LAHS71
Los Alamos High School
Elijah Pelofske
RSA Based Primality Test

Interim Report

Modern public key cryptography is based upon the difficulty of factoring large semiprimes, and the relative ease with which one can find two or more large primes. However, true primality certificates are difficult to create. For example, GIMPS is based upon the search for large mersenne primes using the lucas-lehmer primality test [5]. This test proves primality, and is much easier to perform than other tests, but only applies to mersenne primes. The most inefficient way to check primality is to simply check all factors less than sqrt(N). Currently, the most efficient true primality test is the AKS test [8]. It is very efficient in comparison to other methods, but still takes a massive amount of computing power. The Fermat primality test only proves compositeness, not primality, and given that there are an infinite number of Carmichael numbers, it is not completely trusted. Modern cryptographers use the Miller-Rabin primality test. The Miller-Rabin test so far as no counterexamples, but because the generalized Riemann Hypothesis has not been proven [9, 10, 11], the miller-rabin primality test is technically still a composite test, not primality test. The largest advantage of using the miller-rabin test is its incredible efficiency - seconds or minutes at the most. Consequently, a fully proven efficient primality test is a much desired part of number theory and computer science.

The purpose of this project is to use an asymmetric encryption method called RSA as a primality test. This RSA based primality test uses the product of one known prime and the number being tested as the modulus. There are two significant aspects of the RSA cryptosystem which could yield a type of primality test. The first is the Euler totient function, which determines the count of numbers less than N which are coprime to N [2, 4]. The second is Fermat’s little theorem, which is actually used in proving the mathematics of the RSA cryptosystem [1]. On the one hand, the use of Euler's totient function could in principle be used to prove primality, but on the other hand, fermat's little theorem has an infinite number of counterexamples. So far, the python RSA Based Primality Test [RSA_Based_Primality_Test_v4] has not found any counterexamples.

One disadvantage of using RSA is that modular exponentiation of large numbers can take a significant amount of computing power. Quantum computers are an emerging technology which can be accessed via cloud computing. Quantum computers offer the possibility of very fast modular exponentiation, along with other important algorithms. Using the IBM Quantum Experience [6], there is a possibility of creating a small scale test of an RSA based primality test on a quantum computer, coded in Open QASM 2.0 [7]. Currently, I have created a python script which simulates the output of a

quantum computer [Quantum_Simulator_v3], however, it only uses linear algebra equations based on quantum computing elementary gates [3], and does not account for quantum entanglement, resulting in a maximum of a 10% error on a particular measurement. The IBMQX5 has 16 qubits, but limited CNOT device connectivity, which limits the number of qubits can be used in logic operations at once [6]. So far, I do not have any qasm code specifically for the RSA Based Primality Test. The use of Numpy and linear algebra functions has caused my to not use Matlab, simply because it is not necessary. Additionally, I have found that while C++ gives highly efficient computing abilities, Python offers a much more manageable ability to compute large decimals and large numbers, and therefore, for the scope of this project the generality of Python is much better. The use of a quantum computer is not critical to functionality of this project, however, the ability to efficiently implement a primality test on a quantum computer would help bring the approaching era of quantum computing closer.

In conclusion, this test is still only proving compositeness, not primality, because it is not fully proven. However, so far, no counterexamples have been found. The python coded RSA based primality test has worked for all numbers up to this point. An example of this on a quantum computer could be a great demonstration of the potential of quantum computers, however it has not been coded yet in this project. The quantum simulator coded in python is not fully accurate because it does not account for quantum entanglement, but it will work as a proof of concept for the math behind quantum elementary gates.

References

[1] A Method for Obtaining Digital Signatures and Public-Key Cryptosystems
https://people.csail.mit.edu/rivest/Rsapaper.pdf

[2] An Introduction to the Theory of Numbers
http://www.fuchs-braun.com/media/532896481f9c1c47ffff8077fffffff0.pdf

[3] Elementary gates for quantum computation
https://arxiv.org/pdf/quant-ph/9503016.pdf

[4] Eulers Theorem http://sites.millersville.edu/bikenaga/number-theory/euler/euler.pdf

[5] Great Internet Mersenne Prime Search. https://www.mersenne.org/

[6] IBM Quantum Experience https://quantumexperience.ng.bluemix.net/qx/experience

[7] Open Quantum Assembly Language https://arxiv.org/pdf/1707.03429.pdf

[8] Primes in P https://www.cse.iitk.ac.in/users/manindra/algebra/primality_v6.pdf

[9] The Miller-Rabin Randomized Primality Test
http://www.cs.cornell.edu/courses/cs4820/2010sp/handouts/MillerRabin.pdf

[10] The Miller–Rabin Test
http://www.math.uconn.edu/~kconrad/blurbs/ugradnumthy/millerrabin.pdf

[11] The Rabin-Miller Primality Test
http://home.sandiego.edu/~dhoffoss/teaching/cryptography/10-Rabin-Miller.pdf