Journal of Theoretical and Applied Information Technology
15th October 2012. Vol. 44 No.1

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

PARALLEL COMPUTATION ALGORITHM FOR LUC CRYPTOSYSTEM BASED ON BINARY NUMBER

ZULKARNAIN MD ALI

Senior Lecturer, School Of Computer Science,
Faculty of Technology and Information Science (FTSM),
Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia.

E-mail: zma@ftsm.ukm.my

ABSTRACT

LUC Cryptosystem is a public key cryptosystem based on Lucas Function. It is first discussed by Smith and Lennon in 1993. They proposed a new public key system using Lucas Function instead of using exponentiation based as found in RSA. Lucas Function is the second order linear recurrence relation. The computation of LUC Cryptosystem is based on the computation of Lucas Function. Many of the existing computation algorithms for Lucas Function are suitable for one processor and there is no problem to design a computation algorithm for one processor as the Lucas Function can be implemented directly into programming codes. In this paper, the Binary Numbers will be used as a technique for parallel computation algorithm. The encryption process using $V_e(P,1)(\text{mod } N)$ to get ciphertext, C from plaintext, P. While the decryption used $V_d(C,1)(\text{mod } N)$ to get P from C. Meanwhile N is the product of two relatively primes p and q. In this case, the public key e (usually in decimal numbers) will be converted to the Binary Numbers. Then, this number 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. Both processes 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: *LUC Cryptosystems, Parallel Algorithm, Binary Numbers, Lucas Functions, Distributed Memory Multiprocessors Machine.*

1. INTRODUCTION

Public key cryptosystem is a form of cryptography where a user has a pair of keys which are public key and a private key [1]. The private key is kept secret, while the public key may widely distributed. A message encrypted by public key can be decrypted with the corresponding private key. Among public key cryptosystems, the RSA cryptosystem is probably the most promising and widely used public key cryptosystem [11].

The computation of its encryption and decryption is based on exponentiation. Two researchers in [10] introduced another public key cryptosystem based on Lucas Function and it is known as LUC Cryptosystem. Lucas Function also used in factoring technique designed in [7]. Said and Loxton in [9] extended the LUC system into a cubic scheme of Lucas Function. Meanwhile, Castagnos in [6] worked on a public key cryptosystems over quadratic field quotients and also proposed another cryptosystem.

Related discussions of Luc Cryptosystem security can be found in [1], [3] and [5]. The existing sequential algorithms that were designed for LUC cryptosystem can be found in [4], [8], [12] and [13]. The current research on speed up the computation of LUC Cryptosystems can be found in [14].

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 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 work gives an idea where the

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

parallel computation is possible for any kind of public-key cryptosystem. Some idea in this paper gives an idea to do research in the same thrust for LUC Cryptosystems. Therefore, new parallel technique will be proposed. The algorithm that is proposed will be using Binary Numbers and manipulate some of the important properties of Lucas Functions. For example, if public-key is 17. Then, the Binary Number of 17 is 10001. The number 10001 is then use in manipulating some properties of Lucas Functions.

Then, the implementation part is very important. The parallel algorithms are writing in C language combined with the Message Passing Interface (MPI) package. All computer codes in C then run on distributed memory multiprocessors machine known as Sun Fire V1280. 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.

2. OVERVIEW OF LUC CRYPTOSYSTEMS

2.1 LUCAS FUNCTIONS

Let α and β be the roots of the polynomial equation $x^2 - Px + Q = 0$. It is easy to obtain $P=\alpha+\beta$ and $Q=\alpha\beta$. Two solutions of the general second order linear recurrence:

$$U_n = (\alpha^n - \beta^n)/(\alpha - \beta) \quad (1)$$

$$V_n = \alpha^n + \beta^n \quad (2)$$

Two functions in Lucas Sequence can be derived from Equations (1) and (2). They are defined as:

$$U_n = PU_{n-1} - QU_{n-2} \quad (3)$$

where $n\geq 2$, $U_0=0$ and $U_1=$

$$V_n = PV_{n-1} - QV_{n-2} \quad (4)$$

where $n\geq 2$, $V_0=2$ and $V_1=P$

As mentioned in [10], 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 $n\geq 2$, $V_0=2$ and $V_1=P$

Only some equations related to this research will be included in this paper. There are several functions that are useful but not used in this research can be found in [10].

$$V_{2n} = V_n^2 – 2 \quad (6)$$

$$V_{2n+1} = PV_n^2 - V_nV_{n-1} – P \quad (7)$$

$$V_{2n-1} = V_nV_{n-1} – P \quad (8)$$

Functions (6), (7) and (8) seems very appropriate to be use with the Binary Number of this study. Later the design of parallel algorithms using the Binary Number will be discussed in great detail in Section 3.

2.2 LUC CRYPTOSYSTEMS

LUC Cryptosystems is a public key cryptosystem. Two different keys are need for encryption and decryption. Throughout this paper, we denoted the public key as e and the private key as d. Key e is publicly known and key d is remain secret. The plaintext (original text) denoted as P and the ciphertext denoted as C.

Two relatively primes such as p and q are also important. Then, N is the product of p and q. The encryption function is $C = V_e(P,Q)(\text{mod } N)$. Meanwhile, the decryption function is $P = V_d(C,Q)(\text{mod } N)$.

In LUC Cryptosystem, Q=1 and this apply for its computation and design. This feature has been explained in detail by Smith and Lennon [10]. Therefore, to simplified encryption and decryption of LUC Cryptosystem, the encryption supposed to be $C=V_e(P,1)(\text{mod } N)$ and the decryption supposed to be $P = V_d(C,1)(\text{mod } N)$.

3. THEORITICAL ASPECTS

3.1 THE BINARY NUMBERS

Theorem 1: *Given an integer n, a sequence for n is a sequence of integers $\{b_0, b_1, \ldots, b_x\}$ such that b_0 = (0 or 1), b_1 = (0 or 1) and b_x = 0. The reverse of this sequence form a Binary Numbers.*

Proof: *It is straightforward, for b_0 until b_x, let z = n mod 2, if z = 1 then $\{b_0=1$ and $n=n/2\}$, otherwise if z = 0 then $(b_0=0$ and $n=n/2)$. Finally $b_x=0$. The reverse of $b_0, b_1,\ldots,b_x$ is a Binary Numbers* ■

Definition 1: *Given a sequence $\{b_0, b_1, \ldots, b_x\}$, the length of the sequence is x.*

Example 1: *Let consider that n=1103. An array k for Binary Numbers is $k[j]=\{1,1,1,1,0,0,1,0,0,0,1\}$. Then j=10.*

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

Algorithm 1 shows very simple steps for generating the Binary Numbers.

Algorithm 1: Generating Binary Numbers

Input: n and i=0
Output: Array k[0,1,2,…,j]
Repeat
  If (n mod 2 =1)
    k[i] =1
    n = n/2
  Else
    k[i] = 0
    n = n/2
  End if
  j++
Until n >= 1

The input for Algorithm 1 is the decimal number. For example, given n=1103, then Algorithm 1 will generate array k with k[0]=1, k[1]=1, k[2]=1, k[3],=1, k[4]=0, k[5]=0, k[6]=1, k[7]=0, k[8]=0, k[9]=0, k[10]=1. The reverse of this sequence is k[10]=1, k[9]=0, k[8]=0, k[7],=0, k[6]=1, k[5]=0, k[4]=0, k[3]=1, k[2]=1, k[1]=1, k[0]=1. The item within the array in reverse order formed Binary Number for 1103, there are 10001001111. Obviously, the array in reverse order will be use in manipulating the properties of Lucas Functions as found in Equations (6), (7) and (8).

This decimal number comes from e in the encryption $V_e = V_n(M,1) \pmod N$ or from d in the decryption $V_d = V_n(C,1) \pmod N$. The LUC Cryptosystem is using the Lucas Function in its computation. The output of this algorithm is array k[0,1,2,…,j]. An array k is used to keep the value of a sequence. Once array k is generated, this array will be used to organize the parallel computation of LUC Cryptosystems.

3.2 COMPUTATION OF LUC CRYPTOSYSTEMS

This computation algorithm is design to be used only with one processor. Once the sequence of array k is generated, this array is used in the algorithm. Algorithm 2 represents the computation algorithm for one processor that can be run in any computer machine. It is also can be run in any distributed memory multiprocessor machine.

Algorithm 2: Basic LUC Cryptosystem Computation

Initial: Generate a Binary Numbers, k[j] (refer to Algorithm 1)
Input: N=p*q, $V_n=V_1=P$, $V_j=V_0=2$ and i=j
Output: V_n
Repeat
  $V_{2n} = V_n^2 - 2 \pmod N$;
  If k[i] = 0 then
    $V_{2n+1} = PV_n^2 \cdot V_n V_{n-1} - P \pmod N$;
    $V_n=V_{2n}$
    $V_j=V_{2n-1}$
  Else
    $V_{2n-1} = V_n V_{n-1} - P \pmod N$;
    $V_n=V_{2n+1}$
    $V_j=V_{2n}$.
  EndIf
  i--
Until i=0

Let e=1103, P is a message (plaintext), p and q are relatively primes number. The initial part of Algorithm 2 is to generate a Binary Number for public key, e. All of this depends on Algorithm 1. Given e=1103, then Algorithm 1 will generate array k with k[0]=1, k[1]=1, k[2]=1, k[3],=1, k[4]=0, k[5]=0, k[6]=1, k[7]=0, k[8]=0, k[9]=0, k[10]=1. The reverse of this sequence is k[10]=1, k[9]=0, k[8]=0, k[7],=0, k[6]=1, k[5]=0, k[4]=0, k[3]=1, k[2]=1, k[1]=1, k[0]=1.

The item within the array in reverse order formed Binary Number for 1103, there are 10001001111. Clearly, by Algorithm 2, the computation of LUC Cryptosystems will be V_2, V_4, V_8, V_{17}, V_{34}, V_{68}, V_{137}, V_{275}, V_{551}, V_{1103}.

The following table, explain about how Algorithm 2 works. Noted that No indicates no computation for specific equations, while Yes indicates need computation for specific equations.

It shows direct representation for computation algorithm that can be run on one processor, such as Intel, AMD or any kind of processors. Algorithm 2 is suitable for both encryption and decryption process (Ali, 2010). For encryption the public key is needed, while for decryption the private key is needed. In real implementation, the public

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

key is known by public. Therefore, the experiment is only concentrate on the computation time for encryption.

Table 1: Calculating LUC Cryptosystems Using Algorithm 2.

i	k[i]	V_n	V_j	V_{2n}	V_{2n-1}	V_{2n+1}
10	k[10]	V_1	V_0	No	No	No
9	k[9]	V_2	V_1	Yes	Yes	No
8	k[8]	V_4	V_3	Yes	Yes	No
7	k[7]	V_8	V_7	Yes	Yes	No
6	k[6]	V_{17}	V_{16}	Yes	No	Yes
5	k[5]	V_{34}	V_{33}	Yes	Yes	No
4	k[4]	V_{68}	V_{67}	Yes	Yes	No
3	k[3]	V_{137}	V_{136}	Yes	No	Yes
2	k[2]	V_{275}	V_{274}	Yes	No	Yes
1	k[1]	V_{551}	V_{550}	Yes	No	Yes
0	k[0]	V_{1103}	V_{1102}	Yes	No	Yes

3.3 PARALLEL STRATEGIES

The main concern here is the computation of V_e, the encryption process. The public-key, e is known in public. Furthermore, the parallel strategies can be achieved by using Algorithm 1. The Binary Numbers can also be used in designing the parallel algorithms. The main idea is manipulating functional decomposition for Lucas Functions.

It means that, each part of computation task is actually the small part of the function. Simply, every function involved in the calculation of LUC Cryptosystem can be broken into small parts. These small parts will be process by each processor available. Each time the processor is handling these parts, every time that, the calculation results will be returned to the main processor.

For example, the Equation (7) in Section 2.1 above can be decomposed into small parts. Equation (7) as $V_{2n+1} = PV_n^2 - V_nV_{n-1} - P$, computation of PV_n^2 will be handled by one processor and the other part of that equation, V_nV_{n-1} will be handled by another processor. The result will be instantly sends back to the master processor and this master processor will compute V_{2n+1}. This is the concept of functional decomposition used in designing the parallel algorithms.

Overall, the result of each small task is collected and will be synchronized to produce the final result of parallel computation. Furthermore, one task can be assigned into one slave.

The master will synchronized the movement of result from each task and decided which value is using for the next computations. The synchronization is the compulsory steps in this parallel computation. If the process of computation cannot be synchronized, the result did not satisfy the nature of Lucas Function computations.

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 Binary Numbers.
c. Use each item in the Binary Number 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.

4. THE PROPOSED ALGORITHM

4.1 OVERALL CONCEPTS

The algorithms discussed in this section can be divided into three parts. Part 1 is the algorithm used to generate an array sequence for Binary Numbers. This already discussed in Section 3.1.

Meanwhile, second part is the algorithm on how to use Algorithm 1 to design a solution of LUC Cryptosystem computation for one processor. This is also discussed in Section 3.2. The third part shows on how to design parallel computation for LUC Cryptosystem depending on the number of processors.

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

Overall idea on using the Binary Numbers for parallel algorithm is discussed in Section 3.3. The comparison will be carried out to compare the computation time of one processor compare to the computation of several processors. Up to this stage, the possible maximum number of processors that can be used in this parallel algorithm is seven.

4.2 THE COMPUTATION ALGORITHMS FOR SEVERAL PROCESSORS

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. Algorithm 1 will generate an array k.

Once an array k is generated, this array should be used in reverse order. Therefore, the using of array k should start with item k[j]. Followed by k[j-1] and so on until the index of array k reached 0. Generally, the idea on using the array of Binary Numbers is shown in Figure 1.

The array k[j], k[j-1] until k[0] represents the index of array. Definitely, it is also show the numbers of iterations required to complete the parallel computation.

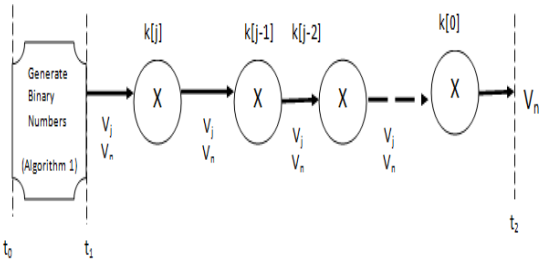

Figure 1: General Idea on Using Binary Numbers

The calculation of V_e starts with two values V_1 and V_0. The initial values are required at the beginning of the computation, therefore the initial values are V_1=2 and V_0=P. Note that, in all algorithms, MPI_Send() is denoted by Send and MPI_Recv() is denoted by Receive.

Symbol 'X' represents the inner side of the parallel algorithms. Figure 2 and 3 show the example of the inner side of the parallel algorithms for two and seven processors. Meanwhile, V_j and V_n show the value of variables that was passed from current iteration to the next iteration. The symbols V_j and V_n show the possible values that should be use for the next computations. The final result is V_n.

The timeline t_0 to t_1 shows the time to generate Binary Numbers. The t_1 and t_2 shows the time to compute LUC Cryptosystem in parallel. Therefore, the time line from t_0 and t_2 shows the total computation time for this parallel approach.

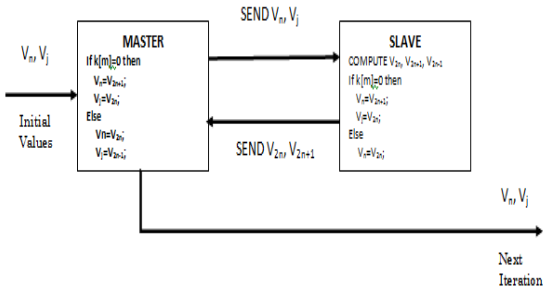

Figure 2: Parallel Strategy For Two Processors.

Figure 2 shows a possible arrangement of functional decomposition for parallel algorithm on 2 processors. Here, one processor will act as master and the other one should be act as a slave. The possible parallel codes that are suitable for Figure 2 are as follows.

The timeline t_0, t_1 and t_2 are not shown in this figure. The timeline t_0 and t_1 is the same as in Figure 1. The timeline t_2 in Figure 2 is surely less than t_2 in Figure 1. Same goes to the other time line in Figure 3, the timeline t_0 and t_1 is the same as Figure 1.

For sure, the timeline t_2 in Figure 3 is surely less than timeline t_2 in Figure 2. It is because the Figure 3 has seven processors compare to Figure 1 and 2. However, all of this timeline is now shown in Figure 2 and 3.

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

Algorithm 3: Computation of LUC Cryptosystems on Two Processors

Initial: Generate a Binary Numbers, k[0,1,2,…,j] (refer to Algorithm 1).
Input: N=p*q, $V_n=V_1$, $V_j=V_0$ and m=j
Result: V_n
MPI Initialization.
Repeat
At master:
Send V_n and V_j to slave.
Receive V_{2n}, V_{2n-1} and V_{2n+1} from slave.
If k[m]=1 **then**
$V_n=V_{2n+1}$
$V_j=V_{2n}$
Else
$V_n=V_{2n}$
$V_j=V_{2n-1}$.

At slave:
Receive V_n and V_j from master.
Compute V_{2n}(mod N) [Using Equ. (6)]
Compute V_{2n-1}(mod N) [Using Equ. (7)]
Compute V_{2n+1}(mod N) [Using Equ. (8)]
Send V_{2n}, V_{2n-1} and V_{2n+1} to master.
If k[m]=1 **then**
$V_n=V_{2n+1}$
$V_j=V_{2n}$.
Else
$V_n=V_{2n}$
$V_j=V_{2n-1}$.
m--
until m=0
MPI Close.

The movement of data between master and slave(s) must be synchronized in order to make sure that the master is presented with the right value of V_n and V_j. If the master is provided by the unsynchronized result, the final result of the parallel computation may be wrong.

The suggestion of synchronization between master and slave(s) can be found in Table 1. At anytime, one processor must be acted as master. The other processors are slaves. Based on Table 1, the parallel strategy for parallel computation on seven processors is shown in Figure 3. Master also has the biggest number of MPI_Send() and MPI_Recv() commands. It is shown clearly in Algorithm 4.

The programming codes for master have a biggest number of Send and Receive commands. The master is also responsible in synchronization of data movement between master and slaves. This technique can be achieved by equally distributes the computation jobs into slaves.

All computations are done by slaves. The master only does synchronization. Slaves are responsible for computation and also the initiation of the next values for the next computation.

Some slaves do very small computation for example Slave 1. Slaves are also responsible to update its values and acknowledge the updated values of the other slaves. The parallel algorithm on seven processors is explained Algorithm 4 below.

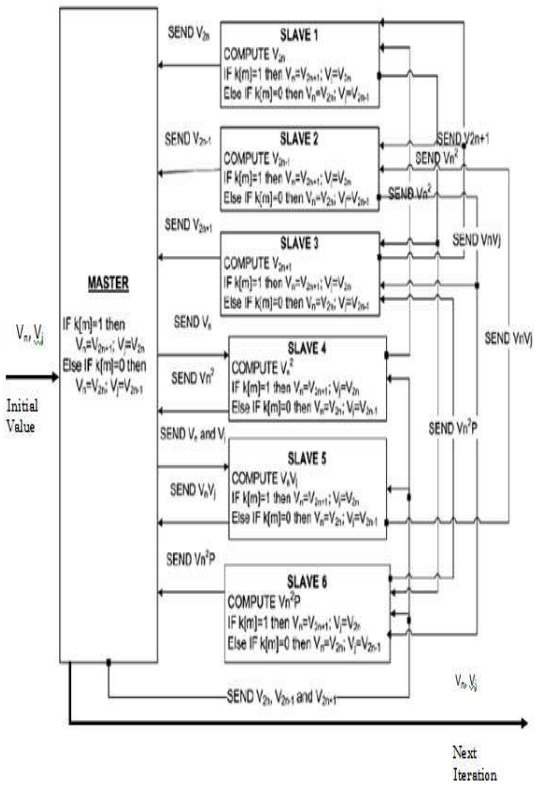

Figure 3: Parallel Strategy for Seven Processors.

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

Algorithm 4: Computation of LUC Cryptosystems on Seven Processors

Initial: Generate a Binary Numbers, k[0,1,2,…,j] (refer to Algorithm 1).
Input: N=p*q, $V_n=V_1$, $V_j=V_0$ and m=j
Result: V_n
MPI Initialization
Repeat
At master :
Send V_n to Slave 4; V_n and V_j to Slave 5
Send V_{2n}, V_{2n-1} and V_{2n+1} to Slave 4, 5 and 6
Receive V_{2n} from Slave 1; V_{2n-1} from Slave 2; and V_{2n+1} from Slave 3
Receive V_n^2 from Slave 4; V_nV_j from Slave 5
Receive V_n^2*P from Slave 6
If k[m]=1 then
$V_n = V_{2n+1}$
$V_j= V_{2n}$
Else
$V_n = V_{2n}$
$V_j= V_{2n-1}$

At slave 1 :
Receive V_{2n+1} from Slave 3; V_n^2 from Slave 4
Compute $V_{2n} = V_n^2 – 2 \pmod N$ [Equ. (6)]
Send V_n^2 to Slave 3 and Slave 6; V_{2n} to master
If k[m]=1 **then**
$V_n = V_{2n+1}$
$V_j= V_{2n}$
Else
$V_n = V_{2n}$
$V_j= V_{2n-1}$
At slave 2 :
Receive V_{2n+1} from Slave 3; V_nV_j from Slave 5
Compute $V_{2n-1} = V_nV_j – P \pmod N$ [Equ. (7)]
Send V_{2n-1} to master; V_nV_j to Slave 3; V_nV_j to Slave 6
If k[m]=1 **then**
$V_n = V_{2n+1}$
$V_j= V_{2n}$
Else
$V_n = V_{2n}$
$V_j= V_{2n-1}$

At slave 3 :
Receive V_{2n} from Slave 1; V_nV_j from Slave 2; V_n^2*P from Slave 6
Compute $V_{2n+1} = V_n^2P - V_nV_j - P \pmod N$ [Equ. (8)]
Send V_{2n+1} to master and Slave 1 and 2
If k[m]=1 **then**
$V_n = V_{2n+1}$
$V_j= V_{2n}$
Else
$V_n = V_{2n}$
$V_j= V_{2n-1}$

At slave 4 :
Receive V_n, V_{2n}, V_{2n-1} and V_{2n+1} from master
Compute V_n^2
Send V_n^2 to master and Slave 1
If k[m]=1 **then**
$V_n = V_{2n+1}$
$V_j= V_{2n}$
Else
$V_n = V_{2n}$
$V_j= V_{2n-1}$

At slave 5 :
Receive V_n, V_j, V_{2n}, V_{2n-1} and V_{2n+1} from master
Compute V_nV_j
Send V_nV_j to master and Slave 2
If k[m]=1, **then**
$V_n = V_{2n+1}$
$V_j= V_{2n}$
Else
$V_n = V_{2n}$
$V_j= V_{2n-1}$

At slave 6 :
Receive V_n^2 from Slave 1; V_nV_j from Slave 2
Receive V_n, V_j, V_{2n}, V_{2n-1} and V_{2n+1} from master
Compute V_n^2*P
Send V_n^2*P to master and Slave 3
If k[m]=1, **then**
$V_n = V_{2n+1}$
$V_j= V_{2n}$
Else
$V_n = V_{2n}$
$V_j= V_{2n-1}$

m--
Until m=0
MPI Close

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.

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

The listed below shows a totally different approach and strategy in design parallel algorithms for a numbers of processors:

a. **Computation of LUC Cryptosystem**. The computation of this cryptosystem is simple done by compute V_{2n}, V_{2n-1} and V_{2n+1} in slaves and send the computations results to master. Master and each slave then check for the next values to be used.

   For example, refer to Algorithm 3 and 4 above. Algorithm 3 requires two processors. One will serve as master and the other as slave. The master sends the value to slave. The slave does the computations and sends back the results to the master. On the other hand, Algorithm 4 does the other approach. The master sends the value to the slaves. Each slave then does their specific task and returns the results to the master. In this case, it is clear that Algorithm 4 distributes the computation task to the slave in better way and surely can achieved high performance computation compared to Algorithm 3.

b. **Distribution of computation jobs among the processors**. The algorithms 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.

   For example, refer to Algorithm 3 and 4 above. In Algorithm 3, the slave suffers with computations task. Here, the slave is required to do all computation for Equation (6), (7) and (8). On the other hand, Algorithm 4 does the other approach. Equations (6), (7) and (8) are chunk into small pieces. Each slave does the computation task and the master brilliantly synchronize the results from slaves.

c. **Maximum number of processors**. The maximum of processors are seven processors. Each processor is used in different strategy. This is clearly shown in Algorithm 4.

d. **Checking the next value to be used**. The next value to be used for the next computation should be known and initiated.

   The main reason of the checking procedure is to make sure the master will sending the right value to the slave. Each item in the Binary Number 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 Binary Number is 1 then the next value for subsequent calculations is V_{2n+1} and V_{2n}. On the hand, if the item of Binary Numbers is 0, then the next value for subsequent calculations is V_{2n} and V_{2n-1}.

5. PERFORMANCE EVALUATION AND RESULTS

In this experiment, the parallel machine that was selected is Sun Fire V1280 server. It is the example of the distributed memory multiprocessors machine.

Offer CPU/memory board Dynamic Reconfiguration, hot swap power supplies and disks, and Lights Out Management (LOM) for reduced downtime and easier maintenance.

It is scales up to twelve award-winning UltraSPARC¨ III processors in a symmetric multiprocessing architecture for investment protection. It is also supports up to 96GB memory to provide the headroom needed for constrained Windows applications. This machine is designed to provide high performance tool in a compact form.

The capability of each algorithm can be determined by running the experiments on different sizes of keys and primes. Different key size must produce different computation time. Table 2 shows different size of keys. Meanwhile, Table 3 shows different size of primes.

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

Table 2: Different Size of Public Keys (In Digits)

Public Key e	Primes p & q	Messages P
159	*100*	*5*
339	*100*	*5*
579	*100*	*5*

Table 3: Different Size of Primes (In Digits)

Primes p & q	Public Key e	Messages P
160	*159*	*20*
220	*159*	*20*
280	*159*	*20*

The computation time is included in each set of data as the comparison of the existing algorithm and the proposed algorithm. Table 4 shows the comparison of encryption computation time for each algorithm on different sizes of public key. The bigger the public key size, the longer computation time is produced for LUC Cryptosystem computation.

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.

Table 4: Computation Time For Different Size of Public Keys (In Seconds)

No. of Processors	Public Key e		
	159 digits	339 digits	579 digits
1	*55.98*	*297.34*	*573.06*
2	*41.99*	*222.00*	*445.09*
7	*13.13*	*75.78*	*149.74*

On the other hand, Table 5 shows the encryption computation time for different prime size. The bigger size of primes, the longer computation time is a need. Note that, the same size of public key and message is used but the sizes of primes are changed. All tables show that the parallel algorithm is better.

Table 5: Computation Time For Different Size of Primes (In Seconds)

No of Processors	Primes p and q		
	160 digits	220 digits	280 digits
1	*218.87*	*290.99*	*419.49*
2	*182.54*	*231.11*	*344.43*
7	*58.45*	*76.29*	*113.38*

6. CONCLUSIONS

The parallel computation algorithms are successfully implemented in the distributed memory multiprocessor machines. The length of array shows that the parallel computation algorithm must use Binary Numbers.

The computation of LUC Cryptosystem shows that in the parallel implementation, it will compute the values of V_{2n}, V_{2n-1} and V_{2n+1} in some slaves and send the results to master. Master and each slave then check the next values to be used. In all computations, modulo N (N=p*q) should be used in all computation of V_{2n}, V_{2n-1} and V_{2n+1}.

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.

ACKNOWLEDGMENTS:

The author wishes to thank Universiti Kebangsaan Malaysia (UKM) and Government of Malaysia for this work. This work was supported by FRGS/1/2012/SG05/UKM/02/1 research grant from Jabatan Pengajian Tinggi, Malaysia.

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

REFRENCES:

[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] C. T. Wang, C.C. Chang and C.H. Lin, A Method for Computing Lucas Sequences, International Journal Computers and Mathematics with Application, 38, 1999, 187-196.

[5] 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.

[6] G. Castagnos, An Efficient probabilistic public-key cryptosystem over quadratic fields quotients, Finite Field and Their Applications 13, Elsevier, 2007, 563-576.

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

[8] M. Joye and J.J Quisquater, Efficient Computation of Full Lucas Sequences, IEEE Electronics Letters, vol.32, no.6, 1996, 537-538.

[9] M.R.M. Said and J. Loxton, A Cubic Analogue of The RSA Cryptosystem, Bull. Austral Math Soc, Vol. 68, 2003, 21-38.

[10] 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.

[11] 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.

[12] S. Chiou and C. Laih, An Efficient Algorithm for Computing The LUC Chain, IEEE Proceedings-Computers and Digital Techniques, vol.147, no.4, 1995, 263-265.

[13] S. Yen and C. Laih, Fast Algorithm for LUC Digital Signature Computation, IEEE Proceeding: Computer and Digital Techniques, vol.142, no.2, 1995, 165-169.

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