QuantCoin$^{\text{TM}}$ – A quantum computation-backed cryptocurrency

Peter P. Rohde[1, *]

[1] *Centre for Quantum Computation and Intelligent Systems (QCIS), Faculty of Engineering & Information Technology, University of Technology Sydney, NSW 2007, Australia*

(Dated: May 19, 2018)

The Bitcoin mining process involves finding bit-strings that hash under SHA256 to a value within some relatively small range. This so-called 'proof-of-work' principle associates computational complexity with the mining process, and since the hashing functions are one-way functions, they must be evaluated via brute-force trial-and-error to find hits.

However, what a waste this is! Our proof-of-work is nothing more than hashing a huge number of random bit-strings, computations which are of no intrinsic value to anyone. The market value in turn has nothing to do with any inherent value earned during the mining process. Rather it is based purely on the psychology of scarcity, since there is an upper bound on the number of Bitcoins that satisfy the legitimacy constraint.

What if we were to replace brute-force hashing of random data with computations of genuine monetary value? Then we would have a sounder currency, whose value derives from the monetary cost of executing useful computations. While it is not so easy to invent such a protocol for classical computation, the idea lends itself very naturally to quantum computation, owing to their ability to undergo encrypted computation and be subject to verification protocols.

Building upon some of the pricing models introduced earlier in this section, there are two main candidates for backing a cryptocurrency with quantum compute-time: the spot market (i.e we execute the computation immediately); and the futures market (i.e we own the right to utilise the computer at some fixed time in the future). We consider the merits of both these candidates.

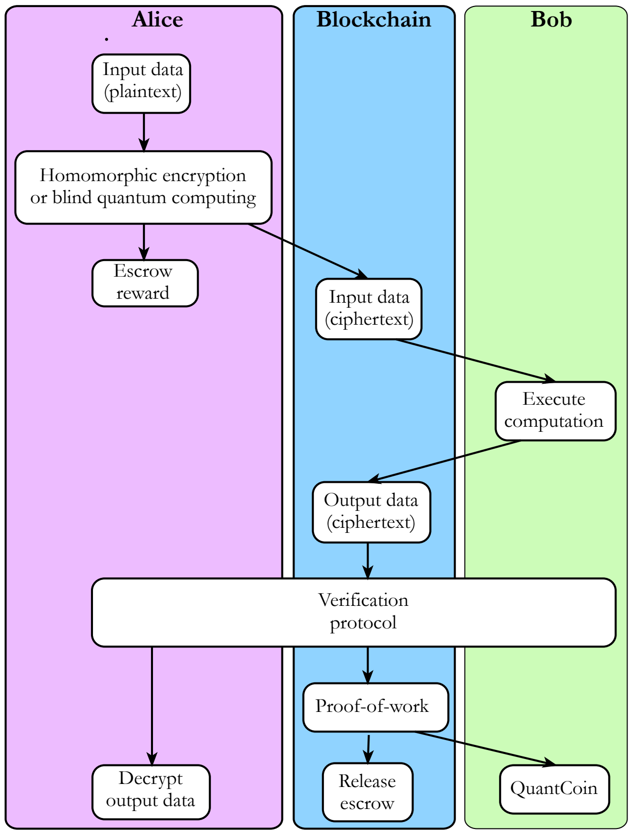

FIG. 1: Flowchart for the QuantCoin$^{\text{TM}}$ protocol, introduced in Alg. I.

1. *Spot market model*

In Alg. I we provide a very rough sketch for how a protocol based on the spot market might be constructed. A corresponding graphical flowchart is shown in Fig. 1. We present the ideas in a very high-level manner, abstracting away the physical implementation details of the computation, encryption, and verification protocols, instead envisaging that we can interface with them using a very high-level API.

It is evident from the flow of Alg. I that the mining process now comprises solving an actual quantum computation of intrinsic value to Alice, since she is exchanging assets (e.g dollars or already-existing QuantCoins$^{\text{TM}}$) in exchange for the computation. Completion of the computation followed by successful verification then further rewards Bob with a fresh QuantCoin$^{\text{TM}}$ courtesy of the distributed Blockchain algorithm.

The described protocol, in addition to mining a new coin, associated with the execution of a computation, acts as a currency converter for converting traditional assets (e.g dollars) into QuantCoins$^{\text{TM}}$. This ability to currency convert is necessary, and completely differs from the original Bitcoin mining process, where coins are fabricated out of thin air by anyone and everyone, independent of their pre-existing monetary wealth – Bitcoins are not created via conversion from any existing asset. The fact that the computation associated with each QuantCoin$^{\text{TM}}$ is of intrinsic value on the other hand, implies that Alice

*dr.rohde@gmail.com; URL: http://www.peterrohde.org

```
function QuantCoin($\hat{U}_{\texttt{comp}}$, blockchain, data, reward):
  1. Alice homomorphically/blindly encrypts $data$,
       $encryptedInput = homoEncrypt(data)$    (1)
  2. Alice commits the $encryptedInput$ to the public $blockchain$,
       $blockchain.commit(encryptedInput)$    (2)
  3. Alice places into escrow Bob's $reward$,
       $escrow.hold(reward)$    (3)
  4. Bob processes the $encryptedInput$ using $\hat{U}_{\texttt{comp}}$,
       $encryptedOutput = \hat{U}_{\texttt{comp}}(encryptedInput)$    (4)
  5. Bob commits encrypted output to the $blockchain$,
       $blockchain.commit(encryptedOutput)$    (5)
  6. Alice and Bob execute a verification protocol,
       $proof = verify(encryptedOutput)$    (6)
  7. The $proof$ is committed to the $blockchain$ (as is every step of the proof if it is an interactive one),
       $blockchain.commit(proof)$    (7)
  8. The $blockchain$ network inspects that the verification is valid.
  9. If valid, the $blockchain$ cryptographically hashes the data and proof, encapsulating it into a QuantCoin$^{\text{TM}}$ token, which is given to Bob,
       $token = SHA256(encryptedInput$
       $+ encryptedOutput + proof)$
       $Bob.receive(token)$    (8)
  10. The $reward$ held in escrow is released to Bob,
       $escrow.release(Bob)$    (9)
```

TABLE I: Sketch for how a quantum computation-backed cryptocurrency might be implemented. We have abstracted away the underlying Blockchain protocol, interfacing with it using a high-level API, since Blockchain technology is highly liable to evolve. We similarly call upon verification subroutines using a high-level implementation-independent API.

ought to be paying something for the service.

Note that we observe an expansion in the money supply with each successfully executed and verified computation – one additional unit of QuantCoins$^{\text{TM}}$ is mined for every unit of computation implemented. Unlike Bitcoin, there is no inherent theoretical upper limit on the number of coins that can exist. However the QuantCoin$^{\text{TM}}$ money supply will be limited for the practical reason that mining each one is associated with a monetary transaction between Alice and Bob, and Alice will eventually run out of assets to exchange for computations.

What relationships characterise the value of QuantCoins$^{\text{TM}}$? First, in a perfectly efficient market (Sec. ??) we have,

$$\begin{aligned}&(\text{Dollar value of a QuantCoin}^{\text{TM}})\\&+(\text{Dollar value of reward paid by Alice for execution})\\&=(\text{Dollar value of cost of execution for Bob}).\end{aligned}\tag{10}$$

Alternately, rather than Alice paying Bob's reward in dollars, she might pay for them in already-existing QuantCoins$^{\text{TM}}$. Suppose Alice pays λ QuantCoins$^{\text{TM}}$ as Bob's reward. Then Eq. (10) reduces to,

$$\begin{aligned}&(\text{Dollar value of a QuantCoin}^{\text{TM}})\\&=\frac{1}{\lambda+1}(\text{Dollar value of cost of execution for Bob}),\end{aligned}\tag{11}$$

providing us with a simple financial model relating the market price of QuantCoins$^{\text{TM}}$ and the monetary cost of execution of computations.

Note that with exception to the scenario where Alice buys into QuantCoins$^{\text{TM}}$ using dollar currency (or any non-electronic asset that cannot be committed to the Blockchain), the entire protocol is self-enforcing via Blockchain transactions. Bob doesn't get paid his newly earned and freshly printed QuantCoin$^{\text{TM}}$ until the verification of the computation has completed and the proof committed to the Blockchain. He therefore cannot get paid until he has executed the computation he promised to, and proven to the network that he actually did.

The main security risk is that of Bob taking Alice's dollars and running, upon receiving the upfront reward in dollars, which necessarily don't reside on the Blockchain since they are not crypto-assets. This could be addressed by introducing trusted third-party escrow agents into the protocol.

However, if the upfront payment were being made in pre-existing QuantCoins$^{\text{TM}}$, the Blockchain might be programmed to not release the reward until completion of the final verification stage of the protocol – effectively an escrow programmed directly into the Blockchain for self-execution.

2. *Futures market model*

As described above, the cryptocurrency is effectively backed by the spot market in computation – we exchange currency for the execution of computations *immediately*.

However, one might also envisage more complex cryptocurrencies being backed by the futures market in the licensing of quantum compute-time.

Intuitively, one would expect such a form of cryptocurrency to be sounder than the one backed by the spot market. This is because our spot market-derived coins, once mined are not guaranteed to be convertible to anything of value, including computations. Recall that the execution of the computation takes place immediately when the coin is created.

On the other hand, a QuantCoin$^{\mathrm{TM}}$ backed by a guarantee to access quantum compute-time at a designated point in the future necessarily has value, so long as the demand for compute-time does, and maintains value until the contract matures and converts into compute-time at which point it becomes worthless.