes)
        # Wait for receiver to send back subset
        of list
        #Receive output from step 3
        index_list = bank.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))
        bank.sendClassical("Bob", m_s)
```

Code for coin holder — Bob

```
def verify_coin():
    print("Verify Coin ID 1")
    with CQCCConnection("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 bank 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()
```

Additional code for our SimulaQron programs can be found on Github in files `bank.py` and `bob.py` [31].

C. Step 5

The holder receives the values m corresponding to each element of \mathcal{L}_{holder} . The value m is used to specify the basis, $\{v_1, v_2, v_3, v_4\}$, which each quantum register (with indexes $i \in \mathcal{L}_{holder}$) will be measured with respect to.

1) *Method*: Formally m is defined as an HMP_4 -query. The bank queries Bob for the measurements of the i quantum registers with respect to specific m values. The outputted values will satisfy the HMP_4 condition.

It is known that $(x, m, a, b) \in HMP_4$ always [28].

Definition 6 (HMP₄-queries): If $m = 0$, let

$$v_1 \stackrel{\text{def}}{=} \frac{|00\rangle + |01\rangle}{\sqrt{2}}, v_2 \stackrel{\text{def}}{=} \frac{|00\rangle - |01\rangle}{\sqrt{2}}, v_3 \stackrel{\text{def}}{=} \frac{|10\rangle + |11\rangle}{\sqrt{2}}, v_4 \stackrel{\text{def}}{=} \frac{|10\rangle - |11\rangle}{\sqrt{2}}$$

otherwise if $m = 1$, let

$$v_1 \stackrel{\text{def}}{=} \frac{|00\rangle + |10\rangle}{\sqrt{2}}, v_2 \stackrel{\text{def}}{=} \frac{|00\rangle - |10\rangle}{\sqrt{2}}, v_3 \stackrel{\text{def}}{=} \frac{|01\rangle + |11\rangle}{\sqrt{2}}, v_4 \stackrel{\text{def}}{=} \frac{|01\rangle - |11\rangle}{\sqrt{2}}$$

Measure $|\alpha(x)\rangle$ in the basis $\{v_1, v_2, v_3, v_4\}$. Let

$$(a, b) = \begin{cases} (0, 0), & \text{if } v_1 \\ (0, 1), & \text{if } v_2 \\ (1, 0), & \text{if } v_3 \\ (1, 1), & \text{if } v_4. \end{cases}$$

2) *Implementation*: Step 5 involves generating an output (a, b) determined by the outcome of measuring with the basis determined by m . As per Sec. II-A, the basis describes what property of the qubit we are measuring and specifically what gates the qubits are passed through. The values $\{v_1, v_2, v_3, v_4\}$ specify the column vectors that make up our basis matrix.

For example, if $m = 0$ then Bob is told to use the following vectors to form the basis matrix:

$$v_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, v_2 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \end{bmatrix}, v_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}, v_4 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \end{bmatrix}$$

The matrix formed by these vectors is equivalent to 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 $m = 1$, the basis is an expanded Hadamard gate and a SWAP gate:

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

Therefore, given each value of m Bob can send each quantum register $|\alpha(x_i)\rangle$ through the appropriate sequence of gates. The output will be either v_1, v_2, v_3 or v_4 and Bob will return an (a, b) corresponding to the v each register returns.

This has shown how to construct the measurement vectors for implementation using standard quantum gates.

D. Step 6

In the final step the bank receives a value (a, b) corresponding to each index in \mathcal{L}_{holder} . The bank knows the classical value $x \in \{0, 1\}^4$ corresponding to each index position and the given $m \in \{0, 1\}$. So the bank uses Def. 5 to verify whether $\forall i \in \mathcal{L}_{holder} (x_i, m_i, a_i, b_i) \in HMP_4$ and thus verify the coin. This is a classical step and involves checking whether each value a sent by Bob gives the specified b output and thus Bob's provided (a, b) pairs satisfy the HMP_4 condition.

V. SUMMARY REMARKS

We briefly review properties of the quantum coin scheme. Many of these arise from its basic design, rather than our implementation.

a) *Security*: To compromise a coin an adversary must supply the correct pair (a_i, b_i) corresponding to the $m = 0$ or 1 chosen by the bank for each register i . Gavinsky shows that makes the coins exponentially hard to counterfeit [17]. In addition, the coins cannot be directly cloned due to the quantum no-cloning theorem and can not be eavesdropped without collapsing the wave functions. Furthermore, unlike classical digital cash schemes, an intruder who pretends to be the bank cannot steal a valid coin from the holder. In fact, Gavinsky proves that this scheme is unconditionally secure even against adaptive multi-round attackers [17].

b) *Efficiency*: The number of verifications that a Q -coin can go through is limited and there is a trade off between the size of the coin and the security of that coin. Gavinsky [17] shows that this dependence is optimal up to a polynomial.

The database of the bank is static, and therefore decentralised branches can exist which can perform verification. In addition the classical communication channel with the bank can remain unencrypted.

Our implementation uses common quantum gates. Other pairings or other gates could prove more efficient as the technology progresses. An integrated circuit of the gates could be used to improve the efficiency of the system.

c) *Limitation*: One current limitation with quantum coins is the need for the quantum entanglement to persist for the lifetime of the coin. However, quantum entanglement is highly susceptible to decoherence, even tiny changes in its environment, such as atomic motion, can cause the entanglement to collapse. Most quantum entanglement technologies protect against decoherence from tens of μs (micro-seconds) to tens of seconds. However Ion Traps show promise with ~ 12 days of entanglement lifetime. Unless a very high rate of coin turnover is envisaged then quantum memories will be required. Quantum memories are being developed [32], but still have short-lifetimes, $\sim 70\mu\text{s}$, as to not be practical at present.

VI. CONCLUSION

In this paper, we have shown how to convert information stored on classical bits to information stored on quantum bits. We have demonstrated a quantum gate configuration that allows the implementation of Gavinsky's quantum coin scheme [17]. We have provided a brief outline of the code necessary for its deployment in SimulaQron [30]. We have also provided additional descriptions for various aspects of the protocol, building on the descriptions provided by Gavinsky.

REFERENCES

- [1] D. A. Miller, *Quantum mechanics for scientists and engineers*. Cambridge University Press, 2008.
- [2] P. Benioff, "The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines," *J. statistical physics*, vol. 22, no. 5, pp. 563–591, 1980.
- [3] K. Igeta and Y. Yamamoto, "Quantum mechanical computers with single atom and photon fields," in *International Quantum Electronics Conf. Optical Society of America*, 1988, p. Tu14.
- [4] A. K. Ekert, "Quantum cryptography based on Bell's theorem," *Physical review letters*, vol. 67, no. 6, p. 661, 1991.
- [5] J. A. Jones and M. Mosca, "Implementation of a quantum algorithm on a nuclear magnetic resonance quantum computer," *J. chemical physics*, vol. 109, no. 5, pp. 1648–1653, 1998.
- [6] P. Marks, "Quantum cryptography to protect Swiss election," *New Scientist*, 2007.
- [7] M. Sasaki, "One more step to commercialization: Study of quantum cryptographic network," https://www.nict.go.jp/en/pdf/copy_of_NICT_NEWS_1102_E.pdf.
- [8] J. Qiu *et al.*, "Quantum communications leap out of the lab," *Nature*, vol. 508, no. 7497, pp. 441–442, 2014.
- [9] J. Markoff, "Sorry, Einstein. Quantum study suggests 'spooky action' is real," *The New York Times*, vol. 21, 2015.
- [10] P. Vermaas, D. Nas, L. Vandersypen, and D. Elkouss Coronas, "Quantum internet: The internet's next big step," 2019.
- [11] V. Mavroeidis, K. Vishi, M. D. Zych, and A. Jøssang, "The impact of quantum computing on present cryptography," *arXiv preprint arXiv:1804.00200*, 2018.
- [12] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proc. 35th Symposium on Foundations of Computer Science*. IEEE, 1994, pp. 124–134.
- [13] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. 28th ACM Symposium on Theory of Computing*, ser. STOC '96. New York, NY, USA: Association for Computing Machinery, 1996, p. 212–219. [Online]. Available: <https://doi.org/10.1145/237814.237866>
- [14] RIPE Network Coordination Centre, <https://www.ripe.net/>.
- [15] "Quantum Internet Hackathon," <https://github.com/PEQI19>, Nov. 2019.
- [16] "Connect hosts pan-European Quantum Internet Hackathon," <https://www.techcentral.ie/connect-hosts-pan-european-quantum-internet-hackathon/>, Nov. 2019.
- [17] D. Gavinsky, "Quantum money with classical verification," in *2012 IEEE 27th Conference on Computational Complexity*. IEEE, 2012, pp. 42–52.
- [18] W. Heisenberg, "Über den anschaulichen inhalt der quantentheoretischen kinematik und mechanik," in *Zeitschrift für Physik* 43. Springer, 1927, pp. 478–504.
- [19] D. Bruss, G. D'Ariano, C. Macchiavello, and M. Sacchi, "Approximate quantum cloning and the impossibility of superluminal information transfer," *Phys Rev A*, vol. 62, no. 6, p. 062302, 2000.
- [20] D. Chaum, A. Fiat, and M. Naor, "Untraceable electronic cash," in *Conf. Theory and Application of Cryptography*. Springer, 1988, pp. 319–327.
- [21] W. Wootters and W. Zurek, "Quantum no-cloning theorem," *Nature*, vol. 299, p. 802, 1982.
- [22] S. Wiesner, "Conjugate coding," *ACM Sigact News*, vol. 15, no. 1, pp. 78–88, 1983.
- [23] C. H. Bennett, G. Brassard, S. Breidbart, and S. Wiesner, "Quantum cryptography, or unforgeable subway tokens," in *Advances in Cryptology*. Springer, 1983, pp. 267–275.
- [24] Y. Tokunaga, T. Okamoto, and N. Imoto, "Anonymous quantum cash," in *ERATO Conference on Quantum Information Science (EQIS)*, 2003.
- [25] M. Mosca and D. Stebila, "Quantum coins," *Error-Correcting Codes, Finite Geometries and Cryptography*, vol. 523, pp. 35–47, 2010.
- [26] A. Lutomirski, "An online attack against Wiesner's quantum money," *arXiv preprint arXiv:1010.0256*, 2010.
- [27] M. A. Nielsen and I. Chuang, *Quantum computation and quantum information*. American Association of Physics Teachers, 2002.
- [28] Z. Bar-Yossef, T. S. Jayram, and I. Kerenidis, "Exponential separation of quantum and classical one-way communication complexity," in *Proc. 36th ACM symposium on Theory of computing*, 2004, pp. 128–137.
- [29] J. C. Garcia-Escartin and P. Chamorro-Posada, "Equivalent quantum circuits," *arXiv preprint arXiv:1110.2998*, 2011.
- [30] A. Dahlberg and S. Wehner, "Simulaqron — a simulator for developing quantum internet software," *Quantum Science and Technology*, vol. 4, no. 1, p. 015001, 2018.
- [31] J. F. Santos, H. Murray, and J. Horgan, "Quantum Coin GitHub Repository," <https://github.com/facocalj/quantumcoin>, Nov. 2019.
- [32] Y. Yu, F. Ma, X.-Y. Luo, B. Jing, P.-F. Sun, R.-Z. Fang, C.-W. Yang, H. Liu, M.-Y. Zheng, X.-P. Xie *et al.*, "Entanglement of two quantum memories via fibres over dozens of kilometres," *Nature*, vol. 578, no. 7794, pp. 240–245, 2020.