Quantum Computers - The Death of Privacy? *Latex Word Count: 2435*

Quantum computers are obfuscated by their inherent weirdness - questions quite naturally arise when one pokes at the topic. What is a quantum computer? How on earth do they work? And perhaps most poignant: why should I care? Indeed the difficulties in implementing them somewhat vindicate the latter question, but from the standpoint of an abstract computational model they offer tangible benefits to the computer that you may be reading this on. The influential hand of the internet brings a growing interconnectedness of society, but we quite often take for granted the protection of our personal information from snoopers and meddlers; indeed close to half of businesses reported cyber security breaches in 2019 [1]. How can we make this data unreadable? The answer to your question: schemes such as RSA (Rivest–Shamir–Adleman) encryption, which relies on how painful it is to decompose large numbers into a product of two primes, making the data gobbledygook to any unintended party. Every time you buy a snazzy new coat online, or send someone that very private message on your phone, RSA encryption has your back. In reality, your comfort is thanks to the exponential time complexity required to break RSA encryption - buy it a pint next time you stumble into it at the pub. But what if those private messages weren't safe, or your credit card in the hands of thieves? Well, quantum computers may threaten this safety, but hold your horses - before you descend into all out panic there are some factors that work in your favour...

The Maths Behind RSA Encryption

To understand RSA encryption, a quick refresh in modular arithmetic is required. Succinctly, $a \mod (b)$ is the remainder we get when we compute $\frac{a}{b}$ [2]. Simple? I'm sure you can convince yourself that $18 \mod (7)$ is 4, and $3 \mod (4)$ is 3. Simple enough.

The grandiosely named fundamental theorem of arithmetic states that any positive integer apart from 1 has a unique representation as a product of primes (POP). Basically, multiply together a very specific selection of primes and you can arrive at whatever number you want.

Integers a and b are said to be relatively prime if there are no common primes in their POP decomposition. For some integer n, imagine $s(n)$ is the set of positive integers smaller than and relatively prime to n. Euler came up with the phi function $\phi(n)$, defined as the cardinality or length of $s(n)$. When we can decompose n into a product of two primes, p and q, we have that $\phi(n) = \phi(p \times q) = (p-1)(q-1)$ [2].

Let's put this jargon to use! Let's say you want to securely send the meaning of life, 42, to your friend John. Now, imagine (as in table 1) that each and every person has two publicly accessible numbers, N and c. N_{john} is a painfully large number and moreover a product of two smaller - but still sizeable - primes, p and q, which quite rightly only John knows. Moreover, c_{john} is a very (very) large number that is relatively prime to $\phi(N_{john}) = \phi(p \times q) = (p-1)(q-1)$ [2]. Of course, John would never publish $\phi(N_{john})$ as one could then easily determine p and q!

We take the number 42, and apply the mapping: $m \mapsto m^c \mod (N_{john})$ to get the encrypted message which we shall call $\tilde{m}$ [2],[3]. Now beckons the question - How on earth would John decrypt this gibberish? Indeed this is a little more painful but

Name	N	c
John	N_{john}	c_{john}
You	N_{you}	c_{you}

Table 1: *Our Publicly accessible RSA details. Adapted from [2].*

still relatively doable.

Firstly, John would find the integer x that satisfies $x \times c_{john} = 1 \mod (\phi(N_{john}))$. He then calculates $(\tilde{m})^x \mod (N_{john})$ to retrieve m [3]. Phew!

Now why is this whole process airtight? Well, to decrypt the message John needs x. We know x can only be retrieved if $\phi(N_{john})$ is known and since $\phi(N_{john}) = (p-1)(q-1)$, one must know p and q [2]. The real crux of the process is that finding p and q is exceptionally time consuming if you don't know it already - if N has n bits, then the best time complexity we can get is $2^{O(n^{1/3})}$ [4], which grows exponentially with the number of digits. Hopefully you'll forgive how irritating I was in saying N_{john} was large so many times.

Using some laborious maths, the problem of factoring N_{john} into p and q essentially boils down to period finding [5].

Going Beyond Classical

Now I'm almost certain you've encountered the binary number system; the parlance of every single computer you've ever used. A 1 is represented by the presence of an electrical signal, and 0 by the absence. These are the fundamental units of computation called bits, or to avoid later ambiguity, cbits [6]. Different permutations of 0's and 1's can be used to transmit information [7]. For example 101001 is equivalent to the decimal representation 41. What use is this at all? Well, devising mappings from binary numbers to letters allow for the transmission of information; for instance "hi" may be represented as 01101000 01101001 using the ASCII (American Standard Code for Information Interchange) encoding system [8].

Quantum computers do things a little differently. Memory states are represented by what are called ket-vectors. For instance, the one state is defined as $|1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$, and the zero state is defined as $|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}$. These are known as basis states [7]. Most have heard of a notorious and somewhat fatigued thought experiment where a cat is put in a the box, and poison is released as a result of (an entirely quantum) radioactive decay. The poor little thing is described as being dead and alive[1] - in a superposition of the two "states". Similarly, any such memory system can exist in a linear superposition of the basis states, and so we construct the general state - known more formally as the wavefunction - of a single cubit as $|\psi\rangle = \alpha|1\rangle + \beta|0\rangle$ [9],[6]. The numbers α and β are called amplitudes may be real or complex, but satisfy the normalisation condition: $|\alpha|^2 + |\beta|^2 = 1$ [6]. The careful eye might notice that we are essentially giving a weighting to each state; if $\alpha > \beta$, there must be some "preference" of state $|1\rangle$ over $|0\rangle$.This is somewhat true - $|\alpha|^2$ is simply the probability that once we measure the memory state (don't worry yourself on how on earth we'd do this) the state is in the $|1\rangle$ [6]. After the measurement, the memory state "collapses" to either $|0\rangle$ or $|1\rangle$ permanently. As a matter of fact, we don't really care about $|0\rangle$ and $|1\rangle$, we could use any vectors that are linearly independent; they'll do a fine job of acting as the basis to construct $|\psi\rangle$ [5]. Quite obviously we'd like something a little more sophisticated. A two qubit state is the linear combination of new basis states: $|0\rangle \otimes |0\rangle = |00\rangle$, $|0\rangle \otimes |1\rangle = |01\rangle$, $|1\rangle \otimes |1\rangle = |11\rangle$ and $|1\rangle \otimes |0\rangle = |10\rangle$ [7]. So for two qubits, we can write $|\psi_2\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|11\rangle + \delta|10\rangle$. Here $\otimes$ is called the tensor product, defined by [10]

$$\begin{bmatrix} a_0 \\ a_1 \end{bmatrix} \otimes \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} = \begin{bmatrix} a_0 \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} \\ a_1 \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} a_0 b_0 \\ a_0 b_1 \\ a_1 b_0 \\ a_1 b_1 \end{bmatrix}.$$

You might guess that generalising this for a memory state of n qubits involves forming the linear combination of all $N = 2^n$ basis states, $|B_i\rangle = |b_1 b_2 \dots\ b_j \dots b_{n-1} b_n\rangle$ where b_j can be a 0 or 1 [7]. As before, we can write the superposition as $\psi_n = \sum_{i=1}^{N} \alpha_i |B_i\rangle$ with the condition $\sum_{i=1}^{N} |\alpha_i|^2 = 1$. Once again, measurements cause the memory state to collapse into one of the $|B_i\rangle$ with probability $|\alpha_i|^2$. It is this superposition that makes quantum computers so special; being able to act on all the states at once means a quantum computer with 300 qubits could perform more calculations in a fraction of a second than there are atoms in this huge universe [11].

Logic Gates, The Hadamard Gate and Deutsch's Problem

Logic gates are an abstract model of computation that operate using the principles of Boolean algebra;

[1]No cats were hurt in the writing of this article...

think true or false, 1 or 0 [12]. For instance, "you've fallen asleep from boredom" is false (0) and "I'm definitely overthinking this example" is true (1). In short, logic gates take a true or false input and return one of the two. Not too complicated. To put this into perspective, let's look at the imaginarily named NOT gate. We can represent its behaviour using a grandly named truth table:

Input	Output
0	1
1	0

Table 2: *Truth table for a NOT gate [12]*

As you might have guessed by now, like the edgy kid in school, quantum computers like to do things differently. The equivalent of the not gate is the Pauli-X gate - I know it sounds cool, but it isn't. It is represented as a matrix [9]

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

One important thing to note is that these logic gates - transformations in maths speak - need to preserve the magnitude squared of whatever they're acting on, so the corresponding matrices must be unitary [5]. The Hadamard gate is a little (but only slightly) more interesting. It maps $|0\rangle \leftrightarrow |s\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|1\rangle \leftrightarrow |d\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, and so may be represented by the matrix [9]

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

The Hadamard gate really comes into its own in something called Deutsch's problem; perhaps the simplest (and earliest) way to demonstrate why a quantum computer is that much more powerful. Let's say after a night out you stumble into your house in a drunkenly state only to forget if that one pesky switch is connected to your porch light. You definitely need this light - stumbling in the dark isn't a good idea sober let alone drunk. Now, you of course know that you'd need to observe the state of the light at two separate switch positions [5]; firstly in the initial state of the switch you must observe the initial state of the light. After a flick of the switch you need to once more look at the light. Quite obviously, if the state of the light has changed in the process, you must be pecking at the right switch.

However difficult it may be in your current drunken state, let's express this mathematically. If the light system is represented as the function L, then we can say $L\colon \{0,1\} \mapsto \{0,1\}$. Here the domain $\{0,1\}$ represents whether the switch is down or up, and the codomain $\{0,1\}$ represents the light being off or on. If $L(0) = L(1) (= 0 \text{ or } 1)$ the state of the light is independent of the switch position so the switch is therefore not connected to the light. Similarly, if $L(0) \neq L(1)$ then the switch is connected to the light [5]. To investigate if $L(0) = L(1)$, we need two uses of L, one for $L(0)$ and the other for $L(1)$, to determine if the switch is connected. What about in the weird world of quantum mechanics? In short, superposition allows us to do this entire process more efficiently; in fact, only one use L is needed. You can think of this as checking the light for two different positions of the switch at once, since the switch can be in a superposition of on and off [5]. This sort of thing falls into a class of computation called Quantum Parallelism. The light for our quantum case is represented by the black box in figure 1. Let's prepare the switch in the state $|0\rangle$ and feed it into the Hadamard gate to get a neat little superposition in figure 2. L can act on the superposition, treating each state independently, and another Hadamard is used to get back to $\pm|0\rangle$ or $\pm|1\rangle$. The state being in $\pm|0\rangle$ requires $L(0) = L(1)$, and so is the case in which the switch is connected [5].

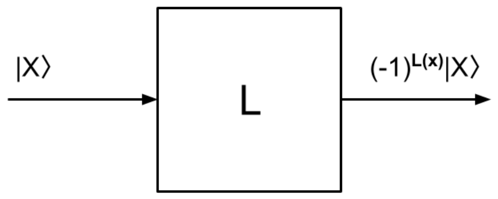
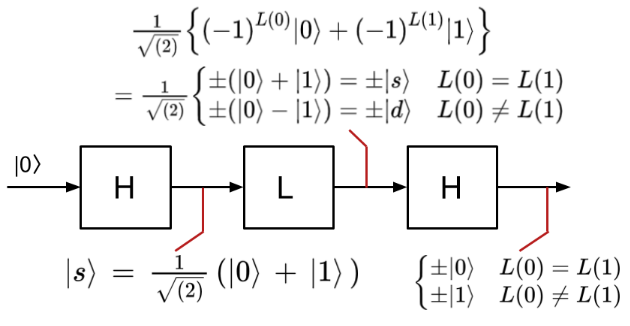


Figure 1: *The function,L, applied to a general quantum state $|x\rangle$. Adapted from [5]*



$\frac{1}{\sqrt{(2)}}\left\{(-1)^{L(0)}|0\rangle + (-1)^{L(1)}|1\rangle\right\}$
$= \frac{1}{\sqrt{(2)}}\begin{cases}\pm(|0\rangle + |1\rangle) = \pm|s\rangle & L(0) = L(1) \\ \pm(|0\rangle - |1\rangle) = \pm|d\rangle & L(0) \neq L(1)\end{cases}$
$|0\rangle$
H
L
H
$|s\rangle = \frac{1}{\sqrt{(2)}}(|0\rangle + |1\rangle)$
$\begin{cases}\pm|0\rangle & L(0) = L(1) \\ \pm|1\rangle & L(0) \neq L(1)\end{cases}$

Figure 2: *How we'd go about using the switch once to test the light. Adapted from [5]*

Deutch interestingly came to the conclusion that

what he was doing was finding the period of this function L. Let's imagine a periodic function $p(x)$. The condition for periodicity is $p(x) = p(x+T)$, straight away noting that T is the period of the function. Applying this to our example, we can write:
$L(0) = L(1)$ goes to $L(x) = L(x+1)$ and
$L(0) \neq L(1)$ goes to $L(x) = L(x+0)$.
Essentially, the switch is connected to the light if $L(x)$ has only the trivial case of zero periodicity [5].

Period Finding and The Quantum Fourier Transform

You would have experienced period finding in the classical sense before; the Fourier transform of a function, say $f(x)$, is a way of stepping out of the spatial domain and into the spatial frequency domain, $\tilde{f}(k)$. If I were to give you a set of discrete data in vector form, you might half-heartedly carry out a discrete Fourier transform (DFT) on the data, perhaps using python or your weapon of choice. Now a DFT is long to calculate, really long, so we can speed the entire process up using a fast Fourier transform (FFT) or even better a quantum Fourier transform (QFT). The DFT and QFT share almost the exact same formalism - the difference is on the data in which they operate and a pesky normalisation factor [13]. The real difference is that the QFT acts on the amplitudes of our quantum state. Remember when we wrote a general memory state or wavefunction of n qubits as $\psi_n = \sum_{i=1}^{N} \alpha_i |B_i\rangle$? Applying a QFT would involve transforming α_i to $\tilde{\alpha_i}$ - the representation in reciprocal space - and so $QFT(\psi_n) = \sum_{i=1}^{N} \tilde{\alpha_i} |B_i\rangle$. As a matter of fact, the QFT has a time complexity of $O(\log^2 N)$ as opposed to $O(N \log N)$ for the FFT - an exponential time improvement [13],[14]. Remember when we brushed over the fact that factoring was fundamentally a period finding problem? Here comes the nail in the coffin - the QFT is used in something called Shor's algorithm to find the period of the function involved in the factoring problem much more quickly [9]. I suppose now you see how quantum computers are able to threaten the integrity of RSA encryption. They attack the main working principle - the difficulty of factoring large numbers - head on.

Why Should I Care?

Phew! That was a whole load of information. Now what does this actually mean? Well in short it's possible for somebody to devise a machine that can break RSA encryption. The impacts of of this are far reaching. Consider for instance the impact of such technologies on the already fraught terrain of national cyber security or indeed electoral politics. The impacts could be equally far reaching for how citizens of states are surveilled by their governments and the ramifications of such technologies for civil rights and liberties.

Hold your horses! Before you absolutely lose it there are several caveats. Firstly, there's the difficulty of constructing such a computer. Interactions from the outside environment such as heat flow, vibrations (or indeed other mechanical disturbances) and electromagnetic wave interference cause decoherence whereby the system loses its "quantumness" and becomes more and more classical [15]. Moreover, looking for errors in quantum computers is tricky - observing the state in the first place would cause the system to collapse into one of the basis states, breaking the superposition that we're so fond of [16]. Finding errors in Quantum computers is like trying to find that pesky socket under your bed to charge your phone in the pitch black of night. Typical error corrections with cbits involve making several copies of the same bit, and so instead of transmitting 1, 1111 could be transmitted instead. Majority voting relies on the fact that only a few bits of the whole cohort would flip, and so the correct state of a given bit is determined by the most common digit. Unfortunately, the no-cloning theorem in quantum mechanics makes this difficult - its impossible to clone a qubit without effecting it's state [16]. This is a big problem - the most intuitive error checking involves making a copy at some point.

Perhaps more philosophically, this is a classic case of built in obsolescence - much like an overly greedy predator eating all of its prey and dying of starvation - we can be sure that any breakthroughs will be met by comparable innovation in encryption [5]. As a matter of fact, several encryption methods already exist that are conjectured to be "quantum secure". The most popular of these fall into a class of methods called hash-based cryptography, including XMSS (eXtended Merkle Signature Scheme) and LMS (Leighton-Micali Signature system)[17].

References

[1] UK. Gov, *Cyber Security Breaches Survey 2020.*

[2] S. G. Krantz, *A Mathematical Odyssey Journey from the Real to the Complex*, 1st ed., 2014.

[3] K. Mann, *The science of encryption: prime numbers and mod n arithmetic.* University of University of California, Berkeley, available from https://math.berkeley.edu/ kpmann/encryption.pdf.

[4] U. Vazirani, *Quantum Mechanics Quantum Computation, Lecture 10: Quantum Factoring.* University of California, Berkely.

[5] P. Hayden, *The Quantum Computational Universe Lecture Series.* Stanford University, Institute for Theoretical Physics. [Online]. Available: https://www.youtube.com/watch?v=AqWuyeh0SxQ&t=3032s&ab_channel=StanfordInstituteforTheoreticalPhysics

[6] A. Dawa, *Quantum Computing, Lecture 1.* University of Cambridge. [Online]. Available: https://www.cl.cam.ac.uk/teaching/1213/QuantComp/lecture1.pdf

[7] N. D. Mermin, *Quantum Computer Science: An Introduction.* Cambridge: Cambridge University Press, 2007.

[8] *ASCII To Binary Converter.* [Online]. Available: https://www.rapidtables.com/convert/number/ascii-to-binary.html

[9] R. de Wolf, *Quantum Computing: Lecture Notes.* CWI and University of Amsterdam, 2019. [Online]. Available: https://arxiv.org/abs/1907.09415

[10] J. Wright and Y. Ding, *CMU 18-859BB, Lecture 2: Quantum Math Basics.* Carnegie Mellon University, 2015. [Online]. Available: https://www.cs.cmu.edu/~odonnell/quantum15/lecture02.pdf

[11] C. Choi, *How Many Qubits Are Needed for Quantum Supremacy?* IEEE. [Online]. Available: https://spectrum.ieee.org/tech-talk/computing/hardware/qubit-supremacy

[12] S. H. Unger, *The essence of logic circuits*, 2nd ed., 1996.

[13] U. Vazirani and B. Bukh, *CS 294-6, Quantum Computing, Lecture 8.* University of California, Berkeley, 2004. [Online]. Available: https://people.eecs.berkeley.edu/~vazirani/f04quantum/notes/lec8.pdf

[14] T. Brun, *EE 520: Introduction to Quantum Information Processing, Lecture 13.* University of Southern California, 2017. [Online]. Available: https://viterbi-web.usc.edu/~tbrun/Course/lecture13.pdf

[15] S. Pakin and P. Coles, *The Problem with Quantum Computers.* Scientific American, 2019. [Online]. Available: https://blogs.scientificamerican.com/observations/the-problem-with-quantum-computers/#:~:text=Quantum%20computers%20are%20exceedingly%20difficult,chance%20to%20run%20to%20completion

[16] A. Cho, *The biggest flipping challenge in quantum computing.* Science, 2020. [Online]. Available: https://www.sciencemag.org/news/2020/07/biggest-flipping-challenge-quantum-computing

[17] F. Campos, T. Kohlstadt, S. Reith, and M. Stöttinger, “Lms vs xmss: Comparison of stateful hash-based signature schemes on arm cortex-m4,” in *Progress in Cryptology - AFRICACRYPT 2020*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2020, pp. 258–277.