Shor's Algorithm

Andrei Laurentiu Ciupan

CS 227, Spring 2018 final project

Abstract. We will prove Shor's algorithm for finding the period of the exponential function in a finite group, which can be used to break RSA encryption. We will introduce the RSA encryption scheme and walk through the proof of Shor's algorithm, introducing the necessary quantum computing tools.

1. Introduction

The RSA algorithm is a public-key encryption scheme. Assume there are two communicating parties, Alice and Bob which communicate over a public channel that can be intercepted by a third party, Eve: Bob wishes to send to Alice a message m. A core version of the RSA scheme is described as follows:

(1) Alice chooses two large primes p, q and a positive integer c such that $\gcd(c, (p-1)(q-1)) = 1$. [1] Alice sends $(N = pq, c)$ through the channel.
(2) Bob sends $h = m^c \pmod N$.

In the core version described above, we assume that $m < N$ and $\gcd(m, N) = 1$. The complementary case $m \geq N$ or $\gcd(m, N) > 1$, and other variants of RSA can be developed from the main scheme described above.

The RSA algorithm is a valid encryption scheme. This algorithm's security comes from the computational difficulty of inferring m after seeing N, c and $h = m^c$.

Moreover, Alice can decrypt the message above: note that Alice has access to p, q and c and can therefore compute $d = c^{-1} \pmod{(p-1)(q-1)}$. Since $m^{(p-1)(q-1)} \equiv 1 \pmod{pq}$, we have that $h^d \equiv m^{cd} \equiv m \pmod N$, so the message m can be found by computing h^d.

Shor's algorithm presents a method for computing m after seeing N, c and $h = m^c$. Consider the function $f_h : \mathbb{Z}_N \mapsto \mathbb{Z}_N$ defined by $f_h(x) = h^x \pmod N$. Assume that there exists an algorithm $S(N, h)$ which computes the period of function f_h. Then consider the following algorithm with inputs N, c, h:

(1) Compute $r = S(N, h)$.
(2) Compute $d' \equiv c^{-1} \pmod r$.

[1] The choices of p, q, c can be made, for example, through a probabilistic algorithm: repeatedly draw random p, q until the two numbers obtained are prime. Then repeatedly draw random c until the obtained value is coprime with $(p-1)(q-1)$. The prime number theorem guarantees that this algorithm will take linear time in expectation.

(3) Compute $m' = h^{d'} \pmod N$.

We claim that the final output m' equals m. Here is why:

First of all, the period r of f_h is the order of h modulo N, the smallest v such that $h^v \equiv 1 \pmod N$. Let r' be the order of m modulo N. Then $m^{r'} \equiv 1 \pmod N$ and r' is minimal with this property. Since $h^r \equiv 1 \pmod N$ we have $m^{cr} \equiv 1 \pmod N$, therefore $m^{\gcd(r',cr)} \equiv 1 \pmod N$. [2] But since r' is the order of m modulo N and $m^{(p-1)(q-1)} \equiv 1 \pmod N$, we must have $r'|(p-1)(q-1)$. However, we are guaranteed that $\gcd(c,(p-1)(q-1)) = 1$, so $\gcd(r', cr) = \gcd(r', r)$, so $m^{\gcd(r,r')} \equiv 1 \pmod N$. However, r' is minimal with this property, so we must have $r'|r$. Then since $d' \equiv c^{-1} \pmod r$, we obtain that $d' \equiv c^{-1} \pmod{r'}$ as well, so $d'c = r'q + 1$, for some integer q, so $h^{d'} \equiv m^{cd'} \equiv m^{r'q+1} \equiv m \pmod N$, as desired. □

Therefore, in order to break the RSA encryption scheme we just need to compute the period $S(N, h)$ of the function $f_h(x) = h^x \pmod N$. Further note that steps 2 and 3 only require logarithmic time as a function of N, so they require polynomial time in the number of bits of N.

We have introduced the RSA algorithm and introduced a potential method for breaking it in polynomial time (in the number of bits of N). We are left to show that the first step of our method, finding the period, can be completed in polynomial time as well. This will be the goal of the next sections, and we will require quantum computing methods in order to achieve this.

2. Quantum Computing: Tools

This section will introduce the theoretical tools at our disposal with quantum computing. We will limit ourselves to the theoretical world and accurately describe what we are able to do with a quantum computer. We will not discuss the potential hardware implementations of these methods in practice.

2.1. The Qbits. Qbits are the quantum-computing equivalent of classical bits. While a classical bit can be described by its value, which is 0 or 1, any 1-dimensional qbit is described by its state $q = \alpha_0|0\rangle + \alpha_1|1\rangle$, where α_0 and α_1 are complex numbers such that $|\alpha_0|^2 + |\alpha_1|^2 = 1$. The symbols $|0\rangle$ and $|1\rangle$ can be interpreted as basis elements in the vector space spanned by them. A harmless remark is to note that we can interpret this representation of qbits as a generalization of the classical representation of bits, if we restrict the values of (a_0, a_1) to $(1, 0)$ or $(0, 1)$.

In 1 dimension, the basis elements of the representation of a qbit are the two states $|0\rangle$ and $|1\rangle$. In n dimensions, a qbit is described by its state $q = \sum_{0\le x<2^n} \alpha_x|x\rangle_n$, where:

(1) α_x are complex numbers such that $\sum_{0\le x<2^n} |\alpha_x|^2 = 1$
(2) The states $\{|x\rangle_n\}$ can be interpreted as basis vectors, where the subscript makes clear that they are n-dimensional. We can represent these basis

[2]Here we use the result that if $m^x \equiv 1 \pmod N$ and $m^{x'} \equiv 1 \pmod N$ then $m^{\gcd(x,x')} \equiv 1 \pmod N$. This result follows from the fact that the gcd of any two numbers can be written as a linear integer combination of the two.

vectors by their binary representation as well, and, for instance identify $|1\rangle|0\rangle|1\rangle$ with the three-dimensional qbit representation $|5\rangle_3$, or $|101\rangle_3$

In general, when I don't specify the dimension of a qbit, I refer to a one-dimensional one. Note that multiple qbits can be combined to produce a higher-dimensional qbit. We can inductively define this process through the tensor product: If $|\phi\rangle = a_0|0\rangle + a_1|1\rangle$ and $|\psi\rangle = b_0|0\rangle + b_1|1\rangle$ then

$$|\Psi\rangle = |\phi\rangle \otimes |\psi\rangle = (a_0|0\rangle + a_1|1\rangle) \otimes (b_0|0\rangle + b_1|1\rangle)$$

$$= a_0b_0|0\rangle|0\rangle + a_0b_1|0\rangle|1\rangle + a_1b_0|1\rangle|0\rangle + a_1b_1|1\rangle|1\rangle$$

$$= a_0b_0|00\rangle_2 + a_0b_1|01\rangle_2 + a_1b_0|10\rangle_2 + a_1b_1|11\rangle_2.$$

This process is extended inductively to define a tensor product between an n-dimensional qbit and a m-dimensional qbit as a $(m+n)$ -dimensional qbit.

2.2. Operations on qbits. An operation on an n-dimensional qbit state is a linear transformation from the space of n-dimensional qbits to itself. Necessarily, such a transformation is defined by its action on the 2^n basis vectors $\{|x\rangle_n\}_{0\leq x<2^n}$. We only deal with unitary operations, i.e. operations U for which the dot product between $U|x\rangle_n$ and $U|x'\rangle_n$ equals 1 if and only if $x = x'$, and zero otherwise, where $|x\rangle_n, |x'\rangle_n$ are any basis vectors. For ease of notation we express the dot product between states $|x\rangle_n$ and $|x'\rangle_n$ by $\langle x|x'\rangle$. The dot product here is the standard dot product in a vector space over complex numbers, defined by the basis rule $\langle a_0x|b_0x'\rangle = \overline{a_0} \cdot b_0\langle x|x'\rangle$, where x, x' are n-dimensional basis vectors and their dot product is defined as 1 if and only if $x = x'$, and zero otherwise.

We can sumarize the previous paragraph as follows: Any operator over the space of n- dimensional qbit states is characterized by the $2^n \times 2^n$ matrix U satisfying $U \cdot U^* = I_{2^n}$, where U^* is the conjugate transpose of U. In this paper we will use the following operations:

(1) **1**. This is the identity operator defined on the space of one-dimensional qbits by the identity matrix $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
(2) H. This is called the Hadamard operator and it's defined on the space of one-dimensional qbits by the matrix $\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$. By construction, $H|0\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$
(3) U_{FT}. This is called the quantum Fourier transform and it is defined on the space of n-dimensional qbits by $U_{FT}|x\rangle_n = \frac{1}{2^{n/2}} \sum_{0\leq y<2^n} e^{\frac{2\pi ixy}{2^n}}|y\rangle_n$. The quantum Fourier transform is indeed a unitary transformation: consider two n-dimensional basis qbits x and x'. Then the dot product between $H|x\rangle$ and $H|x'\rangle$ is written as $\langle Hx|Hx'\rangle$ and equals

$$\left\langle \frac{1}{2^{n/2}} \sum_{0\leq y<2^n} e^{\frac{2\pi ixy}{2^n}} y \middle| \frac{1}{2^{n/2}} \sum_{0\leq y<2^n} e^{\frac{2\pi ix'y}{2^n}} y \right\rangle$$

$$= \langle \sum_{0\leq y<2^n} a_y y| \sum_{0\leq y<2^n} b_y y\rangle = \sum_{0\leq y<2^n} \overline{a_y} \cdot b_y,$$

which in our case is rewritten as $\frac{1}{2^n}\sum_{0\le y<2^n} z^y$, where $z = e^{\frac{2\pi i(x'-x)}{2^n}}$, which is a 2^n-th root of unity equalling 1 if and only if $x = x'$. If $x = x'$ then $z = 1$ and the previous value trivially equals 1. If $z \neq 1$ then $\frac{1}{2^n}\sum_{0\le y<2^n} z^y = \frac{1-z^{2^n}}{1-z} = 0$. So indeed this transformation is unitary. $\square$.

(4) The action of a general function $f : \{0, 1, \ldots, 2^n - 1\} \mapsto \{0, 1, \ldots, 2^m - 1\}$ is defined by the unitary transformation $U_f\left(|x\rangle_n|y\rangle_m\right) = |x\rangle_n|y \oplus f(x)\rangle_m$. We can refer to the first n bits as the input register and the last m bits as the output register.

Note that the last two functions are defined on multiple-dimensional qbit states, while the first two functions are defined on 1-qbit states. We can generalize their action on multiple-dimensional qbit states by inductively defining the tensor product of two one-dimensional unitary transformations. For one-dimensional unitary transformations A and B, the two-dimensional unitary transformation $C = A \otimes B$ is defined by $C|x_0x_1\rangle_2 = (A|x_0\rangle) \otimes (B|x_1\rangle)$.

The rule above inductively defines the tensor product between any unitary transformations. Going back to the definition of the Hadamard gate, notice that its n-th power tensor satisfies $H^{\otimes n}|0\rangle_n = \frac{1}{2^{n/2}}\sum_{0\le x<2^n}|x\rangle_n$. Also notice that the transformation $X = H^{\otimes n} \otimes \mathbf{1}^{\otimes m}$ which applies the n-th power tensor Hadamard gate to the first n bits satisfies $X|0\rangle_{n+m} = \frac{1}{2^{n/2}}\sum_{0\le x<2^n}|x\rangle_n|0\rangle_m$. This relation will be useful in a later section.

It's crucial, and we simply state the result here, that the tensored value of the Hadamard, identity and quantum Fourier transform gates can be applied in polynomial time, as a function of n, to an n-dimensional qbit state.

2.3. Measurements. There are two operations we are allowed to do with an n-dimensional qbit state: apply a unitary transform to it, or measure a subset of its bits. We now define what a measurement does.

First consider a one-dimensional state $q = a_0|0\rangle + a_1|1\rangle$. The measurement function M is a probabilistic function, such that $M|q\rangle = |0\rangle$ with probability $|a_0|^2$ and $M|q\rangle = |1\rangle$ with probability $|a_1|^2$. Once a qbit q is measured, its value becomes $M|q\rangle$, and the original state of q cannot be recovered.

One can perform a measurement of a subset of the bits of an n-dimensional qbit state by measuring the corresponding bit positions one at a time and inductively applying the generalization of the above rule: for an n-dimensional qbit state $q = \sum_{0\le x<2^n} a_x|x\rangle_n$, measuring the i-th bit produces one of the two states:

(1) $\sum_{x\in S_0} a_x|x\rangle_n$ with probability $\sum_{x\in S_0}|a_x|^2$

(2) $\sum_{x\in S_1} a_x|x\rangle_n$ with probability $\sum_{x\in S_1}|a_x|^2$, where S_0 is the set of n-dimensional qbit basis states whose i-th component is 0m and S_1 is the set of n-dimensional qbit basis states whose i-th component is 1.

Remember, as before, that measuring a state ireversibly produces another state.

3. Shor's Algorithm

Having defined the relevant tools and rules at our disposal in quantum computing, we are ready to construct the period-finding algorithm $S(N, h)$ described in the first section. Recall that we are given N and h and want to find the period of the function $f(x) = h^x \pmod N$. Assume that N can be expressed in binary with n_0 bits. The following is Shor's algorithm:

(1) Manually calculate $f(x)$ for a small number of x values[3]. If the period is found, stop. Otherwise move on to the next step.
(2) Consider $n = 2 \cdot n_0$ and prepare the state $q = |0\rangle_n |0\rangle_{n_0}$ [4]
(3) To the previous state, apply the transformation $H^{\otimes n} \otimes \mathbf{1}^{\otimes n_0}$.
(4) To the previous state, apply the transformation U_f defined by $U_f|x\rangle_n|y\rangle_{n_0} = |x\rangle_n|y \oplus f(x)\rangle_{n_0}$
(5) To the previous state, measure the last n_0 bits. We will obtain a new state.
(6) To the previous state, apply the quantum Fourier transform $U_{FT} \otimes \mathbf{1}^{\otimes n_0}$, where U_{FT} is the quantum Fourier transform on n bits.
(7) To the previous state, measure the first n bits. Obtain a state $|y\rangle_n|z\rangle_{n_0}$.
(8) Find the continued fraction representation of $\frac{y}{2^n}$. Let r be its denominator.
(9) Repeat steps 2-8 to obtain another value r'.
(10) Output the least common multiple of (r, r') and confirm this value is a period. Otherwise repeat steps 2-9.

We will prove that with probability at least 0.09, a single iteration of steps $2-10$ ends in the value r which is the period of f. Assume that f has period r.

We will show what the result of each step above produces.

The result of step 3 is that the first n bits get the n-tensor Hadamard transformation, while the last n_0 bits remain at $|0\rangle_{n_0}$. Since $H^{\otimes n}|0\rangle_n = \otimes^n (H|0\rangle)$, from the definition of $H|0\rangle$ in section 2.2 we obtain that $H^{\otimes n}|0\rangle_n = \frac{1}{2^{n/2}} \sum_{0 \le x < 2^n} |x\rangle_n$.

Therefore, at the end of step 3, the qbit is in the state $\frac{1}{2^{n/2}} \sum_{0 \le x < 2^n} |x\rangle_n |0\rangle_{n_0}$.

At the end of step 4, the qbit is now in the state $\frac{1}{2^{n/2}} \sum_{0 \le x < 2^n} |x\rangle_n |f(x)\rangle_{n_0}$.

At the end of step 5 we have measured the last n_0 qbits of the state. According to the measurement rule in section 2.3, the resulting state has a fixed value for the last n_0 bits, equal to $|y_0\rangle_{n_0}$ for some y_0 in the image of f. Since f has period r and since at the end of step 4 each of the states of the first n bits have equal weight, after measuring and obtaining y_0, the only remaining states of the first n qbits are the ones with values x such that $f(x) = y_0$, and they each appear with equal probability. If x_0 is the smallest value of x such that $f(x) = y_0$ and if m is

[3]for instance $0 \le x \le 100$.

[4]We promised we would not discuss the hardware constructions, but one way to prepare such a state is to repeatedly measure each qbit of an $(n + n_0)$-dimensional input and flip any resulting qbit states that are 1.

the smallest integer value such that $x_0 + m \cdot r > 2^n$, then at the end of step 5 the qbit is in the state $\frac{1}{\sqrt{m}} \sum_{0 \le k < m} |x_0 + kr\rangle_n |f(x_0)\rangle_{n_0}$.

Step 6 applies the quantum Fourier transform to the first n bits of the previous state. From the definition in section 2.2 and since the quantum Fourier transform is a linear operator, we know that the resulting state of the first n bits is $\sum_{0 \le y < 2^n} \left(\frac{1}{\sqrt{m} \cdot 2^{n/2}} \sum_{0 \le k < m} e^{\frac{2\pi i (x_0 + kr) y}{2^n}} \right) |y\rangle_n$. This is the state of the first n bits at the end of step 6.

Step 7 measures the first n bits. By definition, state $|y\rangle_n$ is obtained with probability $\frac{1}{m \cdot 2^n} \left| \sum_{0 \le k < m} e^{\frac{2\pi i (x_0 + kr) y}{2^n}} \right|^2$. Let's find this value. It will turn out nice. Let $z = e^{\frac{2\pi i r y}{2^n}} = \cos\left(\frac{2\pi r y}{2^n}\right) + i \sin\left(\frac{2\pi r y}{2^n}\right)$ and notice that the probability of obtaining state $|y\rangle_n$ after measurement is $\frac{1}{m \cdot 2^n} \left| 1 + z + \cdots + z^{m-1} \right|^2$.

Since z has unit norm, this value is maximized at $z = 1$.

When $z \ne 1$ then $\sin\left(\frac{\pi r y}{2^n}\right) \ne 0$ and the previous probability equals

$$\frac{1}{m \cdot 2^n} \cdot \left| e^{\frac{2\pi i x_0 y}{2^n}} \sum_{0 \le k < m} z^k \right|^2 = \frac{1}{m \cdot 2^n} \cdot \left| \frac{z^m - 1}{z - 1} \right|^2$$

$$= \frac{1}{m \cdot 2^n} \cdot \frac{|\cos\left(\frac{2\pi m r y}{2^n}\right) - 1 + i \sin\left(\frac{2\pi m r y}{2^n}\right)|^2}{|\cos\left(\frac{2\pi r y}{2^n}\right) - 1 + i \sin\left(\frac{2\pi r y}{2^n}\right)|^2} = \frac{1}{m \cdot 2^n} \cdot \frac{\sin^2\left(\frac{\pi m r y}{2^n}\right)}{\sin^2\left(\frac{\pi r y}{2^n}\right)}.$$

When y is of the form $y = j \cdot \frac{2^n}{r} + \delta_j$, for some integer j and some δ_j of absolute value less than or equal to $\frac{1}{2}$, then $\frac{\pi m r y}{2^n} = \pi m j + \frac{\pi m r \delta_j}{2^n}$. Remember that $x_0 + mr = 2^n + k_0$ for some $0 \le k_0 < r$, so $\frac{\pi m r \delta_j}{2^n} = \pi \delta_j \cdot \frac{mr}{2^n} = \pi \delta_j \frac{2^n + k_0 - x_0}{2^n} = \pi \delta_j + \pi \delta_j \cdot \frac{k_0 - x_0}{2^n}$.

However, since $0 \le x_0, k_0 < r$ and $|\delta_j| \le \frac{1}{2}$, we have

$$\left| \pi \delta_j \frac{k_0 - x_0}{2^n} \right| \le \pi \frac{r}{2^n} = \pi \frac{r}{2^{2 \cdot n_0}} < \pi \frac{r}{N^2} \le \pi \frac{1}{N},$$

which is a very small quantity when N is large.

So we can approximate $\sin\left(\frac{\pi m r y}{2^n}\right)$ by $\sin(\pi \delta_j)$.

Next notice that $\frac{\pi r y}{2^n} = \pi j + \frac{\pi r \delta_j}{2^n}$, where, as before, $\frac{\pi r \delta_j}{2^n}$ is small, so we can approximate $\sin\left(\frac{\pi r y}{2^n}\right)$ by $\frac{\pi r \delta_j}{2^n}$.

Therefore, whenever $\sin\left(\frac{\pi r y}{2^n}\right) \neq 0$, we can approximate $\frac{1}{m\cdot 2^n} \cdot \frac{\sin^2\left(\frac{\pi m r y}{2^n}\right)}{\sin^2\left(\frac{\pi r y}{2^n}\right)}$ by

$$\frac{1}{m\cdot 2^n}\cdot\frac{\sin^2(\pi\delta_j)}{(\pi r\delta_j)^2\cdot\left(\frac{1}{2^n}\right)^2} = \left(\frac{\sin(\pi\delta_j)}{\pi\delta_j}\right)^2\cdot\frac{2^n}{mr^2}$$

$$= \left(\frac{\sin(\pi\delta_j)}{\pi\delta_j}\right)^2\cdot\frac{1}{r}\cdot\frac{2^n}{mr} = \left(\frac{\sin(\pi\delta_j)}{\pi\delta_j}\right)^2\cdot\frac{1}{r}\cdot\left(1+\frac{x_0-k_0}{mr}\right)$$

$$= \left(\frac{\sin(\pi\delta_j)}{\pi\delta_j}\right)^2\cdot\frac{1}{r} + \left(\frac{\sin(\pi\delta_j)}{\pi\delta_j}\right)^2\cdot\frac{x_0-k_0}{r}\cdot\frac{1}{mr}.$$

We show that the second term in the sum above is negligible. It is a product of three terms. The first term is less than equal to 1. The second term is less than or equal to 1 in absolute value. As for the third term, notice that $mr \geq 2^n - x_0 > 2^n - r$, so $m(r+1) > 2^n$, so $\frac{1}{mr} < \frac{1}{2^n}\cdot\frac{r+1}{r} < \frac{2}{2^n}$, which is negligible.

Therefore the value $\frac{1}{m\cdot 2^n}\cdot\frac{\sin^2\left(\frac{\pi m r y}{2^n}\right)}{\sin^2\left(\frac{\pi r y}{2^n}\right)}$ can be approximated by $\frac{1}{r}\left(\frac{\sin(\pi\delta_j)}{\pi\delta_j}\right)^2$, which is greater than or equal to $\frac{1}{r}\cdot\frac{4}{\pi^2}$, because $\frac{\sin(x)}{x} \geq \frac{2}{\pi}$ for all $-\frac{\pi}{2} \leq x \leq \frac{\pi}{2}$.

Therefore, finally, the probability that qbit state $|y\rangle_n$ appears after the measurement in step 7, when $y = j\cdot\frac{2^n}{r}+\delta_j$, is at least equal to the value $\frac{1}{m\cdot 2^n}\cdot\frac{\sin^2\left(\frac{\pi m r y}{2^n}\right)}{\sin^2\left(\frac{\pi r y}{2^n}\right)}$ when $\sin\left(\frac{\pi r y}{2^n}\right) \neq 0$, which is at least equal to $\frac{1}{r}\cdot\frac{4}{\pi^2}$.

Notice that since $\frac{2^n}{r} \geq N$, the above representation of y is indeed unique. Therefore, with probability at least $\frac{r-1}{r}\cdot\frac{4}{\pi^2}$ the measured state is of the form $y = j\cdot\frac{2^n}{r}+\delta_j$ for some $1 \leq j \leq r-1$ and $|\delta_j| \leq \frac{1}{2}$. Since in step 1 of the algorithm we ensured that r is large enough, this last probability is at least equal to 0.4.

When the measured state $|y\rangle_n$ is of the form described in the previous paragraph, then $\left|\frac{y}{2^n}-\frac{j}{r}\right| = \left|\frac{\delta_j}{2^n}\right| < \frac{1}{2N^2} < \frac{1}{2r^2}$.

Here we use the first result which we don't prove: by computing the continued fraction approximations $\left\{\frac{j_k}{r_k}\right\}_{k\geq 0}$ of $\frac{y}{2^n}$ and stopping at the last step k for which r_k is less than N, we are guaranteed by the inequality above that $\frac{j_k}{r_k} = \frac{j}{r}$. Since $\gcd(j_k, r_k) = 1$ we obtain that $j_k = \frac{j}{\gcd(r,j)}$ and $r_k = \frac{r}{\gcd(r,j)}$. Therefore, with probability at least 0.4, at the end of step 8 we obtain a value of the form $\frac{r}{\gcd(r,j)}$ for a random $1 \leq j < r$.

Therefore, at the end of step 9, with probability at least 0.4^2, the two obtained values r' and r'' are both of the form $r' = \frac{r}{\gcd(r,j')}$ and $r'' = \frac{r}{\gcd(r,j'')}$ for two random $1 \leq j', j'' < r$.

With probability at least $\frac{6}{\pi^2}$, j' and j'' are relatively prime [5], and in this case the least common multiple of r' and r'' is r, as desired. This happens with probability at least $0.4^2 \cdot \frac{6}{\pi^2} > 0.097$, as desired.

We have therefore succeded in showing that with probability at least 0.09, Shor's algorithm correctly finds the period of the exponential function in a field $\mathbb{Z}_N$. A finite iteration of repeating this algorithm and confirming the period leads to the correct answer with high probability. □

4. Conclusion

In this paper, we first introduced the RSA encryption method, then proposed an algorithm for breaking it. This algorithm required a procedure for finding the period of the exponential function in $\mathbb{Z}_N$. We introduced quantum computing and described the useful tools at our disposal : qbits, operators and measurements. Using these tools we produced the missing step in breaking the RSA algorithm - the period finding procedure. Along the way we have used standard mathematical tools from number theory and complex analysis, and saw that they tie together nicely to produce a correct probabilistic algorithm. I tried to keep the paper self-contained, so that a nonexperienced reader can understand the basic premise and methods of the RSA method and Shor's algorithm.

References

[1] Mermin, N. David. Quantum Computer Science: An Introduction. Cambridge: Cambridge University, 2007. Print.

[2] `https://en.wikipedia.org/wiki/Continued_fraction`

[3] `http://www.boazbarak.org/cs127/schedule/`

374 Harvard St, Cambridge, MA 02138

[5] Quick proof: j' and j'' are relatively prime iff they are not both divisible by any prime $p < r$. This holds with probability at least

$$\prod_{p<r,\text{ prime}} \left(1 - \frac{1}{p^2}\right) > \prod_{p\text{ prime}} \left(1 - \frac{1}{p^2}\right) = \prod_{p\text{ prime}} \frac{1}{\sum_{j\geq 0} p^{2j}} = \frac{1}{\sum_{k\geq 1} \frac{1}{k^2}} = \frac{6}{\pi^2}$$