Factorization and Quantum Computing

Hector A. Del Manzano
Advisor: Dr. Lev Steinberg

Mathematics &
Electrical and Computer Engineering Department
University of Puerto Rico, Mayaguez Campus
Mayaguez, Puerto Rico 00681-5000
xzano@hotmail.com

Introduction

In 1994 Peter Shor published the first algorithm that proved the superiority of quantum computers over classical computers for solving certain types of problems. His algorithm did something that had been considered impossible for quite some time: factorizing a very big number in very little time. As we shall see, there are two main characteristics for this factorizing algorithm: (a) it is probabilistic, meaning that it can sometimes fail, and, (b) it is quantum, in the sense that the main step relies on performing operations to superposed states. To understand the impact of such achievement, we study how the quantum computer excels at factorization of large numbers and why it can be dangerous and profitable. This brings us to study in detail the most popular form of encryption - the RSA public key crypto-system- and how we can break it using Shor's factorization algorithm. To do so, we discuss an important result from number theory that relates periodic functions with factors.

1.RSA Crypto-System

The history starts in 1977 when Ronald Rivest, Adi Shamir and Leonard Adleman proposed an encryption method that could be realized without secret meetings and without having to be periodically revised for key renovation. This system has the receiver of encrypted messages broadcast openly an encryption key, that can be used by anyone to encrypt a message and that only enables the recipient to understand the contents. This organization is called the "public key" protocol. The advantages of public key protocols are the anonymity and fast exchanges. These two accomplishments are also the requirements of auspicious commerce over collective environments like telephony and the internet. Separating the intended receiver of the message and an uninvited one, is the difficult task, which is closely related with the factorization of big numbers. As we see on figure 1, which describes the RSA encryption method. If an eavesdropper can "break" our number, then he can calculate the private key the same way the true recipient does, and defeat the security of the communication channel. This is why a working implementation of Shor's Algorithm could foil most of modern communication channels, banks and governments included. But before delving into the advantages and threats of the Shor's algorithm, we introduce the proper definitions, for factorization and size, and also, we provide an encryption

example using the RSA method, detailed in figure2

1.Compute the product, *n*, of two large primes: *p* and *q*.

2.Select an integer *d*, that is coprime with (*p*-1)(*q*-1). Coprime means that the greatest common divisor between the numbers is 1.

3.Compute *e*, from the congruence:

$$ed \equiv 1 \bmod (p-1)(q-1).$$

4.Broadcast to the world {*e*, *n*}. This is the public key.

5.To send a message, the emitter represents first this message as a sequence of numbers each in the range 1 to *n*. This takes the message {**M**text} and transforms it to {**M**int}.

6.Now the emitter uses the public key to encrypt each {**M**int}. Using:

$$E_i = M_i^e \bmod n$$

*7.*The receiver can decrypt the massage using the private key which only he has. Thus,

$$M_i = E_i^d \bmod n$$

8.The final step transforms {**M**int} into the original set of {**M**text}, which can be read in plain letters.

Figure 1. RSA Encryption Scheme

Factorization:

By factorization we mean the splitting of a number into its prime factors. That is, if number *n* is divisible by an integer then the factorization of *n* means to express *n* in the following manner:

$$n = p_1 \cdot p_2 \cdot p_3 \ldots$$

where p_i are primes such that their product is *n*.

Size:

When we measure the "size" of a number, we specifically refer to the number of bits needed to store that number in a register. Size is important to us because it enables us to have a way of measuring and estimating computation time for mathematical processes running on a computer.

2.Example Using RSA Encryption Scheme

1.First the person wishing to receive the messages (us) makes the set of keys {e,n} and {d,n}.

a)We select p=9999991 and *q=9986783. n = pq = 99867740118953.*

b)We select integer *d*, such that gcd(*d*,*k*)=1, where *k*=(*p*-1)(*q*-1).

k=99867720132180,
d=5133716671

c)We compute integer e from

$$ed \equiv 1 \bmod k$$

e=1277225732

d)Now we can broadcast to the world the public key:
pub={1277225732, *99867740118953}*
keeping secret our private key:
priv= {*5133716671, 99867740118953}.*

2. If someone wished to send us a secure greeting, {'H', 'E', 'L', 'L', 'O'} for example, first we transform to numbers:

{'H', 'E', 'L', 'L', 'O'}→{8,5,12,12,15}

3.Now we encrypt using :
{
$8^{1277225732} \bmod 9986774018953, 5^{1277225732} \bmod 9...$
}

{'H', 'E', 'L', 'L', 'O'}→{big, big . . .}

4. To understand what has been received, we decrypt the set of numbers by applying:
{big, big. . .}→

{
$big_1^{5133716671} \bmod 99867740118953, big_2^{51...} ...$
.}
→{8, 5, 12, 12, 15}

5. Finally we reconvert the integers {8,5,12,12,15} to the original alphabet:
{8, 5, 12, 12, 15}→{'H', 'E', 'L', 'L', 'O'}

3.Intractability

Mathematical processes are considered intractable when the steps required to produce an output can be explicitly detailed but the number of iterations or errors grows exponentially or super-polynomially with respect to some parameter of the input. The time response of the best known (classical) factoring algorithms has the following form:

$$T(L) = \exp((\sqrt[3]{L})(\sqrt[3]{LogL}))$$

Where L is the size of the input -Hence we assume that the problem of factorization for large numbers is intractable. What this means is that when feeding a (classical) factoring algorithm it is not realistic to expect to see a correct output anytime in the near future. Illustrating the difficulty of factoring a 2000 digit number, Professor Umesh Vazirani of the University of California Berkeley said: "It's not just a case that all the computers in the world today would be unable to factor that number. It's really much more dramatic... Even if you imagine that every particle in the Universe was a computer and was computing at full speed for the entire life of the Universe, *that* would be insufficient to factor that number."

4.Shor's Factoring Algorithm - Number Theory Result

In Peter Shor's factoring algorithm we will use an important result from number theory that directly relates the period of a special function we create to the factors of a number we used in the creation of this function. Specifically we will be using the following transformation:Construct the function

$$F_n(a) = X^a \bmod n$$

where n is the number to be factored, X is a randomly chosen number that has to be coprime to n, and a is the independent variable. Coprime means that the greatest common divisor between these two numbers is 1; i.e. gcd(X, n)=1. The interesting property this new function exhibits is that it is periodic with a particular period, r. r is important to us because using the gcd operator we can get two factors of n. That is:

$\gcd(X^{\frac{r}{2}} - 1, n) = p$ and

$\gcd(X^{\frac{r}{2}} + 1, n) = q$ such that pq=n.

This is still does not solve the intractability problem because for big numbers detecting the period or repetition of outputs from the $F_n(a)$ function is difficult and time consuming as the original factorization problem. This is what we call the Conservation Of Misery principle.

Yet, the benefits of quantum computation are such that exploiting parallelism and coherence we can successfully turn this result into an efficient way of calculating the factors of a number.

5.Shor's Factoring Algorithm - The Quantum Approach

The following steps describe the application of Shor's algorithm for factoring number *n*.

1. Select number q. This number as we will see later, defines our limit in our search for factors of n. We can think of q as the scope of our search. To be efficient, we should select an integer q such that: $2n^2 \le q \le 3n^2$

2. Select a random integer *x*, so that *x* and q are coprime.

3. Repeat the following steps using the same *x*, every time.

a) Create one big quantum register and partition the q-bits in two sets. the state of the register should be $|reg1, reg2\rangle$ if reg1 describes the state of the q-bits in the first set and reg2 describes the state of the q-bits in the second set. Remember that these two partitions are part of the same register.

b) Load register 1 with a superposition of all the integers between 0 and (q-1). Load register 2 with zeros. The state of the register is now:

$$|\Psi\rangle = \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a,0\rangle$$

c) Now transform each number in register 1, using $F_n(a) = x^a \bmod n$, and place the results in register 2. The register is now described by:

$$|\Psi\rangle = \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a, x^a \bmod n\rangle$$

d) Measure the state of register 2 obtaining some result *k*. This will project register 1 into taking a superposition of just those values that $x^a \bmod n = k$. The whole register is now:

$$|\Psi\rangle = \frac{1}{\sqrt{\|A\|}} \sum_{a' \in A} |a', k\rangle$$

where A= {a : $x^a \bmod n = k$ } and $\|A\|$ is the number of elements in this set..

e) Next compute the discrete Fourier Transform of register 1. This has the effect of mapping the just projected state in register 1 into a superposition given by:

$$|\Psi\rangle = \frac{1}{\sqrt{\|A\|}} \sum_{a' \in A} \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} e^{2pa'j\frac{c}{q}} |c, k\rangle$$

f) Next we measure the state of register 1. This produces some positive integer c, where $\frac{c}{q} = \frac{l}{r}$ for some positive integer l , *r* being the desired period. What this step does is to sample the discrete Fourier transform.

g) Estimate l using a continued fraction expansion. This is done after repeating steps from (*a*) to (*f*) a few times. One samples from the transform until getting a sequence of the form $\frac{1}{r}, \frac{l_1}{r}, \frac{l_2}{r}$.... for various integers l_i. After estimating l we can guess period *r*.

4. Now that we have period *r*, we can obtain two factors of number *n* using:

$\gcd(X^{\frac{r}{2}} - 1, n) = p$ and

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

References

[1] Barenco, A., "Quantum Physics and computers", *Contemporary Physics*, Vol.37, No.5,1996

[2] Ekert, A., Jozsa, R., "Quantum Computation and Shor's factoring algorithm", *Reviews of Modern Physics*, Vol. 68, No. 3, July 1996

[3] Levine, A., "Discovering Higher Mathematics", Academic Press 2000

[4] Steane, A.M. "Error Correcting Codes in Quantum Theory", *Physical Review Letters*, Vol.77, No.5, July 1996

[4] Williams, C.P., Clearwater, S.H., "Explorations in Quantum Computing", Springer- Verlag New York, Inc 1997.