Period Finding on a Quantum Computer using Shor's Algorithm

Samuel Marcillo-Gomez
smarcillogomez@gmail.com

Dr. Alberto La Cava
alacava@saintpeters.edu

Saint Peter's University
2641 John F. Kennedy Blvd, Jersey City, NJ 07306

In recent years, there have been numerous developments in quantum computation. These developments have brought into question, how quantum computers could affect security have risen. For instance, Shor's algorithm is believed to be able to break certain encryptions faster on a perfect quantum computer faster than on, what is known as, classical computers. In a few years or decades, there could be significant developments made that allow for quantum computers to perform Shor's Algorithm. As quantum computers exist now, the implementation of the algorithm is known to be difficult as the computes are very basic. Attempts to create quantum circuits that can compute Shor's Algorithms aid in the understanding of the algorithm.

Keywords

Shor's Algorithm, Quantum, IBM, Cryptanalysis, RSA Encryption, Public Key Cryptography.

1. INTRODUCTION

The most import basis in cyber security is encryption. It provides privacy, anonymity, authorization, authentication, integrity, and assurance. Some of the security algorithms used today are results of many before them that were broken. Examples of this are seen in symmetric cryptography with algorithms like DES and Triple DES. Both precede the symmetric cryptographic algorithm, AES and both were broken, or were insecure. RSA, an asymmetric cryptosystem, has withstood attempts to break it algorithm; however, Elliptic curve cryptography has been becoming more popular as the sizes of its keys are not required to be as big a RSAs. That being said, with the innovation of quantum technology, there is the chance that asymmetric cryptosystems, like RSA and ECC, could potentially become insecure in the future. This idea stems from an algorithm created by, MIT Professor, Peter Shor, in which he showed that a quantum computer could potentially solve RSA Keys and ECC Keys in polynomial time, which is considered to be much faster than what can be done on a classical computer. This has become apparent to departments of government like the NIST (National Institute of Standards and Technology) expressed concern in the rise of quantum technology and are in the process of finding a potential replacement for the RSA cryptosystem, in a Post Quantum World [5]. This excitement over quantum computing, leads one to take into account how and why quantum computers can do what classical computers cannot. To understanding the problem, the goal of this paper is to comprehend how Shor's Algorithm works, specifically with RSA, and to see if the current state of quantum computers is able to implement the algorithm.

2. LITERATURE REVIEW

To begin, a basic understanding of RSA cryptosystem would be helpful in the pursuit to understand Shor's algorithm. From the book, *Understanding Cryptography: A Textbook for Students and Practitioners,* the equation for the cryptosystem goes as followed:

m = message in plain text

c = ciphertext after encrypting message

Encryption: $c \equiv m^e \pmod{n}$

Decryption: $c^d \equiv (m^e)^d \equiv m \pmod{n}$

While *d* is only known to the recipient, an attacker could intercept the encrypted message and attempt to decrypt it by performing a brute force attack; however, performing such an attack is not practical as real RSA keys are significantly large and can range in size of 512 to 1024 bits, and sometime larger. For this reason, mathematical attacks have been conceived to calculate the decryption key more efficiently than brute force attacks [6]. In a Mathematical Attack, instead of guessing the decryption key *d*, the attacker attempts to factor the prime the values of *n*, which are known as *p* and *q* [6]. If *p* and *q* are calculated, the attacker can then figure □ (*n*) as, □ (*n*) = (*p-1*)(*q-1*), and thus calculate the decryption key, *d*, as $d \times e \equiv 1 \bmod n$. However, again arises the problem: because *n* can be a very larger number determining *p* and *q* would take a very long time to calculate [4]. The most asymptotically efficient classical algorithm that can find *p* and *q* is the number theoretic sieve, which factors an integer *n* in time $O(\exp[(\log(n))^{1/3} (\log(\log(n)))^{2/3}])$ [3]. Shor's algorithm is known to be theoretically faster than this as it is not simply factoring *n*. Below it is the steps of Shor's Algorithm [3]:

Step 1:

Choose a number $1 < a < n$ such that is relatively prime with *n,* meaning gcd(*a, n*) = *1*. If the gcd(*a, n*) does not equal 1 then *k* is a factor of *n* and the algorithm ends; otherwise, proceed to **Step 2** [3].

Step 2:

With the value *a*, determine the period, *r*, of the equation *a* mod *n* such that $a^r mod\ n \equiv 1$ [3].

Step 3:

If *r* is found such that $a^r mod\ n \equiv 1$ is true but is odd, then repeat **Step 1**. If *r* is even, proceed to **Step 3**[3].

Step 4:

Since $a^r mod\ n \equiv 1$ is true, then $a^r - 1 \equiv 0 \bmod n$ is also true, which means $a^r - 1$ is a multiple of *n*. Thus there must exist some value integer *k*, such that

$a^r - 1 = kn$. Since r is even, the above equation can be rewritten as, (k is unnecessary for the rest of the equation) [3].

$$(a^{r/2} - 1)(a^{r/2} + 1) = pq$$

Step 5:

Finally, if neither $(a^{r/2} - 1)$ nor $(a^{r/2} + 1)$ are congruent to 0 mod n, then $n = ((a^{r/2} - 1)/p) * ((a^{r/2} + 1)/q)$, and it can be said that, [3].

$$p = \gcd(a^{r/2} - 1, n)$$

$$q = \gcd(a^{r/2} + 1, n)$$

Thus p and q have been determined. The most significant part of Shor's algorithm is **Step 2** as it is the hardest part, for classical computers to do especially if n is a large number [3]. For this reason, Shor's uses a quantum computer to calculate the period of a mod n. Quantum computers have a property known as superposition in which a quantum bit can be in the state of a combination of 1 and 0. When the bits a measured the bit chose a state of either 1 or 0. Using this property Shor's algorithm is able to factor an integer n that takes asymptotically $O(\log(n)^2 \log(\log(n)) \log(\log(\log(n))))$ steps which is polynomial time in the number of digits $O(\log(n))$ of n [7]. Essentially, using the property of superposition would allow to multiple testing $a^r \bmod n \equiv 1$. The quantum portion of Shor's algorithm goes as follows: create two quantum registers and set their sizes to $n^2 < q < 2n^2$, and n - 1 respectively; put register 1 in the uniform superposition of states representing numbers a (mod q) and load register 2 with all zeros; then compute $a^r mod\ n \equiv 1$ in the second register; perform an inverse Quantum Fourier Transform on the first register; lastly, measure the first register [7][3]. The algorithm should give the period of a mod n, which could then be used to find d. The heart of Shor's algorithm is the Quantum Fourier Transform, as it should use resonances to amplify the basis states associated with the correct period and the incorrect answers destructively interfere, which suppress their amplitudes, increasing the speed of algorithm to polynomial time [7].

3. RESEARCH METHODS

In order to begin to determine how Shor's algorithm will affect RSA encryption, it would be beneficial to test current circuits that implement Shor's Algorithm on a quantum computer. For this testing process the web-based user interface, Jupyterlab, running Python Version 3.7.0, in conjunction with IBM's Qiskit API, will be used to create and simulate quantum circuits running on a quantum computer. These programs will be running on a 11-inch MacBook Air, with 8 Gigabytes of memory, running OS X El Capitan 10.11.6. This research is to essentially examine the state of current state of quantum computing as well as the implementation of the algorithm.

4. RESEARCH

To test Shor's Algorithm, experiments were conducted by attempting to implement the algorithm on a quantum computer. To do this, the circuit would be broken up into four parts based on Shor's algorithm. First part is superposition, which is easy to achieve since the Hadamard gates do that already; the second part is modular exponentiation, which is difficult to achieve considering that this can only be achieved using quantum logic gate; third step is to implement the Inverse Quantum Fourier Transform (QTF), which already exist, and, for the purposes of this experiment, will be gathered from the IBM Q Experience community GitHub user, delapuente; lastly, the measurement which is included in the IBM quantum API. All the parts of the circuit are available except for the modular exponentiation part. In *Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance*, a circuit was created to find the period of $7^r \bmod 15$. The circuit yields the correct results, outputting 4 and 0. The value 0 can be ignored since $7^0 \bmod 15$ will always be 1 and is a side effect of the Inverse QFT. The other result 4, is the period of $7^r \bmod 15$. It is the only circuit that was researched so far, that is able to find the period. Figure 1 is a recreation of the circuit in which $n = 15$ and $a = 7$. The design of the circuit is different from Shor's Algorithm, as all the operations are done in one register instead of two.

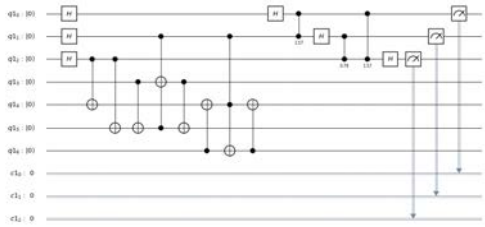

Figure 1: Circuit designed to calculate the period of $7^x mod\ 15$

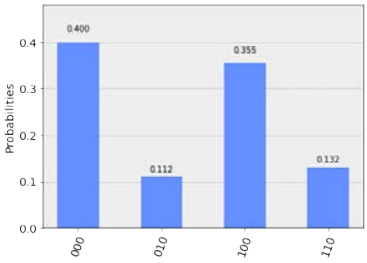

Figure 2: Results from circuit of Figure 1. Yield 0 and 4, which when calculated is $7^4 mod\ 15 \equiv 1$. *Therefore 4 is the period which yields the results* $gcd(7^2 - 1, 15) = 3$ *and* $gcd(7^2 + 1, 15) = 5$ *which are the prime factors of 15.*

Using the circuit from Figure 1 as a guide, a new circuit will be designed. Replacing the part of the circuit that is used for modular exponentiation should yield a different result. In Figure 1, it would appear that the control qubits of the controlled not gates (CNOTs) are in the same position as the results of the circuit which may have a correlation. To test this, in the second attempted period finding circuit, the control qubits of CNOTs should represent the desired period in the circuit. The circuit is designed to find the period of $7^r \bmod 55$. For the equation $7^r \bmod 55$ the expected period is 20, therefore the control qubits will be in Register bit position 2 and 4, which represent the values 4 and 16 respectfully. The number of the initial Hadamard gate and size of the QFT were determined by the number of bits in 55, which is 6.

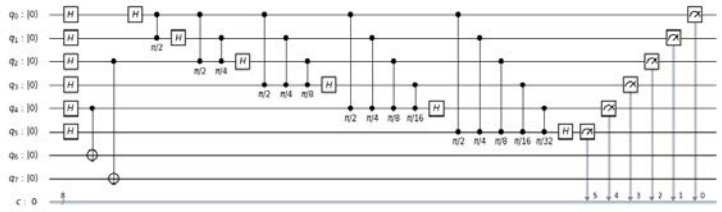

Figure 3: Quantum circuit designed to find period of $7^r \bmod 55$

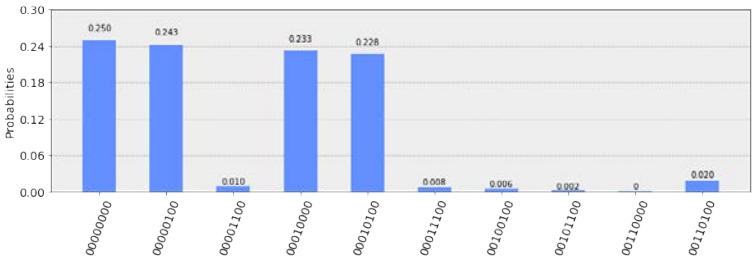

Figure 4: Results of Quantum circuit designed to find period of $7^r \bmod 55$

The results of the circuit in Figure 3 show the expected result, 20 or 00101, has a high probability of appearing on a quantum computer; however, it also shows unexpected results, 4 (0010) and 16 (00001), with close to or equal probability. It may be that because 4 and 16 equal 20, which is the desired result, 4 and 16 are equally likely as 20. To achieve just the desired result, the next circuit will be a modification of the circuit in Figure 3. Two more gates will be added to the circuit to attempt to remove 4 and 16 from appearing. The logic behind the placement of the gates is that if it is assumed that there is some quantum entanglement between superposition of the control qubits and their respective target qubits then maybe clearing the target qubits when either 4 or 16 occur individually would reduce their likelihood of appearing.

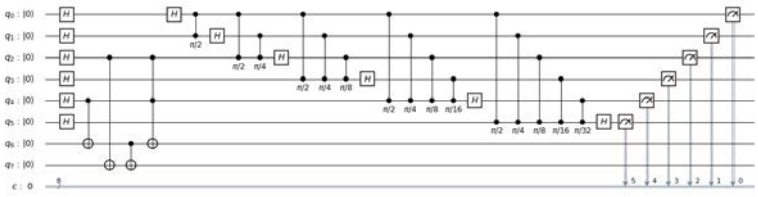

Figure 5: Modified version of the circuit in Figure 3. Added one CNOT on qubit 7 with target qubit qubit 6. Added a Toffoli Gate to Target qubit 6 with control qubits, 2 and 4.

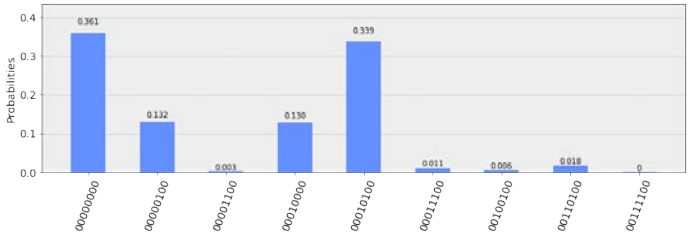

Figure 6: The results of the circuit in Figure 5 have found the desired result

The circuit yields the correct result. To examine whether this logic is exclusive to this circuit, the logical placement of the gates will be applied to a different circuit attempting to find a different period. The circuit in Figure 5 will be modified again to find the period of equation $5^r \bmod 33$ with expected period of 10.

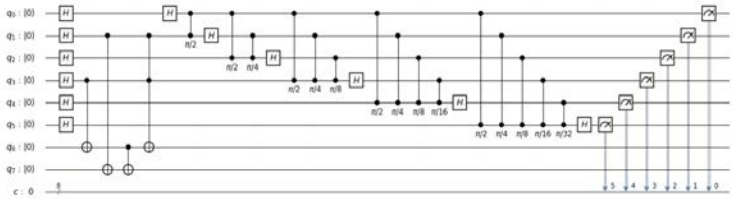

Figure 7: Shows the circuit design to find the period of $5^r \bmod 33$

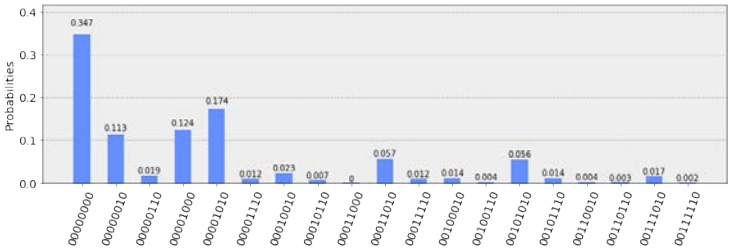

Figure 8: Results from circuit designed to find the period of $5^r \bmod 33$

The results from the second modified circuit, shown in Figure 7, show that the expected period, 10, was achieved. Ignoring 0, 10 has the second highest probability of appearing on a classical computer. This experiment shows that when considering entanglement, results from the quantum circuit can be manipulated to the desired results.

5. CONCLUSION

To conclude the results of the experiments, although not highly significant, represents an approach on how to potentially realize the modular exponentiation of Shor's period finding function. For this research quantum entanglement may be the cause of the logic of the circuit working. This highlights the fact that quantum registers are not like the registers on classical computers. Quantum Phenomena like entanglement and superposition play a role in the behavior of the circuits, meaning that a classic approach to logically understanding quantum circuit may not be the correct approach. The approach of simply storing the results on the second half of the register may not be the right approach for Shor's algorithm. Entanglement of the qubits should be taken into consideration when trying to reach Shor's Algorithm. Lastly, further research can be done to find alternate circuits that could calculate the period. As the quantum technology increases in efficiency, improvements on the circuits could yield better results and process.

REFERENCES

[1] Coles, Patrick J., et al. *Quantum Algorithm Implementations for Beginners*. Los Alamos National Laboratory, Los Alamos, New Mexico, USA, 2018. *arXiv.org*.

[2] IBM. "What is Quantum Computing?" *IBM Research - Home*, www.research.ibm.com/ibm-q/learn/what-is-quantum-computing/.

[3] Lomonaco Jr., Samuel J. "A Lecture on Shor's Quantum Factoring Algorithm." *ArXiv.org*, 9 Oct. 2000, arxiv.org/abs/quant-ph/0010034v1.

[4] Mermin, David. "Breaking RSA Encryption with a Quantum Computer: Shor's Factoring Algorithm." Physics 481-681, CS 483; Spring, 2006, Cornell University. Lecture.

[5] National Institute of Standards and Technology, et al. *Report on Post-Quantum Cryptography*. National Institute of Standards and Technology, 2016.

[6] Paar, C., and J. Pelzl. *Understanding Cryptography: A Textbook for Students and Practitioners*. Springer Science & Business Media, 2010.

[7] Shor, Peter W. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer." *SIAM Journal on Computing*, vol. 26, no. 5, 1997, pp. 1484-1509, *arXiv.org*. doi:10.1137/S0097539795293172.

[8] Vandersypen, Lieven M., et al. "Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance." *Nature*, vol. 414, no. 6866, 2001, pp. 883-887, *arXiv.org*. doi:10.1038/414883a.