Qogecoin

Aaron S. Mondal neqochan
Jannis M. Fengler qogekun
Jaro N. Eichler usaqirin

Abstract

We present Qogecoin, a fancy new blockchain that can do pretty much everything and will fly to the moon. Qogecoin introduces sensible tokenomics and a community-friendly infrastructure on a battle-tested codebase. In this paper, we identify the need for quantum-resistant signatures in cryptocurrencies. We take a look at current transaction models and show how to break their encryption with quantum computers. Furthermore, we discuss why it is necessary to introduce quantum-resistant signature schemes many years before quantum computers become commercially viable.

1 Chain parameters

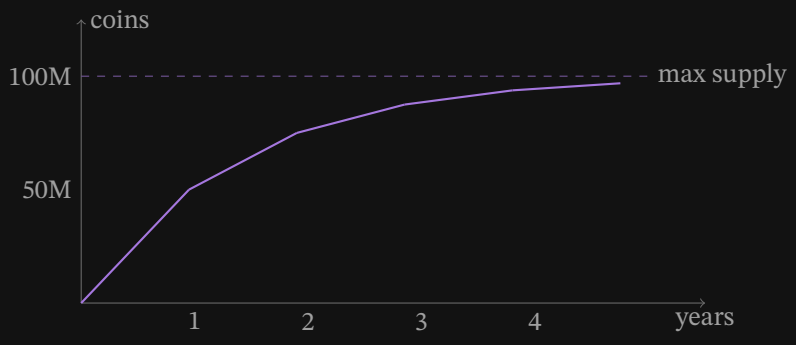
Let us begin with simple things first. Below is a rundown of Qogecoin's blockchain parameters. [4]

Genesis. The genesis block was created on 1 September 2021, 12:40:00 GMT.

Max supply. Maximum supply is capped at 100 million. This is roughly 5x the amount of bitcoin's maximum supply.

Block time. Average block time is 1 minute. To ensure consistency, Qogecoin uses Dark Gravity Wave 3.0 for difficulty adjustment. [5]

Block reward. The initial reward is 100 qogecoin per block. Halving occurs every 500.000 blocks. This implies that the coinbase halves when the circulating supply reaches 50 million, 75 million, 87.5 million, and so on.



Consensus. Proof of work, YescryptR16. [8][9] Mineable on both GPU and CPU.

Compatibility. Qogecoin is fully compatible with existing Bitcoin APIs.

2 Introduction

In the beginning, Jannis went to Aaron and was like, ”BrO LeTs BuIlD QuAnTuMsAfE BlOcKcHaIn” and Aaron was like, ”LeTs JuSt SpIn ThE q”, and Qogecoin was born. And Jannis said, ”MiNeRzZ gO bRrRrRr” and made it mineable. And Aaron said, ”ShIt Is ToO bLoAtEd” and built infrastructure. Then Jannis and Aaron realized that they needed moar quick maths, so they went to Jaro like ”BrO nEeD 2+2 SkIlLz”. And Jaro was like, ”5” and made fancy calculations no one could understand.

At some point the three realized that what they had built was beautiful, so they rebuilt it against the Bitcoin core to benefit from the collective knowledge of those who seemed to know what they were doing.

3 Transaction model

To clearly understand how quantum computers pose a threat to existing blockchains, we need to understand blockchain transactions first. Note that this discussion is specific to Bitcoin derivatives but applies, in other form, to any other blockchain using elliptic curve cryptography.

Addresses

A **private key** is a large randomly generated number. For instance, a base-64 encoded Qogecoin private key looks like

```
XMgizgMqE1VRvq9RZaE4zH8cdmVJdabGwKRzJWXs1ZYiQrZWXXZb.
```

From this private key, we can derive a base-16 encoded **public key**

```
028dddeb2cffebef453d80163ebf36b2c
edc9ebd3857dad4d356b543b2e4233957.
```

This derivation function is usually implemented using elliptic curve cryptography. For ease of use, we apply various hash functions to shorten the public key to a base-58 encoded **address**

```
qN8xXRAoLzRioamfdCw49ng77NQ5a4Gq5b.
```

Transactions

To illustrate transactions, we will restrict ourselves to a heavily simplified model. Our idealized blockchain has only a single coin, and transactions can send this coin from one address to another. Our **transaction** is described by the data structure

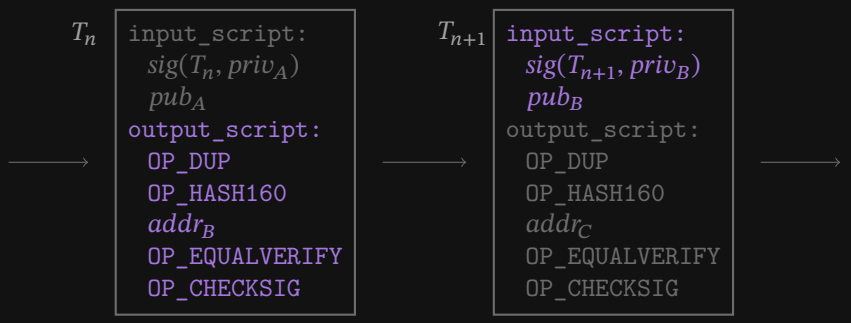
```
input_script:
    <sig>  // Cryptographic signature of the
           // transaction, signed using the
```

```
                // sender's private key.
        <pub_key>  // Sender's public key.
    output_script:
        OP_DUP
        OP_HASH160
        <addr>  // Receiver's address.
        OP_EQUALVERIFY
        OP_CHECKSIG
```

Qogecoin uses the **Script** language in `input_script` and `output_script`. It is the same Turing incomplete language used by Bitcoin. Herein lies a stark contrast to Turing complete languages, such as Solidity, Ethereum's transaction language. [13] Turing completeness has shown to be rather easily exploitable in the past. [3]

The single coin gets minted in a special transaction T_0 that has no input script. This transaction is often called **coinbase transaction**. The address specified in the coinbase transaction can spend the coin by creating a transaction T_1. The receiver of T_1 can now spend the coin by creating a transaction T_2, and so on. In this way, the coin goes on a journey from address to address. This journey describes a chain of transactions $T_0, T_1, \dots, T_n$.

Let us now assume that user B with address $addr_B$, private key $priv_B$ and public key pub_B received the coin from user A via transaction T_n. In order to send it to another user C, user B creates a new transaction T_{n+1} as follows.



We want to ensure that at any time only a single address is able to claim the single coin. This is done using a verification process.

Verification

Assume we have an existing sequence of transactions $T_0, T_1 \dots, T_n$ and a new transaction T_{n+1}. We say that T_{n+1} is **valid** if the concatenation of `input_script` of T_{n+1} and `output_script` of T_n evaluates to `True`. Let us take a closer look at this concatenation.

Commands	Stack
<sig>	<sig>
<pub_key>	<sig> <pub_key>
OP_DUP	<sig> <pub_key> <pub_key>
OP_HASH160	<sig> <pub_key> <hash(pub_key)>

```
<addr>              <sig> <pub_key> <hash(pub_key)> <addr>
OP_EQUALVERIFY      <sig> <pub_key>
OP_CHECKSIG         True
```

Here, `OP_EQUALVERIFY` only removes the top two elements from the stack if `<addr>` is equal to `<hash(pub_key)>`. The `OP_CHECKSIG` command verifies if `<sig>` is a valid signature for T_{n+1}. We require this to only be the case if the private key used to create `<sig>` and the private key used to create `<pub_key>` coincide.

The need for quantum safety

Existing hashing schemes are assumed to be quantum secure. That is, given only `addr` we cannot efficiently derive `pub_key` from it. However, when using elliptic curve cryptography we CAN use a quantum computer to derive the private key given only `pub_key`. We will show how to do this in the remainder of this paper.

As soon as a user issues a transaction, the user's public key is revealed on the blockchain. Hence, any address that ever made a transaction is freely useable by someone with a sufficiently large quantum computer.

Currently, users can keep their funds quantum-safe by spending their entire funds in every transaction. Assume we have 100 coins and want to spend 1 coin. We create a transaction sending 1 coin to our target and 99 coins to a new address that we own and have never used. Then our funds are only insecure for the time it takes the network to verify the transaction. Unfortunately, for users requiring reusable addresses, this is not an option.

Most applications built on top of the basic transaction model use `OP_CHECKSIG` in some form and as such have the same vulnerabilities as basic transactions. Some smart contracts lock funds for several years and reuse addresses frequently. So even if quantum computers are still years away from being commercially viable, it is necessary to update signature schemes a long time in advance.

Qogecoin aims to switch to a quantum-safe signature scheme. Public keys and signatures are stored on the blockchain. Hence, we need to keep these sizes low. Finding quantum-safe signature schemes with reasonable security and small signature/public key sizes is not an easy task and subject of ongoing research. [2]

4 Elliptic curves

Elliptic curves are, unsurprisingly, the central building block of elliptic curve cryptography.

Definition

For a prime number $p > 3$ we denote the **finite field** with p elements by $\mathbb{F}_p$. Its elements are represented by integers where addition and multiplication are calculated modulo p. For instance, if $p = 5$, the elements in $\mathbb{F}_5$ can be represented by $0, 1, 2, 3$ and 4. Here, $3 + 4 = 2$ because 7 modulo 5 is 2.

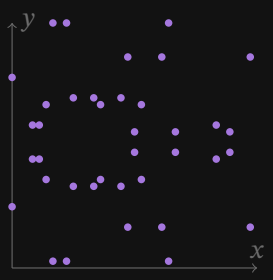
Consider the equation

$$y^2 = x^3 + ax + b$$

with parameters $a, b \in \mathbb{F}_p$. An **elliptic curve** E is the set of solutions $(x, y) \in \mathbb{F}_p^2$ of this equation, together with an element ∞.[1] We can imagine each element of E as a point on a grid. It turns out that E has additional structure. For two points $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ in E we can draw a line L through P and Q, parameterized by

$$(x_Q - x_P, y_Q - y_P) \cdot s + (x_P, y_P).$$

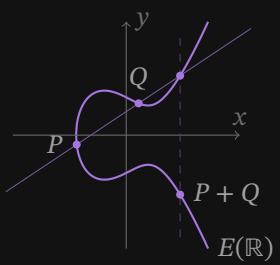
This line intersects E in exactly one more point $R = (x_R, y_R) \in E$. There are various special cases. For instance, the point ∞ is defined as the third point of intersection of a line through (x, y) and $(x, -y)$.



Elliptic curve $y^2 = x^3 + 7$ without ∞ over $\mathbb{F}_{37}$.

We get an abelian group structure on E by defining the inverse and the addition as

$$-R = (x_R, -y_R)$$
$$P + Q = -R.$$



An elliptic curve over the real numbers.

It follows that ∞ is the neutral element and we denote it by 0. For example, let us calculate $P + Q$ for $P = (6, 1)$ and $Q = (8, 1)$ for the curve $y^2 = x^3 + 7$ over $\mathbb{F}_{37}$. The line through P and Q is given by $(2, 0) \cdot s + (6, 1)$. It further intersects $E(\mathbb{F}_{37})$ in $(23, 1)$. Hence, $P + Q = (23, -1)$.

Elliptic curve discrete logarithm problem

Back to signatures for blockchain transactions. Let $p > 3$ be a prime number, E an elliptic curve over $\mathbb{F}_p$, and $P \in E$ a point. For an integer m we can calculate

$$mP = \sum_{i=1}^{m} P.$$

Let us assume that P is of prime order r. This means that r is a prime number and that r is the smallest natural number such that $rP = 0$ in E.

A **private key** is a randomly chosen m. The **public key** to a private key m is the point mP. In practice, points on E are represented by their x-coordinate together with the sign of the y-coordinate. The most prominent example is given by the **secp256k1** standard. [11]

```
p =    FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
       FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F
a =    00000000 00000000 00000000 00000000
       00000000 00000000 00000000 00000000
b =    00000000 00000000 00000000 00000000
       00000000 00000000 00000000 00000007
P = 02 79BE667E F9DCBBAC 55A06295 CE870B07
       029BFCDB 2DCE28D9 59F2815B 16F81798
```

[1] Technically, these are only the $\mathbb{F}_p$ rational points of an elliptic curve but we do not want to delve into the depths of schemes and algebraic geometry.

After issuing transactions, the public keys are visible to anyone. The secureness of elliptic curve cryptography is based on the fact that reverse computing the private key from a public key is a very hard problem. Naively forward computing public keys is nearly impossible because, in practice, the number of potential values for m is gigantic. Deducing m from mP seems simple, but classical computers would essentially take forever for this calculation. [10]

5 Quantum computing

We will now see how quantum computers can be viewed as extensions of classical computers.

Classical computers

In a simplified manner, we may describe a **classical computer** as a tuple

$$(\mathcal{S}, \mathcal{O}).$$

Here, $\mathcal{S}$ is an arbitrary set whose elements we call **states**. The set $\mathcal{O}$ is an arbitrary set of maps $\mathcal{S} \to \mathcal{S}$ such that for $O, O' \in \mathcal{O}$ their composition $O \circ O'$ is again in $\mathcal{O}$. We call the elements of $\mathcal{O}$ **operations**.

In the real world, a processing unit is usually only able to execute very simple operations. In our generalized setting, we can describe this property by considering a subset $\mathcal{G}$ of $\mathcal{O}$. We require every element of $\mathcal{O}$ to be the composition of elements of $\mathcal{G}$. We use the term **gates** for elements of $\mathcal{G}$.

We can now view any program as a series of gates $G_1, \dots, G_n$, sequentially executed to transition an initial state s to a final state t.

Bits

In a typical example, the states of a classical computer are described by n **bits** of memory. For instance, we could consider

$$\mathcal{S} = \{0, 1\}^n$$
$$\mathcal{G} = \{0_i, \texttt{NOT}_i, \texttt{AND}_{ijk} \mid i, j, k = 1, \dots, n\}$$

The 0_i operations set the i-th bit to zero and the $\texttt{NOT}_i$ operations flip the i-th bit. The $\texttt{AND}_{ijk}$ operations compute AND for the i-th and j-th bit and write the result to the k-th bit. For instance, when $n = 8$, we can describe multiplication by 2 as the sequence

Quantum computers

We describe a **quantum computer** as a tuple

$$(\mathcal{H}, \mathcal{U}).$$

Here, $\mathcal{H}$ is a complex Hilbert space. This is a complete complex vector space with a Hermitian inner product

$$\begin{aligned} \langle \cdot \mid \cdot \rangle : \mathcal{H} \times \mathcal{H} &\to \mathbb{C} \\ (v, w) &\mapsto \langle v \mid w \rangle . \end{aligned}$$

We call the vectors of norm 1 **quantum states**.[2] We use the Dirac notation $|v\rangle \in \mathcal{H}$ to denote quantum states.[3] The set $\mathcal{U}$ is an arbitrary subgroup of the group of unitary operators on $\mathcal{H}$. This means that elements of $\mathcal{U}$ are linear maps $\mathcal{H} \to \mathcal{H}$ preserving the norm. We call the elements of $\mathcal{U}$ **quantum operations**. Our construction ensures that quantum operations always map quantum states to quantum states. Let $\mathcal{B}$ be a choice of a generating set of the group $\mathcal{U}$. Then we call the elements of $\mathcal{B}$ **quantum gates**.

From classical computers to quantum computers

A classical computer $(\mathcal{S}, \mathcal{O})$ is called **reversible** if $\mathcal{O}$ consists only of bijective maps. In this case, operations are simply permutations on $\mathcal{S}$. We may always enlarge $\mathcal{S}$ to make a classical computer reversible. [7]

Let $(\mathcal{S}, \mathcal{O})$ be a reversible classical computer with finite $\mathcal{S}$. We construct a Hilbert space from $\mathcal{S}$ by taking the $\mathbb{C}$-span

$$\mathcal{H} = \mathbb{C}^{\mathcal{S}}.$$

The elements of $\mathcal{S}$ are a natural choice for a basis of $\mathcal{H}$. We call this basis the **computational basis**. We are left with defining the Hermitian inner product. The obvious choice is the standard inner product with respect to the computational basis

$$\langle v \mid w \rangle = \sum_{s \in \mathcal{S}} v_s \cdot \overline{w_s}.$$

Since we required $(\mathcal{S}, \mathcal{O})$ to be reversible, classical operations define permutations on the computational basis $\mathcal{S}$. Hence, classical operations uniquely extend to unitary operators on $\mathcal{H}$. We now choose $\mathcal{U}$ to be any subgroup of unitary operators containing $\mathcal{O}$ in the above sense.

Measurements

Let $(\mathcal{H}, \mathcal{U})$ be a quantum computer constructed from a classical computer $(\mathcal{S}, \mathcal{O})$ as previously discussed. Consider a quantum state

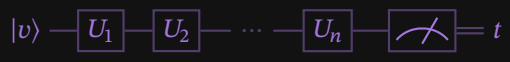
$$|v\rangle = \sum_{s \in \mathcal{S}} v_s \cdot |s\rangle .$$

[2] Usually called pure quantum states.

[3] We say "bra w" for $\langle w| \in \mathcal{H}^*$ and "ket v" for $|v\rangle \in \mathcal{H}$. This is convenient because we can identify $\langle w|(|v\rangle) = \langle w \mid v\rangle$ as "bra ket".

We can observe a classical state after **measuring** the quantum state $|v\rangle$ with respect to the computational basis. We will observe a specific classical state s with probability $|v_s|^2$. After measuring, the quantum computer will be in the quantum state $|s\rangle$. So if we were to measure it again, we would observe the same classical state s again.

We can view a program on a quantum computer as a sequence of quantum gates $U_1, \dots, U_n$ and a measuring. The sequence operates on a quantum state $|v\rangle$ and returns a classical state t after measurement. Note that t is not deterministic, so in practice, one would repeat the computation several times to increase the confidence in the result.

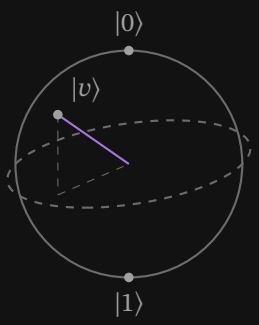


In practice, the set $\mathcal{S}$ is usually given as a product $\mathcal{S} = S_1 \times S_2$. The corresponding Hilbert space then decomposes as

$$\mathcal{H} = \mathbb{C}^{S_1} \otimes \mathbb{C}^{S_2}.$$

We call these factors first and second register. Similarly, we can describe quantum computers with n registers.

Qubits



Bloch sphere representation of qubit state space.

Let us now explicitly transition from a single bit classical computer with $\mathcal{S} = \{0, 1\}$ to a quantum computer with a single **qubit**. The Hilbert space corresponding to $\mathcal{S}$ is the 2-dimensional space $\mathcal{H} = \mathbb{C}^{\{0,1\}}$. The quantum states are

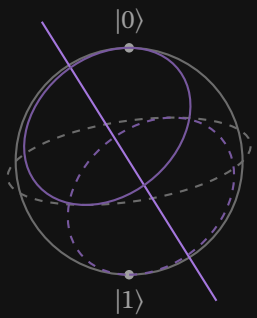
$$|v\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

where $\alpha, \beta \in \mathbb{C}$ such that $\alpha^2 + \beta^2 = 1$. The space of all quantum states is a complex 1-dimensional hyperplane in $\mathcal{H}$. Measuring a qubit in the quantum state $\alpha\,|0\rangle + \beta\,|1\rangle$ results in the classical state 0 with probability $|\alpha|^2$ and in 1 with probability $|\beta|^2$.

Unitary operators on $\mathcal{H}$ are rotations of the qubit state space. For instance, the classical gate `NOT` from our previous example induces a quantum gate X, represented by the reflection matrix

$$\mathrm{X}(\alpha\,|0\rangle + \beta\,|1\rangle) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix} = \beta\,|0\rangle + \alpha\,|1\rangle\,.$$

Quantum gates on Qubits



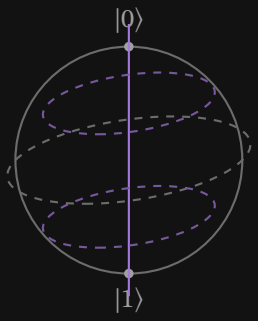
Hadamard rotation by π around the diagonal.

The **Hadamard gate** is the unitary operator acting on a single qubit, represented by the matrix

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

The Hadamard gate operates on the computational basis as

$$\mathrm{H}(|0\rangle) = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \qquad \mathrm{H}(|1\rangle) = \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

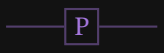


Phase shift by φ around the polar axis.

Measuring after applying the Hadamard gate on $|0\rangle$ will result in the state 0 with probability 1/2 and in the state 1 with probability 1/2. Measuring after applying two Hadamard gates on $|0\rangle$ will certainly result in the state 0.

For $\varphi \in \mathbb{R}$ the **phase shift gate** is the unitary operator acting on 1 qubit, represented by the matrix

$$\mathrm{P}_\varphi = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{bmatrix}.$$



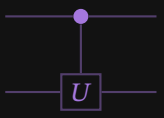
A phase shift alone does not influence a measurement, but as soon as a phase shift interacts with anything other than a phase shift it will influence the measuring.

For many computations, we require multiple qubits to interact with each other. One such interaction can be constructed by using one qubit to control a gate on another qubit. Given a gate U acting on one qubit, we define the **controlled-U gate** on two qubits as the unitary operator represented by the 4×4 matrix

$$CU = \begin{bmatrix} \mathbb{1}_2 & 0 \\ 0 & U \end{bmatrix}.$$

For instance, a **controlled phase shift gate** on 2 qubits is represented by

$$\mathrm{CP}_\varphi = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{2\pi i\varphi} \end{bmatrix}.$$



In circuit notation, we will denote the controlling qubit by a dot.

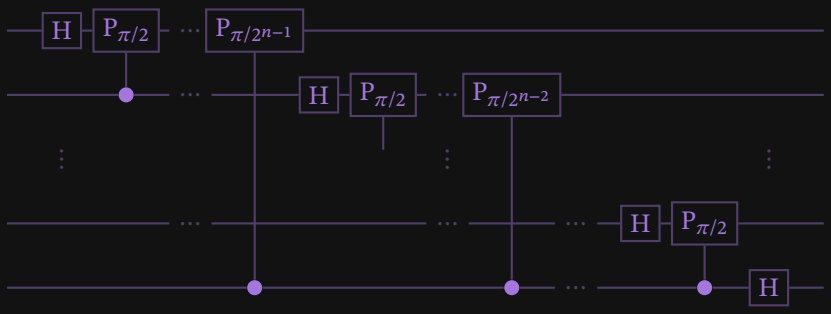
Quantum Fourier transform

Let $\mathbb{Z}/r\mathbb{Z}$ be the cyclic group with r elements. The **quantum Fourier transform**, or in short **QFT**, is the unitary operator acting on $\mathbb{C}^{\mathbb{Z}/r\mathbb{Z}}$, defined as

$$\mathrm{QFT}\left(|g\rangle\right) = \frac{1}{\sqrt{r}} \sum_{x \in \mathbb{Z}/r\mathbb{Z}} e^{2\pi i g x/r} \, |x\rangle \, .$$

Computing with n bits is the same as computing in $\mathbb{Z}/2^n\mathbb{Z}$ by identifying each number with its binary representation. This means that we can construct a quantum Fourier transform in this case. For $r = 2^n$, the quantum Fourier transform can be

implemented using Hadamard gates and controlled phase shift gates. [6]



QFT with reversed output qubits.

6 Shor's algorithm on elliptic curves

Let $p > 3$ be a prime number. Let E be an elliptic curve over $\mathbb{F}_p$ and $P \in E$ a base point of prime order r. We want to compute the private key $m \in \mathbb{Z}/r\mathbb{Z}$ given the public key $Q = mP$ in E.

Algorithm

For simplicity, we now work with an abstract quantum computer. Consider an invertible classical computer on the set

$$S = \mathbb{Z}/r\mathbb{Z} \times \mathbb{Z}/r\mathbb{Z} \times E.$$

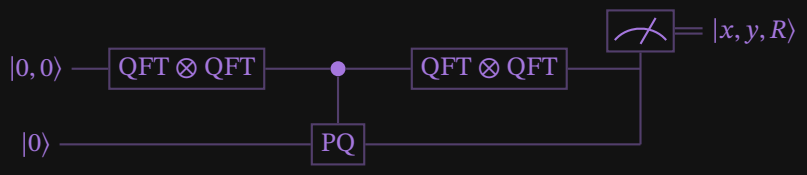
As described earlier, we can extend this invertible classical computer to a quantum computer with Hilbert space

$$\mathcal{H} = \mathbb{C}^S = \mathbb{C}^{\mathbb{Z}/r\mathbb{Z}} \otimes \mathbb{C}^{\mathbb{Z}/r\mathbb{Z}} \otimes \mathbb{C}^E.$$

We denote its quantum states by $|k, l\rangle \otimes |R\rangle$. For our computation, we will need two quantum gates. Consider the classical operation

$$\mathtt{PQ}\,(k, l, R) = (k, l, kP + lQ + R).$$

It induces a quantum gate that we denote by PQ. We can interpret PQ as the addition on E, controlled by the quantum state $|k, l\rangle$. The second quantum gate we need is a QFT on $\mathbb{C}^{\mathbb{Z}/r\mathbb{Z}}$. Using these quantum gates we can build a quantum circuit as depicted below.



Let us walk through this circuit step by step. We initialize the states to

$$|0,0\rangle \otimes |0\rangle$$

where the first register holds the zeros in $\mathbb{Z}/r\mathbb{Z}$ and the second register holds the zero element of the elliptic curve. Now we apply QFT $\otimes$ QFT on the first register. Since $e^0 = 1$, we will be left in the state

$$\begin{aligned}\text{QFT} \otimes \text{QFT}(|0,0\rangle) \otimes |0\rangle &= \left(\frac{1}{\sqrt{r}} \sum_{k \in \mathbb{Z}/r\mathbb{Z}} |k\rangle\right) \otimes \left(\frac{1}{\sqrt{r}} \sum_{l \in \mathbb{Z}/r\mathbb{Z}} |l\rangle\right) \otimes |0\rangle \\ &= \frac{1}{r} \sum_{k,l \in \mathbb{Z}/r\mathbb{Z}} |k,l\rangle \otimes |0\rangle .\end{aligned}$$

Now we apply PQ on the second register, leaving us in the state

$$\frac{1}{r} \sum_{k,l \in \mathbb{Z}/r\mathbb{Z}} |k,l\rangle \otimes |kP + lQ\rangle .$$

Finally, we apply QFT $\otimes$ QFT again and our final quantum state is

$$\frac{1}{r^2} \sum_{x,y,k,l \in \mathbb{Z}/r\mathbb{Z}} e^{2\pi i(xk+yl)/r} |x,y\rangle \otimes |kP + lQ\rangle .$$

Measuring this state will result in a tuple (a, b, Z). We will now show that any non-zero (a, b) will give us the private key as

$$m = \frac{b}{a}$$

with division modulo r.

Correctness

Let (a, b, Z) be the outcome of the measurement of the final state, and let (x, y, R) be a specific tuple with $x, y \in \mathbb{Z}/r\mathbb{Z}$ and $R \in E$ a curve point. We will show that the probability of (a, b, Z) and (x, y, R) coinciding is

$$\begin{cases} \frac{1}{r^2}, & xm - y = 0 \mod r \\ 0, & xm - y \neq 0 \mod r. \end{cases}$$

Furthermore, with probability $(r-1)/r$ the integer a is non-zero modulo r. In this case $m = b/a$ modulo r.

The probability of measuring the final state to a specific (x, y, R) is given by the squared norm of the coefficient of $|x, y\rangle \otimes |R\rangle$ in the final state. Let us take a closer look at this coefficient

$$\frac{1}{r^2} \sum_{\substack{k,l \in \mathbb{Z}/r\mathbb{Z} \\ kP+lQ=R}} e^{2\pi i(xk+yl)/r}.$$

We assume that there is a pair (k_R, l_R) with $k_R P + l_R Q = R$. Otherwise, the sum is empty. Any k, l with $kP + lQ = R$ can then be written as $(k, l) = (k_R, l_R) + t \cdot (m, -1)$

with $t = 0, \dots, r-1$.[4] The coefficient can then be written as

$$\frac{1}{r^2} \sum_{t=0}^{r-1} e^{2\pi i (x(k_R + tm) + y(l_R - t))/r}$$
$$= \frac{1}{r^2} e^{2\pi i (xk_R + yl_R)/r} \sum_{t=0}^{r-1} e^{2\pi i t(xm-y)/r}.$$

Phase shifts at the end of the circuit do not affect the measurement. In other words $|e^{i\varphi}| = 1$. Hence, taking the squared norm will give us

$$\left| \frac{1}{r^2} \sum_{t=0}^{r-1} e^{2\pi i t(xm-y)/r} \right|^2.$$

Recall $\sum_{i=0}^{n-1} s^i = (1 - s^n)/(1 - s)$ for any complex number $s \neq 1$. With $s = e^{2\pi i (xm-y)/r}$, the condition $s \neq 1$ translates to $xm - y \neq 0$ modulo r. We get

$$\begin{cases} \left| \frac{1}{r^2} \sum_{t=0}^{r-1} e^{2\pi i t(xm-y)/r} \right|^2 = \frac{1}{r^2}, & xm - y = 0 \mod r \\ \left| \frac{1}{r^2} \cdot \frac{1 - e^{2\pi i (xm-y)}}{1 - e^{2\pi i (xm-y)/r}} \right|^2 = 0, & xm - y \neq 0 \mod r. \end{cases}$$

We now know with certainty that the tuple (a, b, Z) satisfies $am - b = 0$ modulo r and that Z is a multiple of P. If $a = 0$, then $b = 0$ and there are r possible choices for Z since P is of order r. Hence, the probability of $a \neq 0$ equals $1 - r \cdot (1/r^2) = (r-1)/r$.

7 Conclusion

Whew! That was a lot to take in! We learned how blockchain transactions work, how quantum computers work and how quantum computers can break current elliptic curve cryptography. We discussed why quantum safety is already relevant even if quantum computers are quite a few years away from commercial viability.

Going forward, we will work hard on finding and implementing signature schemes that are suited to blockchain transactions. Additionally, we will stay upstream with Bitcoin core so that we can keep adding all the nice features one would expect from a modern blockchain. Add in various modernizations from other areas of the software industry and we are on a good path to creating a blockchain that is lean, mean, and ready for the era of quantum supremacy.

References

[1] Google Quantum AI and Doublespeak Games. The qubit game, 2022. `https://quantumai.google/education/thequbitgame`, accessed May 10, 2022.

[4] The stabilizer of 0 is a proper subgroup of a group of order r^2, hence of prime order r and $(m, -1)$ is a generator.

[2] Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, John Kelsey, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, et al. Status report on the second round of the nist post-quantum cryptography standardization process. *US Department of Commerce, NIST*, 2020.

[3] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. A survey of attacks on ethereum smart contracts (sok). In *International conference on principles of security and trust*, pages 164–186. Springer, 2017.

[4] The Qogecoin Authors. qogecoin, 2021. `https://github.com/qogecoin/qogecoin`, accessed May 10, 2022.

[5] Evan Duffield. Dark gravity wave 3, 2014. `https://docs.dash.org/en/stable/introduction/features.html`, accessed May 10, 2022.

[6] Phillip Kaye, Raymond Laflamme, and Michele Mosca. *An introduction to quantum computing*. OUP Oxford, 2006.

[7] Alexei Y. Kitaev. Quantum measurements and the abelian stabilizer problem. *Electronic Colloquium on Computational Complexity*, 3, 1996.

[8] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, 2008.

[9] Openwall. yescrypt, 2018. `https://www.openwall.com/yescrypt/`, accessed May 10, 2022.

[10] John M. Pollard. Kangaroos, monopoly and discrete logarithms. *Journal of cryptology*, 13(4):437–447, 2000.

[11] SECG. Sec 2: Recommended elliptic curve domain parameters. `https://www.secg.org/sec2-v2.pdf`, accessed May 10, 2022.

[12] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, Oct 1997.

[13] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.