Unintended Effects of the Development of Quantum Computing on NP-Difficult Problems and the RSA Encryption Algorithm

Matt Fletcher

Abstract—Since the 1980's, the RSA Encryption algorithm has been used as a near foolproof method of encrypting sensitive data sent over the Internet. It relies on the relatively slow computing power of modern day computers as well as prime factorization being an NP-TODO math problem TODO check which version of NP. However, with quantum computers on the horizon of being invented, many cryptologists worry about the safety of encrypted data. This paper analyzes why the safety of encryption has both increased and decreased over the age of computers.

Index Terms—Quantum computing, Cryptography, Encryption, P vs NP, RSA, TODO

1 Introduction

In 1955, mathematician John Nash penned a letter to the National Security Agency that would change the face of computing forever. TODO add citation In this letter, he presented a theory about the computational power required to find the solutions to various mathematical problems. In particular, he focused on the art of cryptography, and the length of time necessary to crack an encryption. Up to that point, computer scientists who created algorithms concentrated solely on the required time to solve a particular length of problem. Nash suggested that instead of fixating on the length of time necessary to solve a problem of a particular length, scientists should consider the rate of difficulty growth of the problem resolution time given the length of the inputs grew at a linear rate. Furthermore, in lieu of computation time, a scientist should focus on the number of computational steps required to solve the problem. TODO adjust sentences A computational step is the number of state changes that the machine processes TODO synonym for processes in carrying out the set of steps to solve the problem. The shape of this growth curve would help classify the difficulty of the problem. For example, the addition of numbers is a problem with a characteristic of linear growth. Doubling the number of digits to be added results in approximately double the computation time. This makes these problems grow at a linear rate[1]. However, problems such as multiplication grow at a slightly faster pace. Doubling the length of the inputs results in a growth of 2^2 times the number of computational steps. Tripling the length of the input results in a growth of 3^2 times the number of computational steps. Although this appears to be a rapid growth, it is actually growing in what is known as polynomial time. The exponent of the growth rate is always constant. Polynomial growth can therefore be displayed as some combination of terms in the form n^k, where n is a variable and k is a constant.

1. Linear growth is still technically considered a polynomial growth of the form n^k with $k = 0$.

However, there is additional class of problem. Take for example cracking a password. For simplicity sake, assume the password is a PIN composed only of digits 0-9. In the worst case scenario, for a 1 digit PIN, the computer would take $10^1 = 10$ guesses. For a 2 digit PIN, the computer would take $10^2 = 100$ guesses. For a 4 digit pin, the computer would take $10^4 = 10\,000$ guesses. This is growing at a rate of 10^k, where k is some constant. This growth pattern is known as exponential growth, where the variable is in the exponent. The resultant curve is extremely sharp. For context, if a password was able to consist of any numbers, letters, or special characters on a standard US keyboard, with only an eight character password, the number of computational steps required skyrockets up to 82^8, or 2 with fifteen zeros following it.

1.1 Classifying P and NP Problems

Mathematicians had been noticing problems like this in all different fields of math and science. So, they decided to classify these problems. If a problem could be solved in a polynomial amount of time, it was classified as difficulty P. Problems with difficulty P are both easy to solve, and easy to verify that the calculated answer is correct.

Problems classified as NP are slightly different. However, "Nondeterministic Polynomial time problems" (also known as NP problems) are difficult to solve, but easy to check a solution to in polynomial time[2]. One of the most common examples of an NP problem is finding a subset of a list that satisfies a given requirement. This is often referred to as the hackey-sack problem. If someone is presented with a collection of small sacks with specified random weights, and then asked to find a subset of those sacks that result in a given weight, the only way to approach this problem with current computational methods is a brute force algorithm,

2. "Easy" and "difficult" in this case mean that they are respectively solvable and not solvable by a computer in a reasonable amount of time.

guessing a random subset, and checking the result. Yet another example of an NP problem is the factoring of a number into its prime factors. In his 1801 book *Disquisitiones Arithmeticae*, mathematician Frederick Gauss proved that any number has exactly 1 prime factorization. It is extremely easy to check if a given factorization for a number is correct (the multiplication of said numbers is a P-difficulty problem). However, in order to find the unique factorization for a number, the only method currently known is guessing and checking every single factor for that number. As the length of the target number grows, so does the length of computation time, but the computation time grows exponentially. As the growth of computation time is not of a polynomial form, the factorization problem is considered an NP-difficulty problem. But what application does this have in the real world?

1.2 RSA Encryption

As a shopper enters their credit card details for an online store website, the shopper is assured that their credit card details are protected if they see the little green lock icon in their browser. Less known is the fact that the only reason this lock icon has any significant value is due to prime factorization being an NP problem, and the difficulty of breaking RSA encryption. In 1978, three computer scientists, Ron Rivest, Adi Shamir and Leonard Adleman, publicly released an encryption algorithm titled after their names, RSA. The basis of this encryption technique[3] is two secret prime numbers (known as private keys) multiplied together to form a public key. Determining whether the two numbers multiplied together result in the public key is only possible through repetitive calculation [4]. For an idea of size, the public key length for most banking systems nowadays is over six hundred digits long[5]. To complete these calculations with current methods and computing technologies would require somewhere on the order of 6.4 quadrillion years.

1.3 Evolution of Computational Methods

The phrase "with current methods" is used throughout this paper, in the context of methods used to solve a problem. This aligns with a computing device known as a Turing computer to solve a problem. In 1936, English computer scientist and cryptologist Alan Turing proposed the idea of a theoretical computing machine that operates off of a memory tape with infinite length divided into discrete cells. Each cell contains a basic instruction. The reading head above the tape identifies the value of the cell on the memory tape, then moves the tape either to the left, to the right, or terminates the program. Although this may seem very simple, determining if an algorithm can be run on a Turing machine constitutes one of the most important problems in the field of computer science. A Turing computer is the most basic computer possible, taking in bits one at a time, and outputting a result based on the bit. These computers are powerful. However, they are insufficient for solving NP problems. Because the difficulty for NP problems grows with non-polynomial growth, the computation time grows rapidly to an unmanageable amount. However, this challenge will change with the introduction of quantum computers.

Computers have evolved drastically over the decades. In first generation computers, or computer technology designed from 1939 to 1954, computers operated on vaccuum tubes. Even a basic computer with hardware specifications orders of magnitude weaker than a modern day pocket calculator would occupy an entire room, and cost hundreds of thousands of dollars. These vacuum tubes, due to the excess heat they created, were extremely unreliable and were prone to failure every few hours. Computers soon transitioned to using transistors, micro-switches that can read either 1 (for on) or 0 (for off). They will exist in exactly one of these states at any given time. By combining these transistors into logical gates, basic functions can be performed, such as addition and multiplication. Compared to the vacuum tubes, transistors could be packed far more densely into the same area. Due to Moore's law [6], the processing power of computers grows at an exponential rate. Eventually, transistors were changed to a special type of transistor known as a MOSFET[7]. Due to their low power usage, MOSFET's could be packed into extreme densities. Eventually, these transistors were integrated into the layers of silicon in the processing chip, which allowed the size of an individual transistor to be in the tens of nanometers[8]

A quantum computer operates uses a slightly different principle. Instead of operating on a physical piece of hardware (the transistor) that produces a bit, it operates by observation of a photon that represents a qubit[9]. A qubit is a special type of bit that can either represent a 1, a 0, or a superposition of a 1 and a 0. But how can something exist in both a 1 and a 0 at the same time? To understand this, one must first understand the principle behind Schrödingers cat.

The infamous Schrödinger's cat experiment is performed as follows. Imagine a living cat inside a sealed bunker, with a sealed beaker of poisonous material, a Geiger counter attached to a hammer, and a small piece of radioactive material. The radioactive material has a 50% chance of having one of its atoms decay in the next hour (which would set off the Geiger counter, smashing the hammer into the vial of gas and releasing it into the bunker, killing the cat). As long as the bunker is sealed and unobserved, the cat has a 50% chance of being dead and a 50% chance of being alive. When the bunker is opened, however, the cat must either be dead or alive. The instant before the bunker is opened, the cat exists in a quantum superposition of both dead and alive.

3. Please note that this is a highly simplified explanation of the encryption algorithm. To describe this in full detail would take far more in-depth explanations and mathematics than would be appropriate for this paper.

4. Although some algorithms exist to reduce the work slightly using methods to eliminate groups of guesses, it is still a polynomial reduction on an exponential growth, which means the reduction is almost unnoticeable.

5. A 617 digit long public key is the key length for 2048 bit encryption.

6. Moore's law is based off of Intel co-founder Gordon Moore's observation that the density of transistors on microprocessors had approximately doubled every year since the early age of computers. Moore's law states that this trend will continue every year into the future.

7. Metal Oxide Semiconductor Field Effect Transistors

8. For example, the Intel i7 Quad Core CPU packs 731 million transistors into a board only 0.63" by 0.63".

9. Qubit is a juxtaposition of QUantum BIT

Just like the cat, as long as the qubit is unobserved, it exists in a quantum superposition both a 1 and a 0. As soon as the qubit is observed, though, it "snaps" to either a 1 or a 0. Another way of thinking about this is like a coin that is flipped and is spinning in midair. At any point in the air, it is impossible to figure out if it is heads or tails. However, smashing a fist down on top of the coin will force it to go to either a heads or a tails orientation. TODO explain this better. But how does this help a quantum computer work faster than a regular computer?

1.3.1 Subsubsection Heading Here

Subsubsection text here.

2 Quantum Computing Pros and Cons

What are the advantages and disadvantages that come with the rise of quantum computing?

2.1 Advantages

The United States Military expresses a large amount of interest in quantum computing, primarily for its use in cryptanalysis, or code-breaking.

Another use of quantum computing arises in the use of autonmous vehicles and route generation. Finding the shortest route between a set of points is considered an NP-hard problem [10]. In math terms, this problem is attempting to find the most efficient Hamiltonian Cycle to take through a set of N nodes.

TODO add more.

2.2 Disadvantages

Appendix A
Proof of the First Zonklar Equation

Appendix one text goes here.

Appendix B

Appendix two text goes here.

Acknowledgments

The authors would like to thank Dunkin Donuts, Reddit ELI5, Wikipedia, and whoever invented copy/paste.

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

10. A simple explanation behind this statement: To find the shortest route between a set of nodes $N_1, N_2, \cdots N_Q$ of size Q, start at any node N_1. At this point, Q guesses have been checked. Now, there are $Q-1$ points left to try. After another point N_2 has been selected, $Q(Q-1)$ guesses have been attempted. After another point, the number of guesses goes to $Q(Q-1)(Q-2)$. This is increasing at a rate of $Q!$, which is a non polynomial growth.

PLACE PHOTO HERE

Michael Shell Biography text here.

John Doe Biography text here.

Jane Doe Biography text here.