

Strategy For Algorithm Design in Factoring RSA Numbers

Xingbo WANG

(Department of Mechatronics, Foshan University, Foshan City, Guangdong Province, PRC, 528000)

Abstract: The article first makes an analysis on characteristic of the RSA numbers based on the factorized RSA numbers and the Digital Signature Standard; then it investigates the affection of the divisor-ratio of a RSA number to the efficiency of searching the number's divisors in term of valuated binary tree and puts forward a framework for designing algorithms of fast factoring the RSA numbers. Thoughts of the framework are presented with detail mathematical deductions.

Keywords - RSA number, Factorization, Algorithm Design, Binary Tree

I. INTRODUCTION

Ever since the RSA Laboratories declared the RSA Factoring Challenge in March 1991, factorization of the RSA numbers has become a practical criterion to test an algorithm of factoring integers and thousands of people have still kept doing the factorization in spite that the RSA Laboratories have retracted their rewards. Literatures show that, nowadays, the Number Field Sieve (NFS) is regarded as the most efficient method to factorize the RSA numbers, as introduced and surveyed in articles [1] to [7]. But unfortunately the NFS will take a vast mount of memory in the computation and it has no chance to apply the method on conventional personal computers. Hence new approaches are still under development. For example, article [8] proposed MPFA, article [9] proposed Factorization Using Multiplication Table, articles [10] proposed an formula for approximately testing divisors of a semiprime, articles [11] and [12] proposed constructing sieves of odd composite numbers. In February 2017, I introduced a new approach in article [13]. That approach can exactly locate the divisors of a composite odd number in a definite interval by means of valuated binary tree that was proposed in article [14] and can be applied in either sequential mode or parallel environment. Articles [15] and [16] followed the approach and designed their algorithms. These days, I had a look at the algorithms in articles [13], [15] and [16] and found that they might not be of high efficiency in factoring the RSA numbers although they can efficiently factorize odd composite numbers in the form $N=pq$ such that $k=q/p$ is relatively big or N has a lot of divisors.

It is obviously that one should know some internal characteristic of a RSA number before starting factoring it. Based on such a thought, this article makes a brief investigation on RSA numbers and points out a direction to design algorithm for factoring a RSA number.

II. LEMMAS

Lemma 1(See in [17]). Let p be a positive odd integer; then among p consecutive positive odd integers there exists one and only one that can be divisible by p .

Lemma 2(see in [13]). Let $n \geq 2$ and $N = p_1 p_2 \dots p_n$, where p_1, p_2, \dots, p_n are odd numbers bigger than 1; then the bigger n is, the easier N is factorized. If $n = 2$ and $p_1 < p_2$; then the bigger $\kappa = \frac{p_2}{p_1}$ is, the easier N is factorized.

Lemma 3(see in [13]). Let $N = pq$ be an odd composite number such that $2^{m+1} + 1 \leq N \leq 2^{m+2} - 1$ and $m > 2$, where p and q are odd coprime numbers that fit $3 \leq p < q$; let symbols $N_{(m+1,0)}^N$ and $N_{(m+1,2^m-1)}^N$ be respectively the leftmost and the rightmost nodes on level $m+1$ in the left branch of the n -rooted valuated binary tree T_N ; let $N_{(m+1,\omega(q))}^N$ and $N_{(m+1,\omega(p))}^N$ indicate respectively the first q 's and p 's multiple-nodes left to $N_{(m+1,2^m-1)}^N$,

$N_{(m+1,\xi(qp))}^N$ be the node that is left to and $\left\lfloor \frac{\sqrt{N} + 1}{2} \right\rfloor$ nodes away from $N_{(m+1,2^m-1)}^N$, and $N_{(m+1,2^{m-1}-1)}^N$ be the mid-node that is right to and 2^{m-1} nodes away from $N_{(m+1,0)}^N$; then it holds

(1) Nodes $N_{(m+1,2^{m-1}-1)}^N = 2^{m+1}N - 1$.

(2) There are exact $\frac{p+1}{2}$ nodes from $N_{(m+1,\omega(p))}^N$ to $N_{(m+1,2^{m-1})}^N$ and exact $\frac{q+1}{2}$ nodes from $N_{(m+1,\omega(q))}^N$ to $N_{(m+1,2^{m-1})}^N$.

(3) $p = \gcd(N, N_{(m+1,\omega(p))}^N)$ and $q = \gcd(N, N_{(m+1,\omega(q))}^N)$, where gcd means the greatest common divisor.

(4) The distribution of $N_{(m+1,0)}^N$, $N_{(m+1,\omega(q))}^N$, $N_{(m+1,2^{m-1})}^N$, $N_{(m+1,\xi(qp))}^N$, $N_{(m+1,\omega(p))}^N$ and $N_{(m+1,2^m-1)}^N$ on level $m+1$ is as figure 1 illustrates.

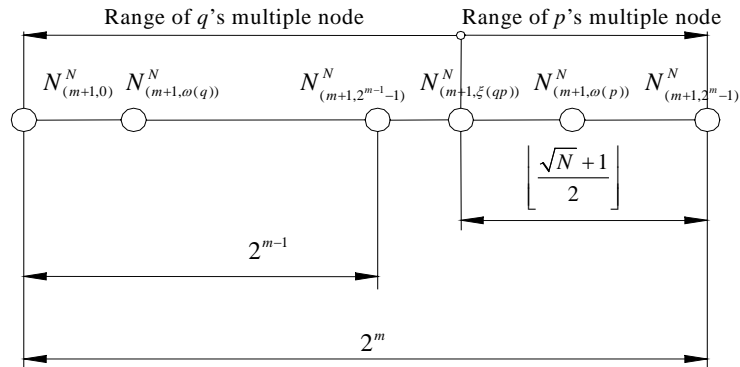


Fig.1 Distribution of Critical Nodes ($m > 2$)

Lemma 4. Let N be an integer; then it holds

$$N - \lfloor \sqrt{N} \rfloor^2 \geq 0$$

Proof. By definition of the floor function, it holds for arbitrary real number x

$$\sqrt{x} - 1 < \lfloor \sqrt{x} \rfloor \leq \sqrt{x}$$

Hence for integer N it yields $\lfloor \sqrt{N} \rfloor \leq \sqrt{N}$ and thus $N - \lfloor \sqrt{N} \rfloor^2 \geq 0$.

III. CHARACTERISTIC OF A RSA NUMBER

The RSA numbers are a set of large semiprimes. A semiprime is an odd composite number N that has exactly two prime factors, say p and q , such that $3 \leq p < q$. The two divisors of a RSA number N , p and q are required to be both safeprimes. A safeprime p is a prime that satisfies $p = 2u + 1$ and u is a prime. Hence a RSA number is a large semiprime that has two safeprimes as its divisors.

According to Lemma 1, if $N = pq$ ($3 < p < q$) is a RSA number, then $k = q/p$, which is called **divisor-ratio**, cannot be too big; otherwise it is not a safe RSA number. Actually, the Digital Signature Standard (DSS) [17] proposes that a safe k should be

$$1 < k < \sqrt{2} \tag{1}$$

We made a simple experiment to calculate the divisor-ratios of the RSA numbers that have already been factorized and found that, most factorized RSA numbers match to (1), as list in table 1.

Table 1 Divisor-ratio in Factorized RSA Numbers

No.	RSA numbers	$k = q/p$	$1 < k < \sqrt{2} ?$	$1 < k < 2 ?$	$1 < k \leq 2\sqrt{2} ?$

1	RSA100	1.056	Yes	Yes	Yes
2	RSA110	1.047	Yes	Yes	Yes
3	RSA120	2.118	No	No	Yes
4	RSA129	93.880	No	No	No
5	RSA130	1.147	No	Yes	Yes
6	RSA140	1.843	No	Yes	Yes
7	RSA150	1.281	Yes	Yes	Yes
8	RSA155	1.039	Yes	Yes	Yes
9	RSA160	1.043	Yes	Yes	Yes
10	RSA170	1.188	Yes	Yes	Yes
11	RSA180	1.190	Yes	Yes	Yes
12	RSA190	1.897	No	Yes	Yes
13	RSA200	2.244	No	No	Yes
14	RSA210	1.290	Yes	Yes	Yes
15	RSA220	2.084	No	No	Yes
16	RSA576	1.188	Yes	Yes	Yes
17	RSA640	1.163	Yes	Yes	Yes
18	RSA704	1.116	Yes	Yes	Yes
19	RSA768	1.098	Yes	Yes	Yes

Based on the above DSS and experimental results and according to Lemma 2, it is natural to draw out the following proposition.

Proposition 1. Let $N=pq$ be a RSA number; then the p 's multiple-node is quite close to the q 's multiple-node in T_N .

IV. CHARACTERISTIC OF FACTORING RSA NUMBERS WITH T_N

Note that $p \leq \sqrt{N}$ and $q \geq \sqrt{N}$; it yields

$$p < \lfloor \sqrt{N} \rfloor + 1 \text{ and } q \geq \lfloor \sqrt{N} \rfloor$$

where the symbol $\lfloor \cdot \rfloor$ means the floor function.

Since p and $\lfloor \sqrt{N} \rfloor + 1$ are integers, it further yields

$$p \leq \lfloor \sqrt{N} \rfloor \text{ and } q \geq \lfloor \sqrt{N} \rfloor \tag{2}$$

Assume $p = \lfloor \sqrt{N} \rfloor - \alpha$ and $q = \lfloor \sqrt{N} \rfloor + \beta$, where α and β are positive integers; let

$$\frac{q}{p} = \frac{\lfloor \sqrt{N} \rfloor + \beta}{\lfloor \sqrt{N} \rfloor - \alpha} = k \tag{3}$$

then

$$k\alpha + \beta = (k-1)\lfloor \sqrt{N} \rfloor \tag{4}$$

On the other hand, $N = pq$ yields

$$N = (\lfloor \sqrt{N} \rfloor - \alpha)(\lfloor \sqrt{N} \rfloor + \beta) = \lfloor \sqrt{N} \rfloor^2 + (\beta - \alpha)\lfloor \sqrt{N} \rfloor - \alpha\beta$$

Namely,

$$N - \lfloor \sqrt{N} \rfloor^2 = (\beta - \alpha)\lfloor \sqrt{N} \rfloor - \alpha\beta$$

By Lemma 3, it holds

$$(\beta - \alpha)\lfloor \sqrt{N} \rfloor - \alpha\beta \geq 0$$

which results in

$$\beta \geq \frac{\alpha \lfloor \sqrt{N} \rfloor}{\lfloor \sqrt{N} \rfloor - \alpha} \tag{5}$$

Now (4) and (5) yield

$$(k-1)\lfloor \sqrt{N} \rfloor - k\alpha \geq \frac{\alpha \lfloor \sqrt{N} \rfloor}{\lfloor \sqrt{N} \rfloor - \alpha}$$

Namely

$$k\alpha^2 - 2k\lfloor \sqrt{N} \rfloor\alpha + (k-1)\lfloor \sqrt{N} \rfloor^2 \geq 0$$

which leads to

$$\alpha \leq \left(1 - \sqrt{\frac{1}{k}}\right)\lfloor \sqrt{N} \rfloor \tag{6}$$

Or

$$\alpha \leq \left\lfloor \left(1 - \sqrt{\frac{1}{k}}\right)\lfloor \sqrt{N} \rfloor \right\rfloor = \left\lfloor \left(1 - \sqrt{\frac{1}{k}}\right)\sqrt{N} \right\rfloor \tag{7}$$

Obviously, the smaller k is, the smaller α is and thus the closer p is to $\lfloor \sqrt{N} \rfloor$. Now turn to finding p 's multiple-node on level $m+1$ in the left branch of the valuated binary tree T_N .

Let $e_{m+1}^b = N_{(m+1, 2^m-1)}^N = 2^{m+1}N - 1$, $e_{m+1}^p = N_{(m+1, \omega(p))}^N$ and $e_{m+1}^0 = N_{(m+1, \xi(qp))}^N = e_{m+1}^b - 2 \left(\left\lfloor \frac{\sqrt{N} + 1}{2} \right\rfloor - 1 \right)$; then by Lemma 3, there are $\left\lfloor \frac{\sqrt{N} + 1}{2} \right\rfloor$ nodes from e_{m+1}^0 to e_{m+1}^b and there are $\frac{p+1}{2}$ nodes from e_{m+1}^p to e_{m+1}^b ; thus there are $\left\lfloor \frac{\alpha + 1}{2} \right\rfloor + 1$ nodes from e_{m+1}^0 to e_{m+1}^p . When α takes its maximal value by $\alpha_{\max} = \left\lfloor \left(1 - \sqrt{\frac{1}{k}}\right) \sqrt{N} \right\rfloor$, it leads to the following Theorem 1.

Theorem 1. Suppose $N = pq$ is a RSA number and $k = \frac{q}{p} > 1$; Let e_{m+1}^0 and e_{m+1}^b be nodes in the valuated binary tree T_N and $\alpha_{\max} = \left\lfloor \left(1 - \sqrt{\frac{1}{k}}\right) \sqrt{N} \right\rfloor$; then it needs at most $n_k = \left\lfloor \frac{\alpha_{\max} + 1}{2} \right\rfloor + 1$ steps to find p 's multiple-node by searching from e_{m+1}^0 towards e_{m+1}^b ; and the smaller k is, the less steps are needed.

For example, when $k = \sqrt{2}$, it requires at most $\lfloor 0.08\sqrt{N} \rfloor + 1$ steps; when $k = 2$ it requires at most $\lfloor 0.15\sqrt{N} \rfloor + 1$ steps; when $k = 2\sqrt{2}$, it requires at most $\lfloor 0.21\sqrt{N} \rfloor + 1$ steps.

V. STRATEGY FOR ALGORITHM DESIGN

Theorem 1 indicates that, for a RSA number $N = pq$, starting searching e_{m+1}^p from e_{m+1}^0 towards e_{m+1}^b will take less time. This provides a direction to design algorithm. A natural and general thought is as follows.

Step 1. Constructing an right odd interval I_k (definition in [16]) that takes e_{m+1}^0 as its left-end and contains n_k nodes.

Step 2. Subdivide I_k into finite subintervals if the number of nodes in I_k is big.

Step 3. Search in the subintervals to find the p 's multiple-node until it is found.

The above thought is obviously very rough. It still needs solving some detail technical problems in a practical algorithm design. For example, the first one is how to subdivide I_k ; the second is how to efficiently search in the subintervals; and the third one is the choice of k . Since the second problem concerns with a popular technical topic, this article does not spend time discussing it and leaves it for another investigations. Here only discuss the subdivision and the k 's affection.

5.1 Subdivision of Interval

Article [16] presented an approach to subdivide the interval I_k . By that approach, I_k is divided into $2n+1$ subintervals with $2n$ equal-length subintervals and a remainder mid-subinterval, as depicted by figure 2. Then the search can be performed by first searching the mid-subinterval and then the other $2n$ symmetrically distributive subintervals. Hence it is a *middle-first subdivision*.

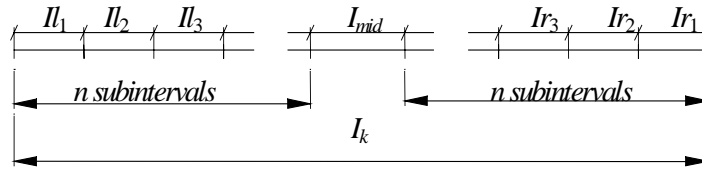
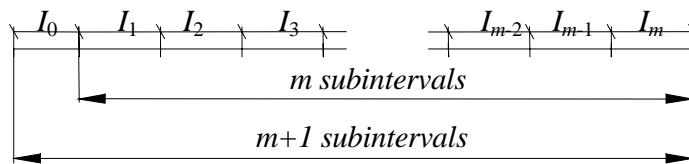


Fig. 2 A middle-first subdivision

This approach of course works well for a general parallel computation provided that I_k is properly set. On the other hand, there is a **left-first subdivision** that is better than the middle-first one when factoring a RSA number. The left-first subdivision subdivides I_k into $m + 1$ subintervals in which m subintervals are of equal length and one that is the closest to e_{m+1}^0 is not equal to the others, as depicted by figure 3.

Fig. 3 A left-first subdivision



Let M be the number of nodes in the m equal-length subintervals; let $n_0 = n_k \bmod M$ be the number of nodes in I_0 and $m = \lfloor \frac{n_k}{M} \rfloor$; then it knows that each of the m equal-length subintervals contains M nodes and it yields

$$I_0 = [e_{m+1}^0, e_{m+1}^0 + 2(n_0 - 1)],$$

$$I_j = [e_{m+1}^0 + 2jM, e_{m+1}^0 + 2(j+1)M - 2], j = 1, 2, \dots, m$$

5.2 Affection of Divisor-ratio k

As stated previously, the divisor-ratio k is also a key to determine the size of I_k . By Theorem 1, it knows that, for a RSA number $N = pq$ a smaller $k = q / p$ will make p 's multiple-node closer to $\lfloor \sqrt{N} \rfloor$ and thus it will take less time to search rightward from e_{m+1}^0 to e_{m+1}^b . Since k is unknown before N is factorized, it is necessary to make clear whatever k generally affects the algorithm design.

By Lemma 3 and Theorem 1, the interval $I_k = [e_{m+1}^0, e_{m+1}^0 + 2 \lfloor \frac{\alpha_{\max} + 1}{2} \rfloor]$, denoted by $[e_{m+1}^0, e_{m+1}^k]$, is the searching objective interval and the interval $I_{abs} = [e_{m+1}^0, e_{m+1}^b]$ is the interval that p 's multiple-node does exist. It immediately sees the following facts, as shown in figure 4.

- (i) When $k \geq 4$, $e_{m+1}^k = e_{m+1}^0 + 2(\lfloor \frac{\sqrt{N} + 1}{4} \rfloor + 1)$ lies in the middle of I_{abs} ;

(ii) When $k \geq 16$, $e_{m+1}^k = e_{m+1}^0 + 2\left(\left\lfloor \frac{\sqrt{N}+1}{8} \right\rfloor + 1\right)$ lies in the middle of right half I_{abs} ;

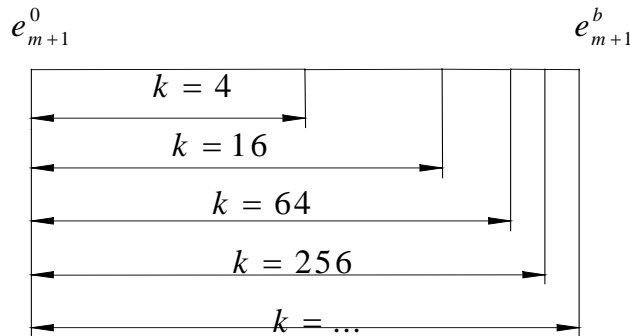


Fig. 4 k & searched interval

Consequently, it can be seen that $k = 4$ is a critical value. When $k \geq 4$, I_k exceeds half of the interval I_{abs} whose size might be very big.

On the other hand, by $p = \lfloor \sqrt{N} \rfloor - \alpha$ and (7), it is known that a bigger k will bring a bigger α and might make e_{m+1}^p closer to e_{m+1}^b since there are exact $\frac{p+1}{2}$ nodes from e_{m+1}^p to e_{m+1}^b . In fact, by $N = pq$, it is true that $p \leq \frac{\sqrt{N}}{\lambda}$ must lead to $q \geq \lambda\sqrt{N}$ for any $\lambda > 0$; therefore $k = q/p \geq \lambda^2$ if $p \leq \frac{\sqrt{N}}{\lambda}$, which says e_{m+1}^p must lie in the right half of I_{abs} when $\lambda \geq 2$ or $k \geq 4$.

5.3 Framework of Algorithm Design

Referring to the analysis on I_k 's subdivision and k 's affection to the searching intervals, a general framework for algorithm design is proposed as follows.

Framework of Algorithm Design By Using T_N

Step 1. Calculate $\alpha_{\max} = \lfloor (1 - \sqrt{1/k})\sqrt{N} \rfloor$, $e_{m+1}^b = 2^{m+1}N - 1$, $e_{m+1}^0 = e_{m+1}^b - 2\left(\left\lfloor 0.5(\sqrt{N} + 1) \right\rfloor - 1\right)$,

$e_{m+1}^\alpha = e_{m+1}^0 + 2\left\lfloor 0.5(\alpha_{\max} + 1) \right\rfloor$ and $e_{m+1}^h = e_{m+1}^0 + 2\left\lfloor 0.25(\sqrt{N} + 1) \right\rfloor + 1$;

Step 2. Construct intervals $[e_{m+1}^0, e_{m+1}^h]$, $[e_{m+1}^h + 2, e_{m+1}^\alpha]$,

Step 3. Subdivide $[e_{m+1}^0, e_{m+1}^h]$ and $[e_{m+1}^h + 2, e_{m+1}^\alpha]$ into subintervals if their sizes are big;

$[e_{m+1}^0, e_{m+1}^h] = I_l^1 \cup I_l^2 \cup \dots \cup I_l^s$; $[e_{m+1}^h + 2, e_{m+1}^\alpha] = I_r^1 \cup I_r^2 \cup \dots \cup I_r^t$;

Remark: The above framework has got only one exception RSA129 referring to Table 1.

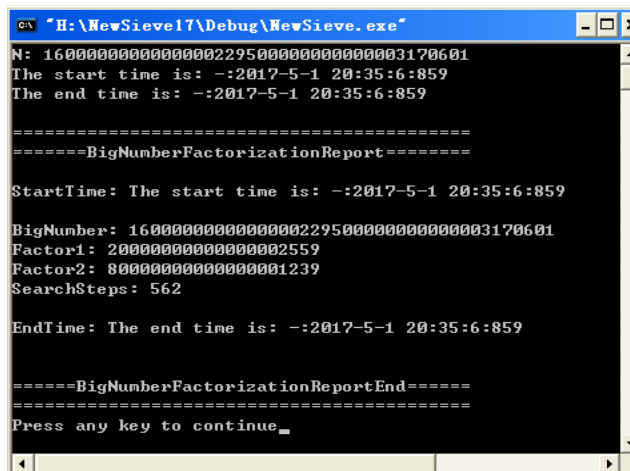
Based on the above framework, several algorithms are designed to factorize semiprimes. A most memorable sample is the factorization of a 40-digit odd semiprime

$$N=16000000000000002295000000000003170601$$

whose factorization is

$$N= 20000000000000002559 \times 8000000000000001239$$

It took only 562 steps of search to factorize it with a personal computer with an Intel Xeon E5450 CPU and 4GB memory via C++ gmp big number library, as shown in figure 5.



```
H:\NewSieve17\Debug\NewSieve.exe
N: 16000000000000002295000000000003170601
The start time is: -:2017-5-1 20:35:6:859
The end time is: -:2017-5-1 20:35:6:859

=====
=====BigNumberFactorizationReport=====

StartTime: The start time is: -:2017-5-1 20:35:6:859

BigNumber: 16000000000000002295000000000003170601
Factor1: 20000000000000002559
Factor2: 80000000000000001239
SearchSteps: 562

EndTime: The end time is: -:2017-5-1 20:35:6:859

=====
=====BigNumberFactorizationReportEnd=====
Press any key to continue_
```

Fig. 5 Screen-shot of program's running

VI. CONCLUSION

Factorization of The RSA numbers continues testing kinds of factoring algorithm. This article analyzes the characteristic of the RSA numbers in a way different from the classical ones. Like the lock and its key, a suitable key can naturally unlock a lock. I hope the analysis in this article will be the key to unlock the RSA lock.

Acknowledgements

The research work is supported by Department of Guangdong Science and Technology under projects 2015A030401105 and 2015A010104011, Foshan Bureau of Science and Technology under projects 2016AG100311, and projects 2014SFKC30 and 2014QTLXXM42 from Guangdong Education Department. The authors sincerely present thanks to them all.

REFERENCES

- [1] Dan Boneh. Twenty Years of Attacks on the RSA Cryptosystem. *Notices of the American Mathematical Society*, 46(2), 1999,203-213
- [2] Song Y. Yan. *Cryptanalytic Attacks on RSA*. Springer-Verlag New York Inc, 2008
- [3] Aoki K. Advances in Integer Factoring Technique: The Way to Factor RSA-768. *Ipsj Magazine*, 51,2010,1030-1038
- [4] Kruppa A, Leyland P. Number Field Sieve for Factoring, *Encyclopedia of Cryptography and Security*. 2011,862-867.
- [5] Wanambisi A W, Aywa S, Maende C, et al, Advances in composite integer factorization. *Materials & Structures*, 48(5), 2013,1-12.
- [6] Surhone Lambert M , Tennoe Mariam T , Henssonow Susan F. *RSA*. Betascript Publishing, 2013
- [7] Gary C. Kessler. An Overview of Cryptography, *ERAU Scholarly Commons*, 2016
- [8] Singh K, Verma R K, Chehal R. Modified Prime Number Factorization Algorithm (MPFA) For RSA Public Key Encryption. *International Journal of Soft Computing & Engineering*, 2(4),2012,1-9
- [9] Jongsoo Park,Mathology Sys, Prime Sieve and Factorization Using Multiplication Table, *Journal of Mathematics Research*, 4(3),2012,7-12
- [10] W Aldrin, Wanambisi Shem Aywa, Cleophas Maende and Geoffrey Muchiri Muketha, Factorization of Large Integers, *International Journal of Mathematics and Statistics Studies*, 1(1), 2013,39-44
- [11] Xingbo WANG. Seed and Sieve of Odd Composite Numbers with Applications In Factorization of Integers, *OSR Journal of Mathematics (IOSR-JM)*, 12(5, Ver. VIII), 2016,01-07
- [12] Xingbo WANG, Factorization of Large Numbers via Factorization of Small Numbers, *Global Journal of Pure and Applied Mathematics*, 12(6), 2016,5157-5173
- [13] Xingbo WANG, Genetic Traits of Odd Numbers With Applications in Factorization of Integers, *Global Journal of Pure and Applied Mathematics*,13(1),2017,318-333
- [14] WANG Xingbo, Valuated Binary Tree: A New Approach in Study of Integers, *International Journal of Scientific and Innovative Mathematical Research (IJSIMR)*, 4(3), 2016, 63-67
- [15] Jianhui LI.Algorithm Design and Implementation for a Mathematical Model of Factoring Integers,*IOSR Journal of Mathematics*,13(1 Ver. VI),2017,37-41
- [16] Dongbo FU, A Parallel Algorithm for Factorization of Big Odd Numbers,*IOSR Journal of Computer Engineering (IOSR-JCE)* ,19(2, Ver. V),2017,51-54
- [17] WANG Xingbo, New Constructive Approach to Solve Problems of Integers' Divisibility, *Asian Journal of Fuzzy and Applied Mathematics*, 2(3), 2014, 74-82