New Prime Factorization Algorithm and Its Parallel Computing Strategy

Lican Huang[1,2]
[1]School of Informatics, Zhejiang Sci-Tech University
[2]Hangzhou Domain Zones Technology Co., Ltd
Hangzhou, China
Huang_lican@yahoo.co.uk

Abstract—**Prime factorization is vital important for mathematics and RSA security system. In this paper we present a new algorithm to factorize numbers and a new parallel strategy for prime factorization, which may be help to find the fault of RSA system and design more safe RSA keys, and to arise the discussion about the safety of RSA security system.**

Keywords—Distribted Computation; prime factorization; parallel computing

I. INTRODUCTION

Prime factorization or integer factorization of a number is the determination of the set of prime numbers which multiply together to give the original integer. It is also known as prime decomposition. Prime factorization is vital important because RSA [1] security system is based on it. For prime factorization, polynomial time factoring algorithm on classical computers has been not found yet, however, P. Shor [4] presented a polynomial time algorithm on quantum computers. Of course, quantum computers have been not made yet, and we do not know when they are made. As for classical computations, there are several prime factorization algorithms presented such as Quadratic Sieve (QS) [5], Number Field Sieve (NFS)[2] , Pollard's p-1 method [3] , H. C. Williams's p+1 method [6], etc. There are also parallel computing methods to solve factor factorization problem such as using hadoop[7]. However, for big numbers, factorization is hard to get. In this paper, we present a novel method which constructs suitable solution tree by modulo and prunes search branches by equations set and other methods such random, and we also present a parallel strategy by using huge number of nodes.

The remainder of this paper is organized as follows: section II gives new algorithm of prime factorization; section III gives distributed parallel strategy of prime factorization; section IV presents implementation of the algorithm; finally we give conclusions.

II. NEW ALGORITHM OF PRIME FACTORIZATION

For number system base as s, number N with r digits can be written as:

$$N = n_{r-1}\ s^{r-1} + n_{r-2}\ s^{r-1} + \ldots + n_1\ s^1 + n_0 \qquad (1)$$

Suppose N can be factorized as A with p digits for base s, and B with q digits for base s, that is;

$$N = A \times B \equiv (a_{p-1}\ s^{p-1} + a_{p-2}\ s^{p-1} + \ldots + a_1\ s^1 + a_0) \times (b_{q-1}\ s^{q-1} + b_{q-2}\ s^{q-1} + \ldots + b_1\ s^1 + b_0)$$

$$\equiv a_{p-1} b_{q-1}\ s^{p+q-2} + \cdots + (a_0 b_2 + a_1 b_1 + a_2 b_0)\ s^2 + (a_0 b_1 + a_1 b_0)\ s^1 + a_0 b_0$$

$$\equiv \Sigma_{k=0.\ p+q-2}\ (\Sigma_{i+j=k}\ a_i b_j)s^k \qquad (2)$$

The algorithm has two parts:

1. First part: Calculate all combination pairs which satisfy the modulo of the digits.
   In the beginning, we calcite out all pairs which satisfies
   $a_0 b_0 \equiv n_0 \bmod s;$ (3)

   for each pair (a_{01}, b_{01}) satisfies (3) , c_{01} is the carry of pair (a_{01}, b_{01}).

   Then we can calculate all pairs which satisfies from k =1 to $k = 1 \cdots r-1\ /2$.
   $$\Sigma_{i+j=k}\ a_i b_j)s^k + C_{kl} \equiv n_k \bmod s \quad k = 1 \cdots r-1\ /2 \quad (4)$$

In this case , because we already suppose the $a_i b_j$ except $a_0 b_k$ and $a_k b_0$, the only two a_k, b_k are to be determined . We calculate all pairs which satisfy (4).

In first part the partial of A and B are equal length. We check if the partial of A and B are factors of N. If they are factors, then end. If they are not factors, then go to second part.

2. Second part : for each partial of A and B from digit length k (k= 1 … r-1 /2) , first we suppose A's digit length is k, then we can calculate B 's digit length k +m (m =1…r-2-k) which satisfy (5) from m =1.

$\sum_{i+j=k+m} a_i b_j) s^{k+m} + C_{(k+m)l} \equiv n_{k+m} \mod s \quad m = 1 \cdots r-2-k , \ i=0 \cdots k)$ (5)

We check if A and B with k+1 digit length are factors of N. If they are factors, then end, else calculate B with k+2 according (5), and check again, until B with k+ r-2-k. Then we suppose B with digit length k, and calculate A with digit length k +m (m =1…r-2-k) in the same way.

Because the algorithm is to find the path among the problem tree, and if we find the factors, the program ends, the computation time depends on that some paths can be pruned and the priority of search paths. In the algorithm, the computational time for the second part is constant due to no branch paths, where as in the first part of the algorithm, how to choose the sub-tree to search is much relative to the computational time.

We first check how to prune the branches of paths.

Suppose factor a with length la, and b with length lb.

After first part of the algorithm, we already get part of a and b with equal length lp , noted as a_0 and b_0 (here a_0 and b_0 are multi digits) . We will use the following methods to check if the part of a and b is not the genuine part of factor a and b and can be pruned.

Suppose the part of a and b are genuine, then

$(a_x\, 10^{lp} + a_0)(b_x\, 10^{lp} + b_0) = N$ (6)

$a_x b_x\, 10^{2lp} + a_0 b_x\, 10^{lp} + b_0 a_x\, 10^{lp} = N - a_0 b_0$ (7)

Select n numbers to get module , we can get :

$a_x b_x m_{i11} + b_x m_{i10} + a_x m_{i01} \equiv m_{i00} \mod (m_i)$ (i =1 … n) (8)

We can write (8) as :
$a_x b_x m_{i11} + b_x m_{i10} + a_x m_{i01} + r_i m_i = m_{i00}$ (i =1 … n) (9)

Each (9) multiply m_{j11} (except i = j) ; that is:

$a_x b_x m_{i11} m_{j11} + b_x m_{i10} m_{j11} + a_x m_{i01} m_{j11} + r_i m_i m_{j11} = m_{i00} m_{j11}$
i =1 … n , j =1 … n and $i \neq j$ (10)

Each two (10) ($i \neq j$) substrate ; that is:
$b_x (m_{i10} m_{j11} - m_{i11} m_{j10}) + a_x (m_{i01} m_{j11} - m_{i11} m_{j01}) + r_i m_i m_{j11} - r_j m_j m_{i11} = m_{i00} m_{j11} - m_{j00} m_{i11}$
That is:
$b_x (m_{i10} m_{j11} - m_{i11} m_{j10}) + a_x (m_{i01} m_{j11} - m_{i11} m_{j01}) + r_i m_i m_{j11} - r_j m_j m_{i11} = m_{ij}$

i =1 … n, j =1 … n and $i \neq j$ (11)

We have n(n-1)/2 equations as format (11) with n+2 variables .

We can use another method to prune branches.

$(a_x\, 10^{lp} + a_0 + b_x\, 10^{lp} + b_0)^2 - (a_x\, 10^{lp} + a_0 - b_x\, 10^{lp} - b_0)^2 = 4N$ (12)

$((a_x + b_x)\, 10^{lp} + (a_0 + b_0))^2 - ((a_x - b_x)\, 10^{lp} + (a_0 - b_0))^2 = 4N$ (13)

$(a_x + b_x)^2\, 10^{2lp} + 2(a_x + b_x)\, 10^{lp}(a_0 + b_0) + (a_0 + b_0)^2 - ((a_x - b_x)^2\, 10^{2lp} + 2(a_x - b_x)\, 10^{lp}(a_0 - b_0) + (a_0 - b_0)^2) = 4N$ (14)

Let $s = a_x + b_x$; $t = a_x - b_x$;

$(s)^2\, 10^{2lp} + 2(s)\, 10^{lp}(a_0 + b_0) + (a_0 + b_0)^2 - ((t)^2\, 10^{2lp} + 2(t)\, 10^{lp}(a_0 - b_0) + (a_0 - b_0)^2) = 4N$ (15)

$(s)^2\, 10^{2lp} + 2(s)\, 10^{lp}(a_0 + b_0) - (t)^2\, 10^{2lp} - 2(t)\, 10^{lp}(a_0 - b_0) = 4N - 4a_0 b_0$

$(s^2 - t^2)\, 10^{2lp} + 2s\, 10^{lp}(a_0 + b_0) - 2t\, 10^{lp}(a_0 - b_0) = 4N - 4a_0 b_0$

(16)

Select n numbers to get module , we can get:

$(s^2 - t^2) m_{i11} + s\, m_{i10} - t\, m_{i01} \equiv m_{i00} \mod (m_i)$ (i =1 … n) (17)

We can write (17) as :
$(s^2 - t^2) m_{i11} + s\, m_{i10} - t\, m_{i01} + r_i m_i = m_{i00}$ (i =1 … n) (18)

Each (18) multiply m_{j11} (except i = j) ; that is:

$(s^2 - t^2)\, m_{i11}\, m_{j11} + s\, m_{i10}\, m_{j11} - t\, m_{i01}\, m_{j11} + r_i m_i\, m_{j11} = m_{i00}\, m_{j11}$

$i = 1 \ldots n, \quad j = 1 \ldots n \quad \text{and} \quad i \neq j \qquad (19)$

Each two (19) ($i \neq j$) substrate ; that is:

$s(m_{i10}\, m_{j11} - m_{i11}\, m_{j10}) - t\,(m_{i01}\, m_{j11} - m_{i11}\, m_{j01}) + r_i m_i\, m_{j11} - r_j m_j\, m_{i11} = m_{i00}\, m_{j11} - m_{j00}\, m_{i11}$

That is:

$s(m_{i10}\, m_{j11} - m_{i11}\, m_{j10}) - t\,(m_{i01}\, m_{j11} - m_{i11}\, m_{j01}) + r_i m_i\, m_{j11} - r_j m_j\, m_{i11} = m_{ij}$

$i = 1 \ldots n, \quad j = 1 \ldots n \quad \text{and} \quad i \neq j \qquad (20)$

We have n(n-1)/2 equations as format (20) with n+2 variables .

Especially, we can use a_0 , b_0 , $a_0\, b_0$, $a_0 + b_0$, $a_0 - b_0$, a_0^2 , a_0^3 … , to get module.

So we can solve the equation set chosen from (11) or (20), if there is no solution or if the a_x digits length is not equal to la-lp, or b_x digits length is not equal lb-lp, then these parts of a and b are contract, and we can discard this branch.

III. DISTRIBUTED PARALLEL STRATEGY FOR PRIME FACTORIZATION

The algorithm in the II section is to find solution path in problem tree. The distributed parallel strategy for prime factorization is that each node computes parts of paths in problem tree. In fig.1, there is one dispatcher which keeps list of free nodes, list of jobs, and create new class of job and dispatches to the free nodes; there are lots of nodes which compute parts of paths in problem tree and report the results or their free status to dispatcher. The description java class for the algorithm is factor class in the following.

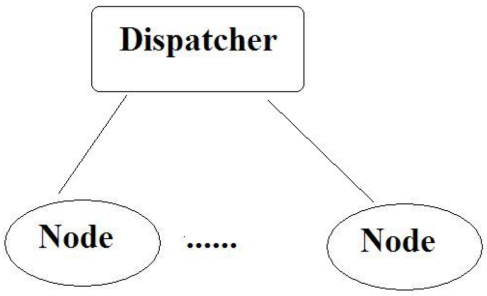

Fig.1 parallel prime factorization

```
 public class factor {
 private ArrayList<Integer>
   numberInteger ;
 private int digitnum;
 private ArrayList<Integer> factora ;
 private ArrayList<Integer>  factorb ;
 private ArrayList<Integer> carry_digit ;
 public String numberString;
 public static int klength;
 public void factoralgorithm( ) ;
 public  void reportresult();
 public  void reportpartjob();
 public  void reportfree();
 public  void receivepartjob();
 public  void Guestfromcandidate(pairab
pab);


}
```

```
 public class dispatcher {
 private ArrayList<job> jobs ;
 private ArrayList<node> freenodes ;
 public  void receivejobs();
 public  void dispatchjobs();
 public  void receiveresult();
 public  void receivefreestatus();

}
```

The node uses function Guestfromcandidate from pairset of partial a and b factors to select one pair to search and use other function reportpartjob to send jobs to dispatcher.

IV. IMPLEMENTATION OF THE ALGORITHM

We have implemented the algorithm in Java. The following is output of the factorization of number 6338502007003 .

```
Mon Jan 25 09:43:16 CST 2016prime factor
psize!168
prime factor  OK 269
prime factor X !(269,23563204487)
prime factor psize!168
actor.primes.size()23563204487)2356320448
7
actor.primes.size()168
prime factor psize!168
prime factorization number!23563204487
(1,7,0)
checkResultOK1(152981,154027)23563204487
```

```
prime factor psize!168
prime factor psize!168
prime factorization number!152981
(1,1,0)
(3,7,2)
(9,9,8)
finished!152981
prime factor psize!168
prime factor psize!168
prime factorization number!154027
(1,7,0)
(3,9,2)
finished!154027
Mon Jan 25 09:43:22 CST 2016
```

V. CONCLUSIONS

In this paper, we present a novel method which construct suitable solution tree by modulo and prunes search branches by equations set and other methods such random, and a parallel strategy using lot of nodes. By try and test, the algorithm and strategy may be potential for factorizing big numbers.

ACKNOWLEDGMENT

The paper is supported by the project “Hangzhou Qinglan Plan--scientific and technological creation and development (No.20131831K99” of Hangzhou scientific and technological committee. The software copyrights is owned by Hangzhou Domain Zones Technology Co., Ltd., and Chinese patent applied is owned by Hangzhou Domain Zones Technology Company.

REFERENCES

[1] Ron Rivest, Adi Shamir, and Leonard Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, in Communications of the ACM Vol.21 Issue.2, Feb 1978.

[2] A. K. Lenstra and H. W. Lenstra, Jr. (editors), The Development of the Number Field Sieve, Lecture Notes in Mathematics 1554, Springer-Verlag, 1993.

[3] J. M. Pollard, Theorems on Factorization and Primality Testing, Procedings of Cambridge Philosophy Society, 76 (1974), 521{528.

[4] P. Shor, Algorithms for Quantum Computation:Discrete Logarithms and Factoring , Proceedings of 35th Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, 1994,124-134.

[5] R. D. Silverman, The Multiple Polynomial Quadratic Sieve , Mathematics of Computation, 48 (1987), 329-339.

[6] H. C. Williams, A p+1 Method of Factoring", Mathematics of Computation, 39, (1982), 225-234.

[7] Son T. Nguyen, et.al. Integer Factorization Using Hadoop， 2011 Third IEEE International Conference on Coud Computing Technology and Science，2011 , 628-633