Journal of Theoretical and Applied Information Technology
15th October 2016. Vol.92. No.1
© 2005 - 2016 JATIT & LLS. All rights reserved·

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

# IMPLEMENTATION OF PARALLEL ALGORITHM FOR LUC CRYPTOSYSTEMS BASED ON ADDITION CHAIN BY A MESSAGE PASSING INTERFACE

**ZULKARNAIN MD ALI[1] AND ARNIYATI AHMAD[2]**

[1] Software Technology and Management Center, Faculty of Information Science and Technology, National University of Malaysia, 43600 Bangi, Selangor Darul Ehsan, Malaysia
[2] Department of Computer Science, Faculty of Defence Science and Technology, National Defence University of Malaysia, Kem Sungai Besi, 57000 Kuala Lumpur, Malaysia

Email: [1]zma@ukm.edu.my ; [2]arniyati@upnm.edu.my

**ABSTRACT**

The LUC cryptosystem is a modification of RSA cryptosystem. It was based on Lucas Function and has been introduced by Smith and Lennon. The computation of the LUC Cryptosystem is totally based on the computation of Lucas Function. Fast computation algorithm is required since the public key, message, primes are all big enough in order to have very secure cryptosystems. In this paper, the Addition Chain technique will be implemented for a parallel computation algorithm. In this case, the public key will be turn into the suitable array where this array will be used for computation of LUC cryptosystem based on Addition Chain. This Addition Chain will be use in manipulating the Lucas Functions properties such as $V_{2n}$, $V_{2n+1}$ and $V_{2n-1}$ to find the fast computation techniques for Lucas Functions. The capability of the standard Message Passing Interface (MPI) is implemented. The process run on special distributed memory multiprocessors machine known as Sun Fire V1280. The proposed techniques can reduce a computation time for LUC Cryptosystem computation compare to the computation algorithm for one processor. As a comparison, the computation time for one processor and several numbers of processors are also included.

**Keywords:** *Parallel Algorithm, Addition Chain, MPI, Public Key Cryptosystem.*

## 1. INTRODUCTION

Nowadays the typical desktop computer contains a multi-core processor. This parallel hardware makes the software designers reconsider the software design to get the most possible computational power from these powerful processors. This can be done by exploiting parallel programming techniques and using them to make the execution of the software components concurrent.

Parallel algorithms on the other hand play a significant role in maintaining rapid growth. Not only, multicore processors, but also a powerful graphics cards are becoming more and more available [7]

Some of the most commonly executed algorithms by computer users nowadays are cryptographic algorithms, which are used to encrypt and decrypt data in order to send it safely and securely over an unsafe environment like the internet. The well known RSA is the public key cryptosystem. It is a form of cryptography where a user has a pair of keys which are public key and a private key [1] and the RSA is probably the most promising and widely used public key cryptosystem [8].

Two researchers who is Smith and Lennon in [6] then introduced another public key cryptosystem based on Lucas Function and it is known as LUC Cryptosystem. Lucas Function also used in factoring technique designed [5].

Related discussions of LUC Cryptosystem security can be found in [1], [3] and [4]. The two primes used in LUC Cryptosystem should be big enough to ensure the security of the message. When the computation involved very big numbers of primes, the computation for encryption and decryption also required huge computation time. The research on speed up the computation of LUC Cryptosystems can be found in [9].

The ability of computation of with some number of processors is better than only use one processor. The parallel computation algorithm for RSA can be found in [2]. This kind of computation is possible for any kind of public-key cryptosystem. Some ideas in this paper gives an idea to do research in the same thrust for LUC Cryptosystems.

Therefore, a parallel computation technique will be proposed. The algorithm that is proposed will be using Addition Chain and manipulate some of the important properties of Lucas Functions.

Small example of this computation technique can be described as the following one. Let n = 14, then the sequence for a given value n is {2,0,1,0,1,0}. It is simply shown that m = 5. The first item in a sequence is not used for any computation. It is only used to indicate the end of sequence.

The sequence generated here should be used in backward {0,1,0,1,0,2}. Then, this sequence can be used in manipulating some properties of Lucas Functions. The more explanation of this sequence can be found later in this paper presentation where it is finally can be organized as Addition Chain.

Then, the implementation part is very important. The parallel algorithms are writing in C language combined with the Message Passing Interface (MPI) package. As a result, when the number of processors is increased, the computation time will be reduced.

It is also improved the efficiency of computation for LUC Cryptosystem. In this paper, the concentration is just on fast computation of the LUC cryptosystem since the security of this systems will be more concerned by other researchers. Another possible attack on this system can be found in other research area.

## 2. LUC CRYPTOSYSTEMS.

Key elements of LUC Cryptosystems are shown below:

a. Find two large primes p and q. The product of p and q is N=p∗q.
b. Choose e less than N and relatively prime to (p−1), (p+1), (q−1) and (q+1).
c. Find d, such that ed≡1(mod-(n)), where (n)=lcm(p−(D/p), q−(D/q)).
d. The public key is the pair (e, n). The public key process is done by $V_e$(P, 1)(mod N) to get C.
e. The private key is (d, n). The private key process is done by $V_d$(C, 1)(mod N) to get P.

There are two different keys that are needed for the encryption and decryption processes. The key e is publicly known and key d is remain secret. The number e must be chosen so it is relatively prime to (p − 1) (q − 1) (p + 1) (q + 1).

Therefore, the public encryption key e and the secret decryption key d are related by ed ≡ 1 (mod S (n)), where there are four possible values for the function S (n) is either lcm (p−1, q−1) or lcm (p−1, q+1) or lcm (p+1, q−1) or lcm (p+1, q+1). It is quite easy to compute least common multiple (lcm).

The plaintext (original text) denoted as M and the ciphertext denoted as C. The LUC Cryptosystems is known public key cryptosystem. It is some time called as asymmetric cryptosystems.

The concept of public key cryptosystem is very simple and prove to be more secure than the symmetric cryptosystem such as DES, 3DES and others. The public key needed two relatively primes such as p and q.

The N is the product of p and q where it is should be N=p*q. The encryption function is C = $V_e$(M,Q)(mod N). Meanwhile, the decryption function is M = $V_d$(C,Q)(mod N). We simply choose Q=1 and this apply for its computation and design. This feature has been explained in detail by Smith and Lennon [6].

Therefore, to simplified encryption and decryption of LUC Cryptosystem, the encryption supposed to be C=$V_e$(M,1)(mod N) and the decryption supposed to be M=$V_d$(C,1)(mod N).

Consider the situation when Bob and Alice communicated on an insecure channel:

- To send a message P to Bob, Alice uses Bob's public key e to compute, C = Ve(P,1) (mod N).
- Bob recover the plaintext P from C with his private key d by computing P = $V_d$(C, 1) (mod N).

Computation of $V_e$ and $V_d$ might look extremely long, for large values of e and d. This is where we need parallel algorithm in order to make the computation really fast.

## 2.1 LUCAS FUNCTIONS.

Let α and β be the roots of the polynomial equation $x^2 - Px + Q = 0$. Then P=α+β and Q=αβ. The general second order linear recurrence:

$$U_n = (\alpha_n - \beta_n)/(\alpha - \beta) \quad (1)$$
$$V_n = \alpha_n + \beta_n \quad (2)$$

There are two function that can be derive from Equation (1) and (2). It is:

$$U_{n+1} = PU_n - QU_{n-1} \quad (3)$$
$$V_{n+1} = PV_n - QV_{n-1} \quad (4)$$

As mentioned in [6], the sequence $V_n$ with Q = 1 is usually used to design LUC cryptosystem. Then, the Equation (4) can simply be derived as,

$$V_n = PV_{n-1} - V_{n-2} \quad (5)$$
where n≥2, $V_0$=2 and $V_1$=P

Some equations are related and to be used with Addition Chain technique. There are:

$$V_{2n} = V_{n2-2} \quad (6)$$
$$V_{2n+1} = PV_{n2} - V_nV_{n-1} - P \quad (7)$$
$$V_{2n-1} = V_nV_{n-1} - P \quad (8)$$

Simple example of LUC cryptosystems computation is shown in the following details. In this example, P = 11111, p = 1949, q = 2089 and e = 1103.

1. The computation of ciphertext C is as follows:

a. Let N = p*q = 1949*2089 = 4071461 and P = 11111, a plain text.
b. To encrypt P = 11111, calculate $V_{1103}$(11111, 1)(mod 4071461) and the result is C = 3975392.
c. Then Q = 1, $V_0$ = 2, $V_1$ = 11111 and need to calculate $V_{1103}$.
d. First, calculate $V_2 = PV_1 - QV_0$ (mod 4071461).
e. Second, calculate $V_3 = PV_2 - QV_1$ (mod 4071461).
f. Continues, until $V_{1103} = PV_{1102} - QV_{1101}$(mod 4071461).
g. The final value $V_{1103}$ is the required value.
h. Therefore at this point C = $V_{1103}$ where <u>C = 3975392.</u>

2. The calculation of private key d is as the following steps:

a. C = $V_{1103}$=3975392.
b. Let D = $C^2 - 4$; then (D/1949) = −1 and (D/2089)=−1 are the two Legendre Symbols.
c. Calculate r. r is actually the Least Common Multiple of 1949+1 and 2089+1. Then, r = lcm(2.3.52.13, 2.5.11.19) = 407550.
d. Use the Extended Euclid Algorithm to find the secret key d (decryption key). The public-key e = 1103, by solving the modular equation ed = 1(mod 407550). Finally, <u>d = 24017</u>.

3. The decryption of the ciphertext can be in the following details:

a. Calculate $V_d$(C, 1)(mod N).
b. It means that $V_{24017}$(3975392,1)(mod 4071461)=1111.
c. Initial values are $V_0$ = 2, $V_1$ = 3975392 and need to calculate $V_{24017}$.
d. First, calculate $V_2=CV_1 - QV_0$ (mod 4071461).
e. Second, calculate $V_3= CV_2 - QV_1$ (mod 4071461).
f. Continues, until $V_{24017}= CV_{24016}-QV_{24015}$(mod 4071461).

g. The final value $V_{24017}$ is <u>P = 11111</u>.

Then the encryption and decryption processes are done.

## 3. BASIC ASPECTS OF PARALLEL COMPUTATION

### 3.1 ADDITION CHAIN

**Theorem 1**

Given an integer n, a sequence for n is $\{a_0, a_1, a_2 \dots, a_m\}$ such that $a_0 = 2$, $a_1 = (0 \text{ or } 1)$, $a_2 = (0 \text{ or } 1)$ and $a_m = 0$. The reverse of sequence can be used to form an Addition Chain.

**Proof**
Initial value is $a_0 = 2$. For $a_1$ until $a_{m-1}$. Let w = n mod 2, if w = 1 then $a_1 = 1$ and $n = n-1$. Otherwise, if w = 0 then $a_1 = 0$ and $n = n/2$. The value of n for next computation is either $n = n - 1$ or $n = n/2$ and this computation continues until $n = 1$.

It is shown in Theorem 1 that the first value of that sequence, $a_0 = 2$ is only used as the indication to stop the computation. The following definition is very important to show the possible size of sequence that could be generated for the Addition Chain.

Let n = 14, then the sequence for a given value n is {2, 0, 1, 0, 1, 0}. It is simply shown that m = 5. The first item in a sequence is not used for any computation. It is only used to indicate the end of sequence. The sequence generated here should be used in backward {0, 1, 0, 1, 0, 2}.

Based on the reverse sequence {0, 1, 0, 1, 0, 2}, the computation of Lucas Functions is 1, 1+1 = 2, 2+1 = 3, 3+3 = 6, 6+1 = 7, 7+7 = 14 and 1, 2, 3, 6, 7, 14 is Addition Chain with the length of 5.

Therefore, the computation of LUC Cryptosystem with an Addition Chain is $V_2$, $V_3$, $V_6$, $V_7$, and $V_{14}$. These computations should be deal with Equations (6), (7) and (8) where it is needed.

```
Input : n
   k[0] = 2
   m = 0
While (n! = 1)
   m + +
   If (n mod 2) == 1
      n = n − 1
      k[m] = 1
   Else
      ⌊n⌋ = n/2
      k[m] = 0
   End If
End While
Output : Array k[0, 1, ...,m]
```

*Algorithm 1: Generating an array k for Addition Chain*

For a bigger example, an array k[m] for $V_{1103}$ will be generated. In this case n = 1103. An array k[m] = {2, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0} is generated. In this case, m = 15. The array k[m] in reverse manner could be {0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 2}. Therefore, the Addition Chain for n = 1103 is 1, 2, 4, 8, 16, 17, 34, 68, 136, 137, 274, 275, 550, 551, 1102, 1103.

Remember, the array k[m] that was generated in Algorithm 1 is just an array. The array k[m] was only the suitable Addition Chain that could be used for LUC cryptosystems computation. For sure it is not the optimum Addition Chain.

The computation of LUC cryptosystem is started with $V_2$ and followed by $V_4$, $V_8$, $V_{16}$, $V_{17}$, $V_{34}$, $V_{68}$, $V_{136}$, $V_{137}$, $V_{274}$, $V_{275}$, $V_{550}$, $V_{551}$, $V_{1102}$ finally end with computation of $V_{1103}$.

The following figure show how the array k[m] is generated by Algorithm 1. It is depend totally on Theorem 1 and Algorithm 1 that was discussed above.

| k[m] | Value | Description |
|---|---|---|
| k[0]=2 | 1103 | n=n; m=0 |
| k[1]=1 | 1103 mod 2 = 1; 1103-1=1102 | n=n-1 ; m=1 |
| k[2]=0 | 1102 mod 2 = 0; 1102/2=551 | n=n/2 ; m=2 |
| k[3]=1 | 551 mod 2 = 1; 551-1=550 | n=n-1; m=3 |
| k[4]=0 | 550 mod 2 = 0; 550/2=275 | n=n/2; m=4 |
| k[5]=1 | 275 mod 2 = 1; 275-1=274 | n=n-1; m=5 |
| k[6]=0 | 274 mod 2 = 0; 274/2=137 | n=n/2; m=6 |
| k[7]=1 | 137 mod 2 = 1; 137-1=136 | n=n-1; m=7 |
| k[8]=0 | 136 mod 2 = 0; 136/2=68 | n=n/2; m=8 |
| k[9]=0 | 68 mod 2 = 0; 68/2=34 | n=n/2; m=9 |
| k[10]=0 | 34 mod 2 = 0; 34/2=17 | n=n/2; m=10 |
| k[11]=1 | 17 mod 2 = 1; 17-1=16 | n=n-1; m=11 |
| k[12]=0 | 16 mod 2 = 0; 16/2=8 | n=n/2; m=12 |
| k[13]=0 | 8 mod 2 = 0; 8/2=4 | n=n/2; m=13 |
| k[14]=0 | 4 mod 2 = 0; 4/2=2 | n=n/2; m=14 |
| k[15]=0 | 2 mod 2 = 0; 2/2=1 | n=n/2; m=15 |

*Figure 1 : Illustration of generating an Array k[0, 1, ..,m] for $V_{1103}$*

```
Input : k[0, 1, ...,m] (Algorithm 1), P, N, $V_0$=$V_j$= 2,
$V_1$=$V_n$= P;
For (z=m downto 0 )
If (k[z]==0)
   $V_{2n} = V_{2n-2}$ (mod N)
   $V_{2n+1} = PV_{2n} - V_nV_j - P$ (mod N)
   $V_{2n-1} = V_nV_j - P$ (mod N)
   $V_n = V_{2n}$
   $V_j = V_{2n-1}$
Else
   $V_n = V_{2n+1}$
   $V_j = V_{2n}$
End If
End For
Output : $V_n$
```

*Algorithm 2 : Computation Algorithm Based on Addition Chain With 1 Processor*

The result of the illustration in Figure 1 can be used in designing the computation for one processor. It is relatively easy for computation on one processor by directly implement the algorithm. It is also can be used for several processors in parallel computation for LUC cryptosystem. The algorithm 2 shows the computation of LUC cryptosystem for one processor.

## 3.2 PARALLEL STRATEGIES FOR ADDITION CHAIN

In the design of parallel computation algorithm, the algorithm must be suitable for the architecture of Message Passing Interface (MPI) library. This library is suitable in this research. This library is suitable for the use of programming code in C language. Initialization can use the command of MPI_Init() to initiate the using of MPI standard.

Parallel computing frequently relies upon message passing to exchange information between computational units. In high performance computing, the most common message passing technology is the Message Passing Interface (MPI).

MPI enables the description of different communication patterns that enable the most efficient usage of resources, from an inter-process communication perspective, to optimize the performance of a High Performance Computing cluster.

The initial value of Lucas Function are $V_0 = 2$ and $V_1 = P$. Look at Equations (6), (7) and (8). All this equations will be using in computing the LUC cryptosystems. This is a crucial and important feature that should be considered in the parallel algorithms.

The general strategies are:

a. Given public-key, e, then generate an array of sequence of array k using Algorithm 1.
b. Reverse the sequence because it represented exactly the Addition Chain.
c. Use each item in the Addition Chain in the calculation of LUC Cryptosystem.
d. Synchronizations of movement of data are done by master.
e. Master or slaves will decide which value should be used for the next computation. The nature of Lucas Function works by recursive. Therefore, the next values for the next computations are determined by the checking procedure. In this case, the parallel computation algorithms have a checking procedure in master and also in all slaves.

Algorithm 3 show different techniques and strategies in computing the LUC Cryptosystems. In this parallel solution, the value of $V_n$ is either $V_{2n+1}$ or $V_{2n}$ or $V_{2n-1}$. It is also applied to $V_j$ , where it is either $V_{2n+1}$ or $V_{2n}$ or $V_{2n-1}$.

The values to be assigned to $V_n$ and $V_j$ were depending on the value of array k[m]. The number of slaves can be added depending on how to decompose the function.

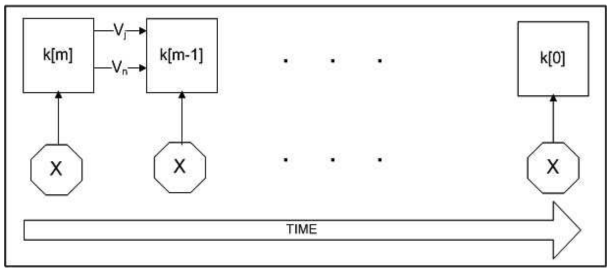

*Figure 2 : General Idea For Parallel Algorithms Based On Addition Chain*

General idea on how to use Addition Chain for this parallel algorithm is shown in Figure 2. Symbol 'X' represents parallel algorithm for specific number of processors. In all parallel implementation, one processor should act as master, while the others as slaves. On the other hand, two variables $V_j$ and $V_n$ are used to keep the values that should be used for next computation.

The decomposition technique could be very clear in Figure 3 below. In all parallel implementation, one processor should act as master, while the others as slaves. On the other hand, two variables $V_j$ and $V_n$ are used to keep the values that should be used for next computation. The computation time of generating Addition Chain is included in this parallel algorithm.

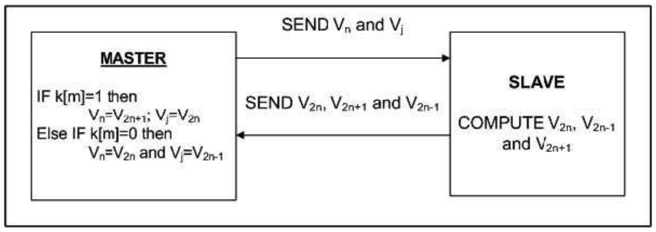
SEND $V_n$ and $V_j$
MASTER
IF k[m]=1 then
$V_n=V_{2n+1}$; $V_j=V_{2n}$
Else IF k[m]=0 then
$V_n=V_{2n}$ and $V_j=V_{2n-1}$
SEND $V_{2n}$, $V_{2n+1}$ and $V_{2n-1}$
SLAVE
COMPUTE $V_{2n}$, $V_{2n-1}$ and $V_{2n+1}$

*Figure 3 : Parallel Strategy Based On Addition Chain With 2 Processors*

Figure 3 shows possible arrangement of functional decompositions with two processors. Here, one processor as the master and the other as the slave. The function of master processor is to synchronize the movement of data. All the computation job is done by slave.

Algorithm 3 is designed for two processors, where one as master and the other one as slave. The master will synchronize the job. The possible decomposition could involved the master and some slaves. Therefore, for the bigger number of processors, the reorganization of the steps in 'At Slave' to suit to the number of slaves (processors).

The numbers of slaves could be two, three, four, five and six. We only included the best performance of computations with six slaves. It means that the total numbers of processor should be seven processors. In this paper, the parallel machine that was selected is Sun Fire V1280 server. It is the example of the distributed memory multiprocessors machine.

```
MPI Initialization
Input: Array k[0, 1, ...,m] (Algorithm 1), P, N, $V_0 = V_j = 2$, $V_1 = V_n = P$

For (z=m downto 0 )

At master :
    If k[z] = 1 then
        $V_n = V_{2n+1}$ and $V_j = V_{2n}$
    If k[z] = 0, then
        $V_n = V_{2n}$ and $V_j = V2_{n-1}$
    Send $V_n$ and $V_j$ to Slave
    Receive $V_{2n}$, $V_{2n+1}$ and $V_{2n-1}$ from Slave

At slave :
    Receive $V_n$ and $V_j$ from master
    Compute $V_{2n} = V_{2n} - 2 (\mathrm{mod}\ N)$
    Compute $V_{2n+1} = PV_{2n} - V_nV_j - P (\mathrm{mod}\ N)$
    Compute $V_{2n-1} = V_nV_j - P (\mathrm{mod}\ N)$
    Send $V_{2n}$, $V_{2n-1}$ and $V_{2n+1}$ to master

End For
Final Result, $V_n$
MPI Finalization
```

*Algorithm 3 : Parallel Computation Algorithm Based On Addition Chain With 2 Processors*

All functional decompositions involved master and slaves. Therefore, for the bigger number of processors, the reorganization of the steps in 'At Slave' to suit to the number of slaves (processors). For example, lets have a look at Equation (7).

This equation can be decomposed into smaller equations and each portion of equation will perform by the slave(s). One portion of equation refers to one task.

To calculate $V_{2n+1}$, perhaps one slave can perform the calculation of $PV_{2n}$ and at same time another slave will calculate $QV_nV_{n-1}$. It is actually the functional decomposition of big task into smaller task. After smaller task done their computation jobs, it will send back the result to master.

Master will responsible to synchronize the next value for the next computation. If there are more slaves, Equation (7) can be distributed to several slaves. Master and slaves should communicate to each other to make sure all entities do the right computation jobs.

With this functional decompositions arrangement, the reduction of the computation time compared to only one processor. This machine is designed to provide high performance tool in a compact form. It is clear that up to 7 processors is the best and suitable numbers that can be used for the parallel computation of the LUC cryptosystems

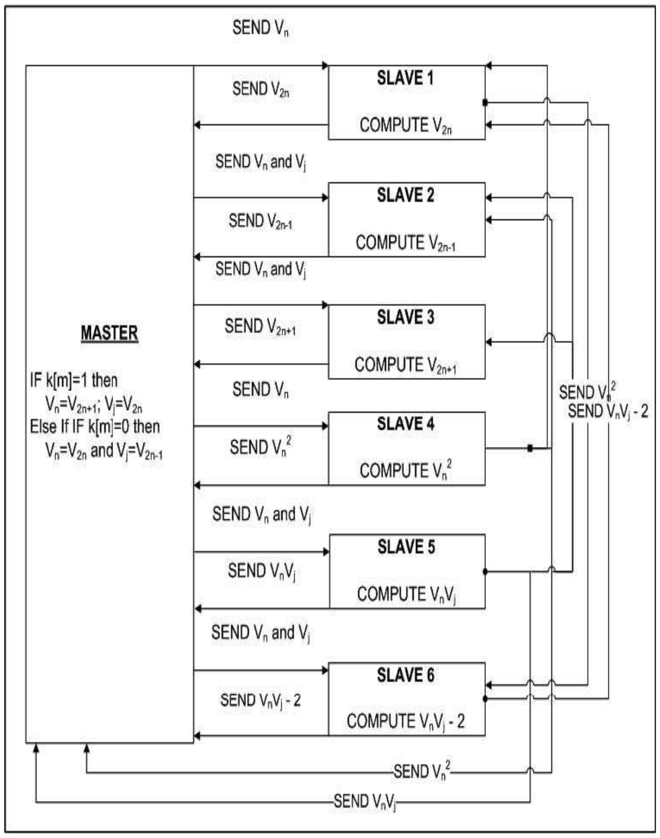

*Figure 4 : Parallel Strategy Based on Addition Chain with 7 Processors*

```
MPI Initialization
Input: Array k[0, 1, ...,m] (Algorithm 1), P, N,
$V_0 = V_j = 2$, $V_1 = V_n = P$

For (z=m downto 0 )

At master :
   If k[z] = 1, then
      $V_n = V_{2n+1}$ and $V_j = V_{2n}$
   If k[z] = 0, then
      $V_n = V_{2n}$ and $V_j = V_{2n-1}$
   Send $V_n$ to Slave 1 and 4
   Send $V_n$ and $V_j$ to Slave 2, 3, 5 and 6
   Receive $V_{2n}$ from Slave 1 and $V_{2n-1}$ from Slave 2
   Receive $V_{2n+1}$ from Slave 3 and $V_{n2}$ from Slave 4
   Receive $V_nV_j$ from Slave 5 and $V_nV_j - 2$ from Slave 6

At Slave 1 :
   Receive $V_n$ from master and $V_nV_j - 2$ from Slave 6
   Receive $V_{n2}$ from Slave 4
   Compute $V_{2n} = V_{2n} - 2 (\mathrm{mod}\ N)$
   Send $V_{2n}$ to master and Slave 6

At Slave 2 :
   Receive $V_n$ and $V_j$ from master
   Receive $V_{n2}$ from Slave 4 and $V_nV_j$ from Slave 5
   Compute $V_{2n-1} = V_nV_j - P (\mathrm{mod}\ N)$
   Send $V_{2n-1}$ to Master

At Slave 3 :
   Receive $V_n$ and $V_j$ from master and $V_nV_j$ from
   slave 5
   Compute $V_{2n+1} = PV_{2n} - V_nV_j - P (\mathrm{mod}\ N)$
   Send $V_{2n-1}$ to Master

At Slave 4 :
   Receive $V_n$ from master
   Compute $V_{n2}$
   Send $V_{n2}$ to Master, Slaves 1 and 2

At Slave 5 :
   Receive $V_n$ and $V_j$ from master
   Compute $V_nV_j$
   Send $V_nV_j$ to Master, Slaves 2 and 3

At Slave 6 :
   Receive $V_n$ and $V_j$ from master and $V_{2n}$ from
   Slave 1
   Compute $V_nV_j - 2$
   Send $V_nV_j - 2$ to Master and Slave 1

End For
Final Result, $V_n$
MPI finalization
```

*Algorithm 4 : Parallel Computation Algorithm Based on Addition Chain With 7 Processors.*

The main reason of the checking procedure by the master is to make sure the master will sending the right value to the slave. Each item in array k[m] is useful in determining the value to be used in subsequent calculations. It is clearly shown that this parallel algorithm has the checking procedure in master and all slaves.

If the item of array k[m] is 1 then the next value for subsequent calculations is $V_{2n+1}$ and $V_{2n}$. On the hand, if the item is 0, then the next value for subsequent calculations is $V_{2n}$ and $V_{2n-1}$.

## 4. RESULTS

Performance of parallel algorithms can be determined by its computation time. The execution of computer codes in C programming language with support of Message Passing Interface standard to solve two different situations on different size of keys and primes.

The capability of the algorithm can be determined by running the experiments on different sizes of keys and primes. Different key size must produce different computation time. Table 1 shows different size of keys. Meanwhile,

Table 2 shows different size of primes. The public key with size of 579 digits will generate longer array k[m]. Therefore, the public key with size of 579 digits required huge efforts and computation time

| Public Key e | Primes p and q | Messages M |
|---|---|---|
| 159 | 100 | 5 |
| 339 | 100 | 5 |
| 579 | 100 | 5 |

*Table 1 : Different size of public keys (Digits)*

| Primes p & q | Public Key e | Messages M |
|---|---|---|
| 160 | 159 | 20 |
| 220 | 159 | 20 |
| 280 | 159 | 20 |

*Table 2 : Different Size of Primes (Digits)*

Table 3 shows the comparison of the encryption computation time for the number of processors. The bigger the public key size, the longer computation time is produced for LUC Cryptosystem computation. It is really clear to show that the best computation time with seven processors.

The bigger the number of processors, the better computation time is produced. In all cases, the parallel computation algorithm with seven processors distinctly better computation time compared the smaller number of processors.

| Public Key e (digits) | 1 Processor (second) | 2 Processors (second) | 7 Processors (second) |
|---|---|---|---|
| 159 | 139.88 | 132.86 | 45.26 |
| 339 | 489.74 | 473.15 | 160.55 |
| 579 | 869.60 | 839.38 | 289.25 |

Table 3 : Computation time on different size of public keys

The same situation happened for the different size of primes. Table 4 shows each computation time for number of processors where the computation with seven processors is achieved the best computation time. The bigger size of primes, the longer computation time is a need. Remember that, in this computation experiment, there are the same size of public key and message is used but the sizes of primes are changed.

| Public Key e (digits) | 1 Processor (second) | 2 Processors (second) | 7 Processors (second) |
|---|---|---|---|
| 160 | 358.03 | 353.43 | 120.94 |
| 220 | 599.32 | 571.32 | 195.72 |
| 280 | 850.02 | 819.32 | 263.44 |

*Table 4 : Computation time on different size of primes*

The primary issue with speedup is the communication to computation ratio. To get a higher speed up, the parallel design should be considered and tried to have less communications, make connections faster, communicate faster and each slave can compute more task.

Unfortunately, this paper will not discuss the speedup and communication speed issues. However, we just discussed simple term in achieving good speedup.

The less communication between master and slaves in the proposed parallel algorithm should be designed. The reduction of redundant computation and increase the communication between slaves and master should be considered.

Please remember that the simple formula to calculate speedup is given by $S = T_1/T_n$. Here, $T_1$ is the computation time to simulate the problem on one processor. $T_n$ is the computation time to simulate the problem on n processors.

The rest of the listed features show a totally different approach and strategy in designing parallel algorithms:

1. Length of an array k. The PCABAC has an array with the size of m.
2. Computation of LUC Cryptosystem. The PCABAC will compute all $V_{2n}$, $V_{2n-1}$,$V_{2n+1}$ for each k[m]=1 in slaves only, then send the results to master. Master will only initiate next value for each k[m] = 0.
3. Distribution of computation jobs among the processors. The algorithm distribute the computation jobs depending on the equations used in Lucas Function. One processor will act as the master that will synchronized the movement of data. The less distribution of computation jobs will result in a better computation time. Less distribution of computation jobs will result in a less communication delay between master and slaves.
4. Maximum number of processors. The algorithm require seven processors as the possible maximum number of processors. Each processor is used in different strategy.
5. Checking the next value to be used. The nature of Lucas Functions work by recursive manner. The next value to be used for the next computation should be known and initiated. The parallel algorithms require two initial values ($V_0$ and $V_1$) that could be used in LUC computations. The algorithm has the checking procedure that is conducted by master.
6. Communications of slaves and master. Less communications will result in a better computation time. The communication is important in order to update the current values to be used in the computation.

## 5. CONCLUSIONS

The parallel computation algorithms are successfully implemented in the distributed memory multiprocessor machines. The computation of LUC Cryptosystem shows that the parallel algorithm will compute all $V_{2n}$, $V_{2n-1}$ and $V_{2n+1}$ for each k[m]=1 in slaves only, then send the results to master. Master will only initiate the next value for each k[m]=0. Master and each slave then check the next values to be used.

The Parallel Computation Based on Addition Chain (PCBAC) shows the best solution for distribution of computation jobs among slaves. All algorithms require seven processors as the possible maximum number of processors. Each processor is used in different strategies. The strategy used is already shown in each parallel algorithm.

The nature of Lucas Function works by recursive manner. Therefore, the next values for the next computations are determined by the checking procedure. In this case, PCBAC has the checking procedure done by master only.

Last but not least for the speedup and efficiency, the PCABB algorithm shows deliberately the better speedup and efficiency compared to the existing parallel algorithms for all experiments.

In this paper, one new parallel algorithm has been carried out for solving the time domain for LUC Cryptosystems using MPI on Sun Fire V1280. Performances for algorithm are studied. The computation time for a new algorithm and existing algorithms is also studied. From the results, it is observed that a new parallel implementation provides significant reduction in computational time compared to the previous parallel algorithm.

The efficiency of this algorithm is increased when the number of processors is increased. The reduction of a communication delay among master and slaves should be considered for further investigation. Another interesting research topic for consideration is how to reduce the arithmetic computations of Lucas Function.

The parallel implementation on seven processors shows better computation time for all experiments. The distribution of computation jobs shows that all algorithms successfully the computation jobs depending on the equations used in Lucas Function. The lesser the distribution of computation jobs the better computation time.

Lesser distribution of computation jobs will reduce communication delay between master and slaves. Each processor is used in different strategies. The strategy used is already shown in each parallel algorithm.

## REFERENCES:

[1] B. Schenier, Applied Cryptography (Protocols, Algorithms and Source Code in C), Second Edition, John Wiley & Sons Inc, 1996.

[2] C. K. Koç, High Speed RSA Implementation, Technical Report, RSA Laboratories, (RSA Data Security INC, CA 1994).

[3] C. S. Laih, F.K. Tu and W.C Tai, Remarks on LUC public-key system, Electronic Letters, Vol 30, No. 2, 1994, 123-124.

[4] D. Bleichenbacher, M. Joye and J.J. Quisquater, A New and Optimal Chosen-message Attack on RSA-Type Cryptosystems, Y. Han, T. Okamoto and S. Qing (Eds), Information and Communications Security ICICS97, LNCS 1334, Springer-Verlag, 1997, 302-313.

[5] H. C. Williams, A ρ+1 Method of Factoring, Mathematics of Computation, vol.39, 1982, 225-234.

[6] P. Smith and M. Lennon, LUC: A New Public Key System, Ninth IFIP symposium on computer security, E.G. Douglas, Ed, Elsevier Science Publishers, 1993, 103-117.

[7] P. Lara, F. Borges, R. Portugal and N. Nedjah, Parallel modular exponentiation using load balancing without pre computation, Journal of Computer and System Sciences, Vol.78, No.2, pp. 575–582, 2012.

[8] R. L. Rivest, A. Shamir and L.M. Adleman, A Method for Obtaining digital signatures and public-key Cryptosystems, Comm, ACM 21, 1978, 120-126.

[9] Z. Md Ali, Reduce Computation Steps Can Increase the Efficiency of Computation Algorithm, Journal of Computer Science 6(10), Sciences Publication, 2010, 1203-1207.