On computing \( \text{ord}_N(2) \) and its application

Lihua Liu \(^a\), Zhengjun Cao \( ^{b,c,*,1} \)

\(^a\) Department of Mathematics, Shanghai Jiao Tong University, Shanghai, China
\(^b\) Department of Mathematics, Shanghai University, Shanghai, China
\(^c\) Key Lab of Mathematics Mechanization, Academy of Mathematics and Systems Sciences, Chinese Academy of Sciences, Beijing, China

Received 10 March 2005; revised 1 December 2005
Available online 5 June 2006

Abstract

Shor proposed a polynomial time algorithm for prime factorization on quantum computers. For a given number \( N \), he gave an algorithm for finding the order \( r \) of an element \( x \) in the multiplicative group \((\text{mod} \, N)\). The method succeeds because factorization can be reduced to finding the order of an element using randomization. But the algorithm has two shortcomings, the order of the number must be even and the output might be a trivial factor. Actually, these drawbacks can be overcome in a particular case, i.e., \( N \) is an RSA modulus. In this paper, we propose a new quantum algorithm for factoring RSA modulus without the two drawbacks. Moreover, we show that the cost of the algorithm mainly depends on the calculation of \( \text{ord}_N(2) \).

© 2006 Elsevier Inc. All rights reserved.

AMS: 81P10; 11Y05; 68Q10; 03D10

Keywords: Shor’s quantum algorithm; RSA modulus; Repunit
1. Introduction

Factoring integers is generally thought to be hard on a classical computer. But it is now held that prime factorization can be accomplished in polynomial time on a quantum computer. This remarkable work is due to Peter W. Shor [1]. For a given number \( N \), he gave a quantum computer algorithm for finding the order \( r \) of an element \( x \) in the multiplicative group \((\mod N)\). The method succeeds because factorization can be reduced to finding the order of an element using randomization [2]. We now briefly give this reduction [1].

To find a factor of an odd number \( N \), given a method for computing the order \( r \) of \( x \), choose a random \( x(\mod N) \), find its order \( r \), and compute \( \gcd(x^{r/2} - 1, N) \). The Euclidean algorithm can be used to compute \( \gcd(x^{r/2} - 1, N) \) in polynomial time. Since \((x^{r/2} - 1)(x^{r/2} + 1) = x^r - 1 \equiv 0 \mod N \), the numbers \( \gcd(x^{r/2} - 1, N) \) and \( \gcd(x^{r/2} + 1, N) \) will be two factors of \( N \). This procedure fails only if \( r \) is odd, in which case \( r/2 \) is not integral, or if \( x^{r/2} \equiv -1 \mod N \), in which case the procedure yields the trivial factors 1 and \( N \). Using this criterion, it can be shown that this procedure, when applied to a random \( x(\mod N) \), yields a nontrivial factor of \( N \) with probability at least \( 1 - 1/2^k - 1 \), where \( k \) is the number of distinct odd prime factors of \( N \).

We find that in Shor’s algorithm, the order of the number must be even and the output might be a trivial factor of \( N \). Actually, these drawbacks can be overcome in a particular case, i.e., \( N \) is an RSA modulus. One phenomenon might be observed that existing prime factorization algorithms [3–8] as well as Shor’s quantum algorithm all aim to factor arbitrary numbers. None of them pays more attention to some numbers of a particular structure, especially, product of two primes. But those numbers are of great importance in public key cryptography. They are usually called RSA modulus.

In this paper, we propose a new algorithm for factoring RSA modulus on quantum computers. The algorithm overcomes the two drawbacks of Shor’s algorithm. Another special merit of the algorithm is that it only needs to compute the order of 2 relative to \( N \). But we should emphasize that computing \( \text{ord}_N(2) \) does not currently have a fast efficient classical algorithm.

2. Preliminary

Let \( N = pq \) be a product of two distinct odd primes, \( \Phi(\cdot) \) be Euler Totient Function. We know

\[
\Phi(N) = (p - 1)(q - 1) = pq - p - q + 1
\]

Hence,

\[
N - \Phi(N) + 1 = p + q
\]

Considering the following equation

\[
x^2 - Mx + N = 0 \tag{\ast}
\]

where \( M \) is undetermined, we obtain two roots

\[
x_1 = \frac{M + \sqrt{M^2 - 4N}}{2}, \quad x_2 = \frac{M - \sqrt{M^2 - 4N}}{2}
\]
If
\[ M = N - \Phi(N) + 1 \]
then equation (*) can be rewritten as
\[ x^2 - (p + q)x + pq = 0 \]
Therefore,
\[ x_1 \mid N, \quad x_2 \mid N \]
If
\[ M \neq N - \Phi(N) + 1 \]
then neither \( x_1 \) nor \( x_2 \) is an integer (since \( x_1 x_2 = pq \)).

By the above discussion, we have:

**Theorem 1.** If \( N = pq \) is a product of two distinct odd primes, and \( M \neq N + 1 \), then
\[
\frac{M + \sqrt{M^2 - 4N}}{2} \mid N \iff M = N - \Phi(N) + 1.
\]

**Proof.\; \Leftarrow\Rightarrow** It is trivial.
\[ \Rightarrow \quad \text{Since } N = pq \text{ is a product of two distinct odd primes and} \]
\[
\frac{M + \sqrt{M^2 - 4N}}{2} \mid N
\]
without loss of generality, we assume that
\[ \frac{M + \sqrt{M^2 - 4N}}{2} = p \]
Hence
\[ M + \sqrt{M^2 - 4N} = 2p \]
\[ M^2 - 4N = 4p^2 - 4pM + M^2 \]
Therefore, \( M = p + q = N - \Phi(N) + 1. \quad \square \)
3. A quantum algorithm for factoring RSA modulus

3.1. The new algorithm

Denote by $\text{ord}_N(2)$ the order of 2 relative to $N$, where $N$ is a product of two distinct odd primes. Obviously,

$$\text{ord}_N(2) \mid \Phi(N)$$

Set $s := \left\lfloor \frac{N}{\text{ord}_N(2)} \right\rfloor$, where $\lfloor x \rfloor$ denotes the integral part of number $x$. Clearly,

$$\Phi(N) \leq s \times \text{ord}_N(2)$$

Therefore,

$$\Phi(N) \in \{\text{ord}_N(2), 2 \times \text{ord}_N(2), \ldots, (s-1) \times \text{ord}_N(2), s \times \text{ord}_N(2)\}.$$ 

It is well known that $\Phi(N)$ must be kept in secret. How to search for $\Phi(N)$ in the set $\{\text{ord}_N(2), 2 \times \text{ord}_N(2), \ldots, (s-1) \times \text{ord}_N(2), s \times \text{ord}_N(2)\}$? In the following, we design a quantum algorithm based on Theorem 1, which takes advantage of the relation between computing $\Phi(N)$ and factoring $N$. The algorithm succeeds in computing $\Phi(N)$ and factoring $N$ synchronously.

A quantum algorithm for factoring RSA modulus

1. Input $N$, compute $\text{ord}_N(2)$ using Shor’s quantum algorithm.
2. $s \leftarrow \left\lfloor \frac{N}{\text{ord}_N(2)} \right\rfloor$.
3. $M \leftarrow N - s \times \text{ord}_N(2) + 1$.
4. If $M^2 - 4N$ is not a square, then $s \leftarrow s - 1$, goto step (3).
5. $t \leftarrow \frac{M + \sqrt{M^2 - 4N}}{2}$, if $t$ is not an integer, then $s \leftarrow s - 1$, goto step (3).
6. Output $t, N/t$.

How much time does this algorithm take? Apart from the time of computing $\text{ord}_N(2)$ in step (1), it seems that the running time of the algorithm mainly depends on the number of iterations of the loop, i.e., the value of $s$. In fact, it only depends on the upper bound for

$$\frac{p + q - 1}{\text{ord}_{pq}(2)}$$

If

$$\frac{p + q - 1}{\text{ord}_{pq}(2)} \leq k$$

where $k$ is an integer, then above loop will terminate in at most $k$ iterations.

Further, we know that $p$ and $q$ are usually safe primes in practice, i.e.,

$$p = 2p' + 1, q = 2q' + 1$$
where $p'$ and $q'$ are primes. Apart from the special case $\text{ord}_{pq}(2) = 4$, we have

$$k \leq \frac{p + q - 1}{p'} = \frac{2p' + 2q' + 1}{p'}$$

Usually, $p'$ and $q'$ are of the same scale. So, $k \leq 4$. This means the number of iterations in the algorithm is very small.

As for to check whether $M^2 - 4N$ is a square, easy! So, the cost of the algorithm depends almost on the calculation of $\text{ord}_N(2)$.

### 3.2. Comparison

We now make a comparison between the original algorithm and the new algorithm for factoring RSA modulus as follows.

<table>
<thead>
<tr>
<th>The original algorithm</th>
<th>The new algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Randomly choose $x$, compute the order $l$ of $x$ relative to $N$.</td>
<td>1. Compute the order $l$ of 2 relative to $N$.</td>
</tr>
<tr>
<td>2. If $l$ is not even goto step 1.</td>
<td>2. $s \leftarrow \left\lceil \frac{N}{2} \right\rceil$.</td>
</tr>
<tr>
<td>3. Compute $D_1 = \gcd(x^{l/2} + 1, N)$ and $D_2 = \gcd(x^{l/2} - 1, N)$.</td>
<td>3. $M \leftarrow N - s \times l + 1$.</td>
</tr>
<tr>
<td>4. Output $D_1, D_2$.</td>
<td>4. If $M^2 - 4N$ is not a square, then $s \leftarrow s - 1$, goto step 3.</td>
</tr>
</tbody>
</table>

Comment: It might fail to output nontrivial factors of $N$.

5. $t \leftarrow \frac{M + \sqrt{M^2 - 4N}}{2}$, if $t$ is not an integer, then $s \leftarrow s - 1$, goto step 3.

6. Output $t, N/t$.

### 4. Conclusion

It’s interesting to find that the cost of our algorithm mainly depends on the calculation of the order of 2 relative to $N$. We also observe that the number 2 is the base of binary system. How to compute the order of 2 relative to $N$ on classical computers? Given a number $x$, we all know that to compute the order of $x$ relative to an RSA modulus $N$ (whose factors are kept in secret) is hard on classical computers. How about the special number 2? We know $(2^{\text{ord}_N(2)} - 1)_2$ is a repunit, i.e.,

$$(N)_2 + (N)_2 + \cdots + (N)_2 = (\underbrace{11\cdots1}_\text{ord}_N(2))_2$$

There is only a binary additive operation in above expression. Now, what interests us is how to design a special equipment instead of a quantum computer to compute $\text{ord}_N(2)$ in polynomial time.
Acknowledgments

The authors would like to express their sincere gratitude to the referee who contributed with his expertise to the final version of the paper.

References