MODIFIED RSA ALGORITHM: ENHANCEMENT OF RSA ENCRYPTION ALGORITHM USING PARALLEL PROGRAMMING TECHNIQUES

A PROJECT REPORT

Submitted by

SHRESHTH VATS (18BCE1077)
SHIVIKA MANGLICK (18BCE1339)
ANMOL CHOUBEY (18BCE1276)

In partial fulfillment for the award of the degree of

Bachelor of Technology

In

(Computer Science and Engineering discipline)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
VELLORE INSTITUTE OF TECHNOLOGY
VANDALUR – KELAMBAKKAM ROAD, CHENNAI - 600127

November 2020

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

DECLARATION

We hereby declare that the project entitled MODIFIED RSA ALGORITHM: ENHANCEMENT OF RSA ENCRYPTION ALGORITHM USING PARALLEL PROGRAMMING TECHNIQUES submitted by us to the School of Computer Science and Engineering, VIT Chennai, 600 127 in partial fulfillment of the requirements of the award of the degree of Bachelor of Engineering in (Computer Science and Engineering) is a bona-fide record of the work carried out by us under the supervision of Prof. Dr. V. Muthumanikandan. We further declare that the work reported in this project, has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma of this institute or of any other institute or University.

Place: Chennai

Date: 9th November, 2020

Signature of Candidate

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

CERTIFICATE

This is to certify that the report entitled MODIFIED RSA ALGORITHM: ENHANCEMENT OF RSA ENCRYPTION ALGORITHM USING PARALLEL PROGRAMMING TECHNIQUES is prepared and submitted by Shreshth Vats (Reg. No. 18BCE1077), Shivika Manglick (Reg. No. 18BCE1339) and Anmol Choubey (Reg. No. 18BCE1276) to VIT Chennai, in partial fulfillment of the requirement for the award of the degree of Bachelor of Engineering in (Computer Science and Engineering) is a bona-fide record carried out under my guidance. The project fulfills the requirements as per the regulations of VIT and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma and the same is certified.

Guide/Supervisor

Name: Prof. Muthumanikandan
Date: 9th November, 2020

Acknowledgement

We obliged to give our appreciation to a number of people without whom We could not have completed this thesis successfully.

We would like to place on record our deep sense of gratitude and thanks to our internal guide Prof. Dr. V. Muthumanikandan, School of Computer Science and Engineering (SCOPE), Vellore Institute of Technology, Chennai, whose esteemed support and immense guidance encouraged us to complete the project successfully.

We thank our management of Vellore Institute of Technology, Chennai, for permitting me to use the library and laboratory resources. We also thank all the faculty members for giving me the courage and the strength that we needed to complete our goal. This acknowledgment would be incomplete without expressing the whole hearted thanks to our family and friends who motivated us during the course of our work.

Shreshth Vats
Reg. No. 18BCE1077
Shivika Manglick
Reg. No. 18BCE1339
Anmol Choubey
Reg. No. 18BCE1276

Abstract

Encryption Algorithms are necessary part of the data sharing today as every bit of data need encryption so that it could be transferred from one place to another without the fear of data leaking. Every lost bit of data puts the system to danger. Encryption algorithms compose of necessary steps involving a key that helps converting plain text to cipher text. All the steps must be performed carefully to ensure that the data encrypted is found back. Thus, many algorithms are made to run sequentially and in serial.
This makes the algorithm to take much computation time.
We aim to reduce the total computation time needed to complete the process by parallelizing the algorithm using the concept of multi-threading. The complexity, data structures and the overall procedure of the algorithm would remain constant.
Key generation is also an important part of encrypting data and is also a tedious task to be done sequentially. Hence, aside the encryption, we would try to break the key generation procedure too while parallelizing. This would ensure in getting a better time of completion of the algorithm.

Contents

List of Figures

Chapter 1

Introduction

RSA encryption is a public-key encryption technology developed by RSA Data Security. The RSA algorithm is based on the difficulty in factoring very large numbers. Based on this principle, the RSA encryption algorithm uses prime factorization as the trap door for encryption. Deducing an RSA key, therefore, takes a huge amount of time and processing power. RSA is the standard encryption method for important data, especially data that's transmitted over the Internet.
RSA stands for the creators of the technique, Rivest, Shamir and Adelman.

RSA encryption is a public key encryption technology developed by RSA Data Security, which licenses the algorithm technologies and also sells the development kits. RSA is built into many common software products, including Microsoft's Internet Explorer.

A person using RSA encryption finds the product of two large prime numbers, which are kept confidential. With additional mathematical operations, two sets of numbers - public and private keys - are developed. Once the public and private keys are derived, the large numbers can be discarded.

It is one of the famous algorithms for public-key cryptography. It is appropriate for encryption and digital signature. RSA is the utmost far used algorithm in Internet security . In fact, Internet security depends significantly on the security properties of the RSA cryptosystem. Its security depends upon the insolvability of the integer factorization problem and is believed to be vulnerable given sufficiently long keys, such as 1024 bits or more.
To guard data transmitted from snooping by someone other than the receiver. It is desired to hide the message before it is sent to a non-secure communication channel. This is achieved through encryption.

Due to its distinctive ability to distribute and manage keys, public key encryption has become the perfect solution to information security . Public key algorithms (e.g., RSA algorithm) rely essentially on hard mathematical problems (modular multiplication and modular exponentiation) of very large integers, ranging from them.

1.1 Background

Approach 1 - Devise a dedicated piece of hardware to solve the problem. However, there are a number of limitations that can be faced regarding to the issue, like the cost.
Approach 2 – Using parallel programming to solve the problem. It may not reach the same performance enhancement level of a dedicated hardware. However, it provides

a relatively cheap alternative to the dedicated hardware. It can be applied even to normal computer users’ machines.

1.2 Statement

This is a very expensive approach that can be adopted by the big companies and organizations, but not by normal computer users. Usually there is a scalability limitation of the dedicated hardware, so after a period of time another device will be required.

1.3 Motivation

One approach used to enhance the performance of commonly-used algorithms is to devise a dedicated piece of hardware to solve the problem. However, there are a number of limitations that can be faced, such as:

- This is a very expensive approach that can be adopted by the big companies and organizations, but not by normal computer users.
- Usually there is a scalability limitation of the dedicated hardware, so after a period of time another device will be required.

On the other hand, another approach can be used which exploits parallel programming to solve the problem. This approach may not reach the same performance enhancement level of a dedicated hardware; however, it addresses the first approach limitations by providing a relatively cheap alternative to the dedicated hardware, can be applied even to normal computer users’ machines. Furthermore, the scalability is much easier, especially when the parallel algorithm is well-designed. Moreover, a typical computer system nowadays contains a multi-core processor which has eight cores or more, thus, it is a wise decision to make use of all of them to enhance the overall performance.

1.4 Challenges

The RSA algorithm is a very secure, but slow, as it contains heavy mathematical computations, thus parallelizing this algorithm will make it perform faster and may increase the security level provided by this algorithm by increasing the key size. RSA now uses keys in 1024 bit length which are not considered secure enough. However, the longer the key used the more security and the slower the performance is obtained. This leads to the need to use parallel computing to increase the security level without adversely affecting the performance.

Chapter 2

Planning & Requirements Specification

The planning that was necessary and the requirement specifications related to it are all given below.

2.1 Literature Review

Wenjun Fan, and Xudong Chen. This paper presents a novel parallelized implement of RSA algorithm using JCUDA and Hadoop. Secondly, the parallel RSA algorithm is designed and realized in CUDA framework. Thirdly, with JCUDA, the RSA parallel algorithm implement function is called by each node in Hadoop cluster. Our experimental results demonstrate the speed of RSA algorithm enhanced dramatically compared to the original method on the CPU only. In this paper, we presented our experience of porting RSA algorithm on to CUDA architecture. We analyzed the parallel RSA algorithm, described our parallelization techniques at tow parallel levels, and discussed the results in tow aspects.

Mohammed Issam Younis. This paper proposes variant decompositions to gain extra speed up. for the decryption process with CRT, it is noticed that the adopting CRT sequential version gives a speed up gains ~14X. Thus, CRT gives a significant speed up for the decryption process for all three variant implementations. In addition, in both cases for Multi-cores and Many-cores, the speed up is super due to composition of parallel processing and CRT. From our case study on parallelizing RSA algorithm on multi-cores CPU and many-cores GPU by decomposition the algorithm in independent data level and/or task level parts, a noticeable speed up and throughput can be gained.

Rahul Saxena. Experimental results under the lights of graphical representation shows a considerable speed up gained over the traditional implementation. Further the paper also compares the efficiency with other parallel implementations of RSA algorithm where the reported algorithm is found to perform almost equivalent to the GPGPU based implementation without the need of extra hardware in the form of graphics card and with reduced power consumption of the machine.

Balkees Mohamed Shereek. The study shows that the proposed framework provides an improved level of security when using with 65,537 key size consumed 23 milli seconds, while using 257 bits key size which consumed 21 milli seconds. A secure MapReduce model is proposed and implemented that is adaptable for running any application. The goal is derived with a view to provide an improved level of security to MapReduce framework that is being used widely in Cloud in recent years. The model is designed with providing three levels of security first at user level, second at the MapReduce process level and third at the HDFS level. Such policies enforced in the new model provide high level of security for Hadoop's MapReduce and HDFS.

Manisha N. Kella. Parallelization of these security algorithms in order to distribute the complex computational part among the various cores available with the processors today, will achieve higher performance and also be more energy efficient. This paper presents the most efficient, currently known approaches in encryption and decryption of text with RSA on programmable GPUs, achieving up to a great speed on a comparable CPU. If the amount of data is large, the encryption/decryption time required is greatly reduced, if it runs on a graphics processing environment.

Ahsan Ayub. The survey represents several avenues on which the algorithm could be parallelized as well as techniques or methods that allow sequential processes to run simultaneously on different processing elements, be it on Central Processing Unit (CPU) and / or Graphics Processing Unit (GPU). The survey should give researchers a foundation to understand the types of research available in this field of study and help develop strategies to further improve on parallelization of the RSA algorithm. Additionally, our research presents a parallel implementation of the RSA algorithm focused on one of the possible discussed avenues, which is the exponentiation operations.

Heba Mohammed Fadhil. This paper proposes a hybrid system to parallelize the RSA for multicore CPU and many cores GPUs with variable key size. In doing so, three variants implementation for the RSA algorithm are done to facilitate the performance comparison against Crypto++ library and sequential counterpart. The GPU implementation gained approximately 23 speed up factor over the sequential CPU implementation; while the multithread CPU implementation gained only 6 speed up factor over the sequential CPU implementation as far as the latency is concerned.

Yunefi Li. This paper aims at speeding up RSA decryption and signature. This paper proposes a variant of RSA cryptosystem (EAMRSA-Encrypt Assistant MultiPrime RSA) by reducing modules and private exponents in modular exponentiation. The experimental result shows that the speed of the decryption and signature has been substantially improved and the variant can be efficiently implemented in parallel.

Deepali and Namita Kakkar. Parallel AES Algorithm for Fast Data Encryption on with the improvement of cryptanalysis, many applications are starting to use Advanced Encryption Standard (AES) instead of Data Encryption Standard (DES) to secure the information. As current execution of AES algorithm affected from immense CPU resource consumption and less throughput. In this paper, we measured the techniques of GPU parallel computing and its optimized design for cryptography. The test proves that our technique can fasten the speed of AES encryption significantly.

Dr. Prerna Mahajan and Abhishek Sachdeva. In this paper we implemented three encrypt techniques like AES, DES and RSA algorithms and compared their performance of encrypt techniques based on the analysis of its stimulated time at the time of encryption and decryption. Experiments results are given to analyse the effectiveness of each algorithm. Our research work surveyed the performance of existing encryption techniques like AES, DES and RSA algorithms. Based on the text files used and the experimental result it was concluded that AES algorithm consumes least encryption and RSA consume longest encryption time. We also observed that Decryption of AES algorithm is better than other algorithms. From the simulation result, we evaluated that AES algorithm is much better than DES and RSA algorithm. Our future work will focus on compared and analyzed existing cryptographic algorithm like AES, DES and RSA.

M. Preetha. The core idea is enhance the security of RSA algorithm. In this dissertation public key algorithm RSA and enhanced RSA are compared analysis is made on time based on execution time. According to this scheme it has extra advantages, namely its IND-CCA, security remains highly related to hardness of the RSA problem, even in the multi-query setting. The RSA provides highest security to the business application. Moreover, this scheme can be used for encryption of long messages without employing the hybrid and symmetric encryption.

Kalpesh S. Prajapati. This paper mainly focuses on improvements and modification done in RSA algorithm. In this paper, we surveyed different methods modified by various researchers and scholars for faster implementation and security enhancement of RSA algorithm. Those techniques are studied and analyzed deeply to promote the performance of RSA algorithm and also to ensure the security of information. All the techniques are useful to speed up the RSA algorithm and for better security. Everyday new approach is evolving hence fast and secure RSA algorithm always work out with high rate of security.

Rohit Minni. In this paper they present a modified algorithm for RSA with enhanced security. The security feature here is the elimination of n from the original RSA algorithm. Instead, the newly generated replacement for n can be used in both the keys. The RSA algorithm is prone to mathematical factorization attacks. The algorithm that we presented in this paper eliminates this issue making the algorithm more secure with a slight increase of time complexity.

Shaheen Saad Al-Kaabi. The following paper preview different proposals on different methods used to enhance the RSA algorithm and increase its security. Some of these enhancements include combining the RSA algorithm with Diffie-Hellman or ElGamal algorithm, modification of RSA to include three or four prime numbers, offline storage of generated keys, a secured algorithm for RSA where the message can be encrypted using dual encryption keys, etc.

Israt Jahan. In this paper, they have proposed an improved approach of RSA algorithm using two public key pairs and using some mathematical logic rather than sending one public key d irectly. Because if an attacker has an opportunity of getting the public key componet they can find private key value by brute force search.

Devashish Vaghela. We have studied different type of encryption technique like bit rotation, bit reversal, matrix multiplication to improve security. Here during our literature survey we found existing encryption techniques uses secret key in order to encrypt and decrypt image so here we can increase security of image sharing using public private key pair.

Shaheen Saad Al-Kaabi. During our literature survey we found different proposals on different methods used to enhance the RSA algorithm and increase its security.

P. Saveetha and S. Arumugam. Research to speed up the Key generation in the RSA algorithm describes a Secure and Fast Generation of RSA Public and Private Keys on Smart-Card. Smart-Cards are widely used for security services because of their low cost, employ advanced cryptographic algorithms to maintain the desired security levels. Public key cryptography is a popular method that provides security, identification and authorization in such secure systems.

Ahmed Eskander Mezher. In this paper, a method has been designed and implemented to strengthen the RSA algorithm by using multiple public and private keys.

Fausto Meneses, Walter Fuertes, José Sancho, Santiago Salvador, Daniela Flores, Hernán Aules, Fidel Castro, Jenny Torres Alba Miranda and Danilo Nuela. This paper aims to optimize the RSA encryption algorithm and thus improve the security, integrity and availability of information. The results show the efficiency and functionality of the RSA algorithm in terms of information security.

Shaheen Saad Al-Kaabi and Samir Brahim Belhaouari. This paper presents a modified approach which is an enhancement over traditional RSA algorithm by including exponential powers, n prime numbers, multiple public keys, and K-NN algorithm. Modified approach also gives feature of verification at both side's sender and receiver.

Israt Jahan, Mohammad Asif and Liton Jude Rozario. In this paper, we have proposed an improved approach of RSA algorithm using two public key pairs and using some mathematical logic rather than sending one public key directly.

Bodake Vijay and Gawande R.M.. With the development of the GPGPU (General-purpose computing on graphics processing units) more and more computing problems are solved by using the parallel property of GPU (Graphics Processing Unit). CUDA (Compute Unified Device Architecture) is a framework which makes the GPGPU more accessible and easier to learn for the general population of programmers. The target in this project is to study and analyse the majority of algorithms related to the cryptography and then to design and make an implementation of an algorithm in CUDA.

Shweta Kumari and Abhishek Kumar. Parallel computation is a promising technique to improve the performance of cryptography algorithm. Mainly divide-and-conquer strategy is used in parallel computation to solve the algorithms in parallel by partitioning and allocating, number of given subtask to available processing units. Parallel computation can be performed using multicore processors by parallelizing the execution of algorithm in multiple cores. In this paper we explore the implementation of AES (Advanced Encryption Algorithm) cryptography algorithm on dual core processor by using OpenMP API to reduce the execution time.

Sonam Mahajan and Maninder Singh. The main research in computer security domain is how to enhance the speed of RSA algorithm. The computing capability of

Graphic Processing Unit as a co-processor of the CPU can leverage massive-parallelism. This paper presents a novel algorithm for calculating modulo value that can process large power of numbers which otherwise are not supported by built-in data types. First the traditional algorithm is studied. Secondly, the parallelized RSA algorithm is designed using CUDA framework. Thirdly, the designed algorithm is realized for small prime numbers and large prime number.

https://ieeexplore.ieee.org/document/7164703 - The cryptography algorithms are divided into two parts symmetric and asymmetric. There are many different challenges to implement cryptography algorithm specially throughput in terms of time execution. In this paper, we study and analyze the performance of different cryptographic algorithm on multicore processors and also we explore the performance in sequential and parallel implementation of cryptography algorithm on multi core processors.

U. Thirupalu. Cloud computing is one of the fastest growing internet based technology. Data encryption is one of the widely used methods to ensure the data confidentiality in cloud environment. In this paper, we discuss the symmetric and Asymmetric algorithms to provide security in the field of cloud computing with different parameters and propose a new approached public key cryptosystem for security in cloud computing.

Sapna Saxena. The objective of this research is to study and analyze various public key infrastructure based cryptographic algorithms and design new parallel algorithms to implement the public-key algorithms such as the RSA and the Digital Signature Algorithm. This research work has been divided into four main parts – (i) to convert memory-efficient algorithms into new parallel algorithms, (ii) to use important parallel programming techniques to obtain the new high xxiv performance parallel variants of these algorithms, (iii) to use OpenMP API to test the algorithms and analyze performance gained by parallelizing the security algorithms through experiments over large number of data sets and, (iv) to use parallelism to demonstrate more energy-efficient algorithms applicable to portable and mobile devices.

2.2 System Planning

The RSA algorithm is a very secure, but slow, as it contains heavy mathematical computations, thus parallelizing this algorithm will make it perform faster and may increase the security level provided by this algorithm by increasing the key size. RSA now uses keys in 1024 bit length which are not considered secure enough. However, the longer the key used the more security and the slower the performance is obtained. This leads to the need to use parallel computing to increase the security level without adversely affecting the performance.

Implementation Details :

```
Procedure Common_Divisors (vector1, vector2)
Begin
#pragma omp parallel for omp_num_threads(n) // spawning for loop over n threads
for i=1 to length of vector1 and i incremented
by n
```

```
If (i position in vector1 equals in
vector2)
Store in common divisor
vector.
end for
End
```

2.3 Requirements

The RSA algorithm holds the following features −

- RSA algorithm is a popular exponentiation in a finite field over integers including prime numbers.
- The integers used by this method are sufficiently large making it difficult to solve.
- There are two sets of keys in this algorithm: private key and public key.

We will have to go through the following steps to work on RSA algorithm –

I. Step 1: Generate the RSA modulus

The initial procedure begins with selection of two prime numbers namely p and q, and then calculating their product N, as shown −

$N=p*q$

Here, let N be the specified large number.

II. Step 2: Derived Number (e)

Consider number e as a derived number which should be greater than 1 and less than (p-1) and (q-1). The primary condition will be that there should be no common factor of (p-1) and (q-1) except 1

III. Step 3: Public key

The specified pair of numbers n and e forms the RSA public key and it is made public.

IV. Step 4: Private Key

Private Key d is calculated from the numbers p, q and e. The mathematical relationship between the numbers is as
follows −

$ed = 1 \bmod (p-1) (q-1)$

The above formula is the basic formula for Extended Euclidean Algorithm, which takes p and q as the input parameters.

2.4 System Requirements

2.4.1 Software Requirements :

To run our code in a system, it should complete the following requirements :

1. Python 3.x
2. GCC 7.4 and above

2.4.2 Hardware Requirements :

The processor can be multiple core or a single core. The processor should not be single threaded. The performance of all the programs have been tested on Intel i5 8th Generation processors.

Chapter 3

System Design

The aim is to reduce the total computation time needed to complete the process by parallelizing the algorithm using the concept of multi-threading. The complexity, data structures and the overall procedure of the algorithm would remain constant.

The approach is to develop a code for the serial implementation of the RSA algorithm and then its execution. This is needed so as to compare it with the results after the parallelisation of the algorithm.

The architecture diagram for the project can be drawn as :

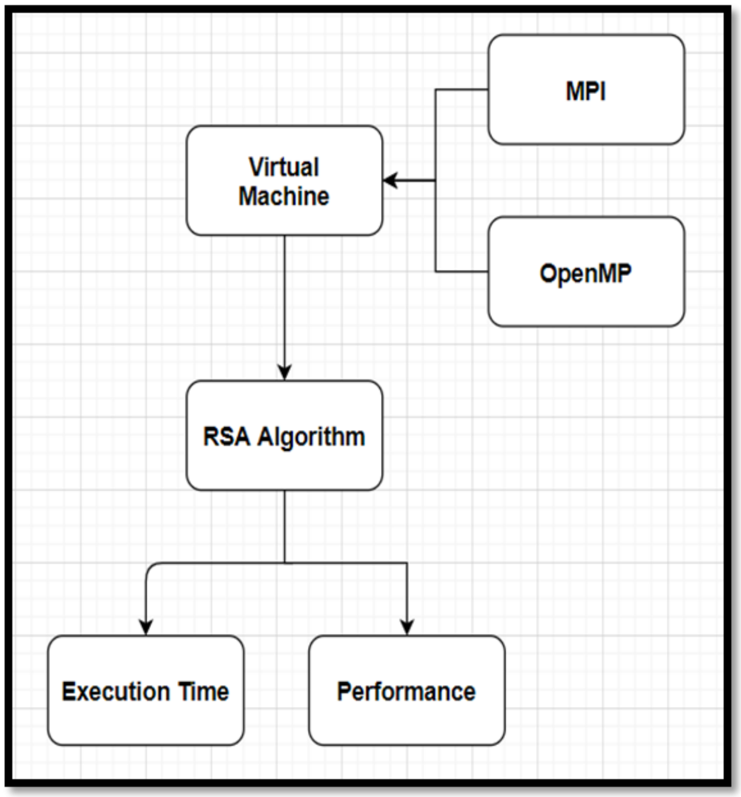


Figure 1: Architecture Diagram

Chapter 4

Implementation of System

The algorithm attempts to distribute an equal amount of work to processors while decrypting the RSA key quickly and efficiently.
The algorithm was designed to address several important observations:

1. The number of primes below some integer x can be approximated as x/log(x). This implies that the gap between primes increases with the natural logarithm of the integer.
2. Finding the next prime number higher than some integer becomes increasing difficult as the integer becomes larger.
3. The prime factorization of a number is unique. This implies that if the given key (n) is divisible by some prime p, then the decryption primes are p and n/p.
4. It is only necessary to determine if any prime in the range [2, sqrt(n)] is divisible by n. The rest of the primes in [sqrt(n), n] will be complements in the division of n.
5. If one processor finds the keys, it must signal all other processors that the key has been found in order to prevent the other processors from doing redundant work.

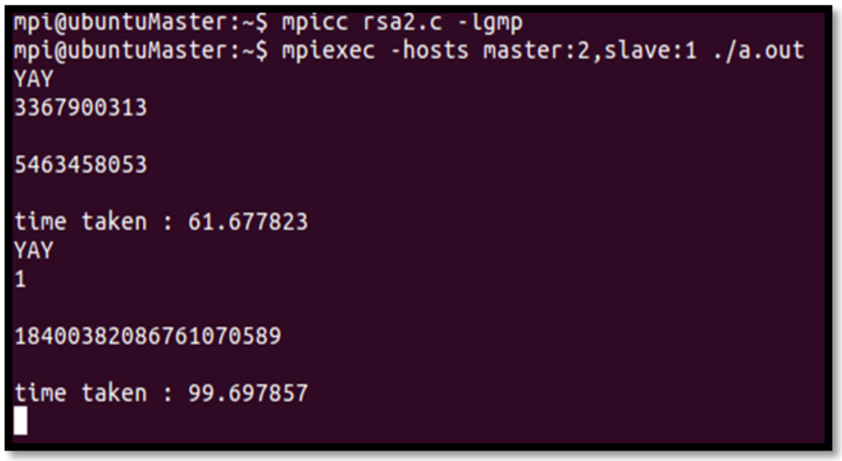
Next, we create an environment for the parallel computing, which is done with the help of two OS, namely master and slave. They help us in creating a cluster. We take the help of the given algorithm to code an OpenMP code of requirement for parallelising the RSA algorithm upon which further MPI techniques will be used.

```
Procedure Divisors (n1, n2, r) // where r= maximum (sqrt (n1), sqrt
(n2))
Begin
#pragma omp parallel
for omp_num_threads (n) // spawning for loop
over n threads
// nowait clause allows the threads to execute two if conditions
simultaneously
#pragma omp nowait
for i=1 to r do
if (n1 mod i)
store result of n1 mod i in vector1.
If (n2 mod i)
store result of n2 mod i in
vector2.
end for
end for
End
```

Chapter 5

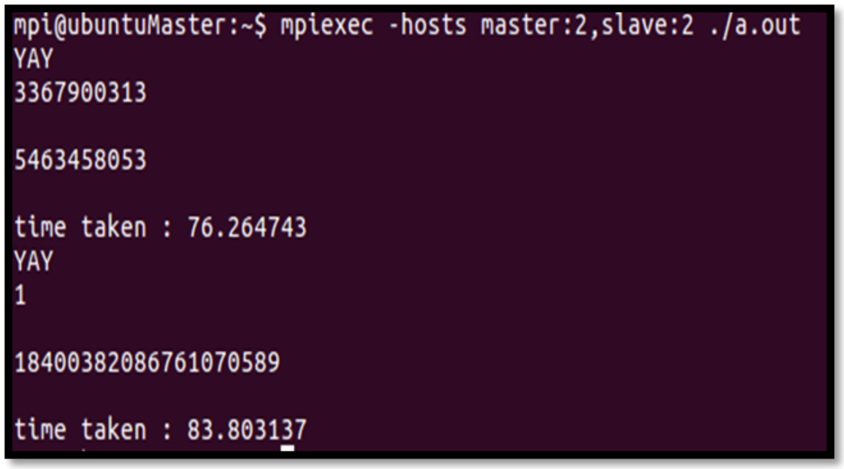
Results and Discussions

First, an environment is set up using a master and a slave OS on a virtual machine to create a cluster using MPI programming. Both the OS are connected and the programming is done on the master OS along with the implementation.
Now, we run the program and subsequently increase the number of slaves, in technical terms – nodes. The greater the number of slaves, the greater is the speed as it is subsequently divided into that number of tasks.



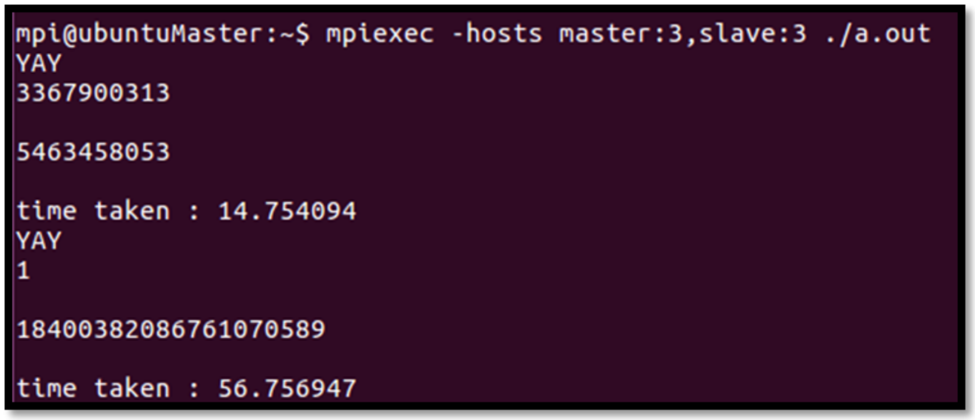
```
mpi@ubuntuMaster:~$ mpicc rsa2.c -lgmp
mpi@ubuntuMaster:~$ mpiexec -hosts master:2,slave:1 ./a.out
YAY
3367900313

5463458053

time taken : 61.677823
YAY
1

18400382086761070589

time taken : 99.697857
```

Figure 2: Slaves = 1



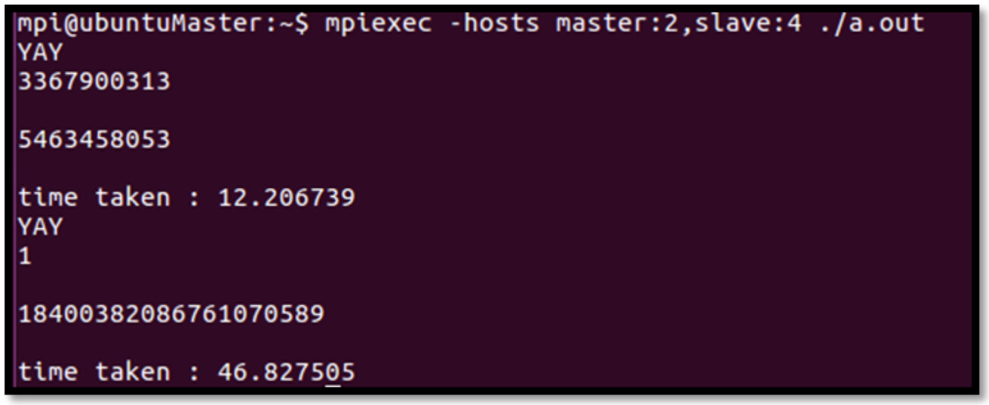
```
mpi@ubuntuMaster:~$ mpiexec -hosts master:2,slave:2 ./a.out
YAY
3367900313

5463458053

time taken : 76.264743
YAY
1

18400382086761070589

time taken : 83.803137
```

Figure 3: Slaves = 2



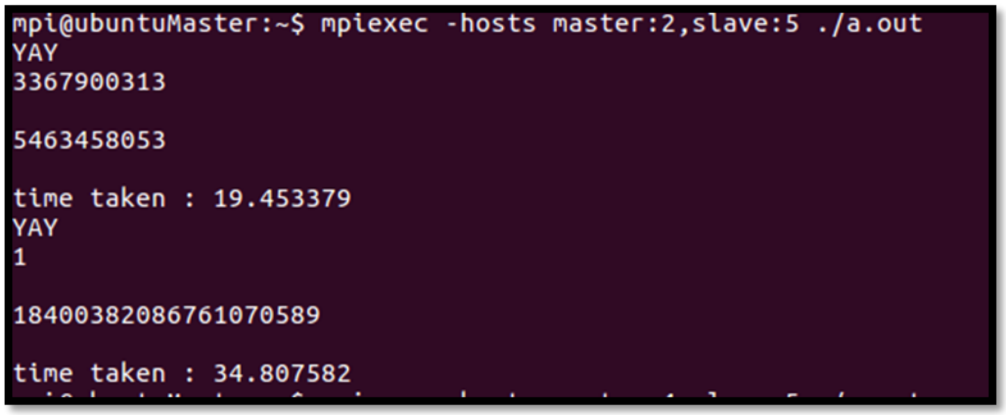
```
mpi@ubuntuMaster:~$ mpiexec -hosts master:3,slave:3 ./a.out
YAY
3367900313

5463458053

time taken : 14.754094
YAY
1

18400382086761070589

time taken : 56.756947
```

Figure 4: Slaves = 3



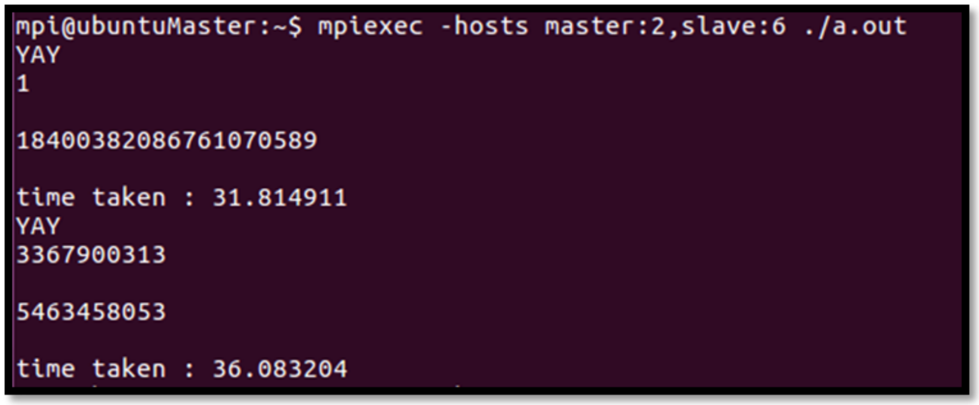
```
mpi@ubuntuMaster:~$ mpiexec -hosts master:2,slave:4 ./a.out
YAY
3367900313

5463458053

time taken : 12.206739
YAY
1

18400382086761070589

time taken : 46.827505
```

Figure 5: Slaves = 4



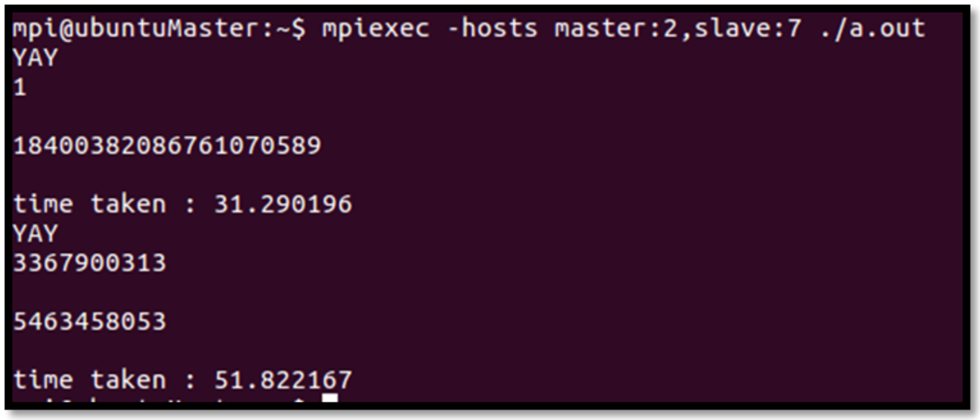
```
mpi@ubuntuMaster:~$ mpiexec -hosts master:2,slave:5 ./a.out
YAY
3367900313

5463458053

time taken : 19.453379
YAY
1

18400382086761070589

time taken : 34.807582
```

Figure 6: Slaves = 5



```
mpi@ubuntuMaster:~$ mpiexec -hosts master:2,slave:6 ./a.out
YAY
1

18400382086761070589

time taken : 31.814911
YAY
3367900313

5463458053

time taken : 36.083204
```

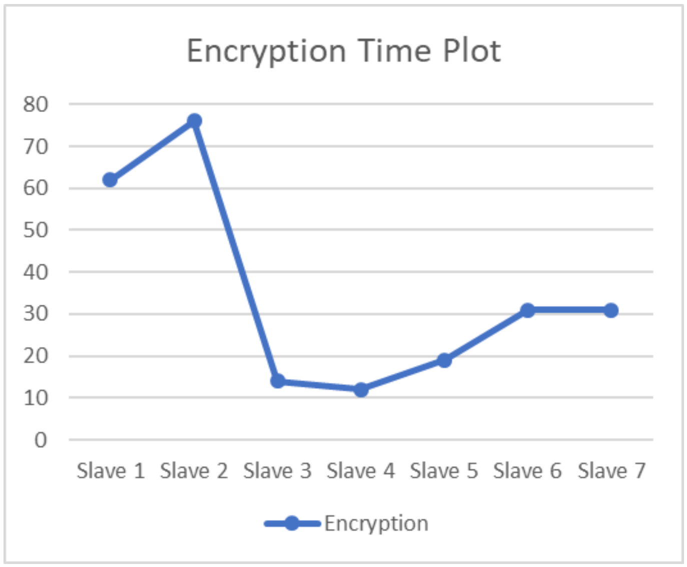
Figure 7: Slaves = 6



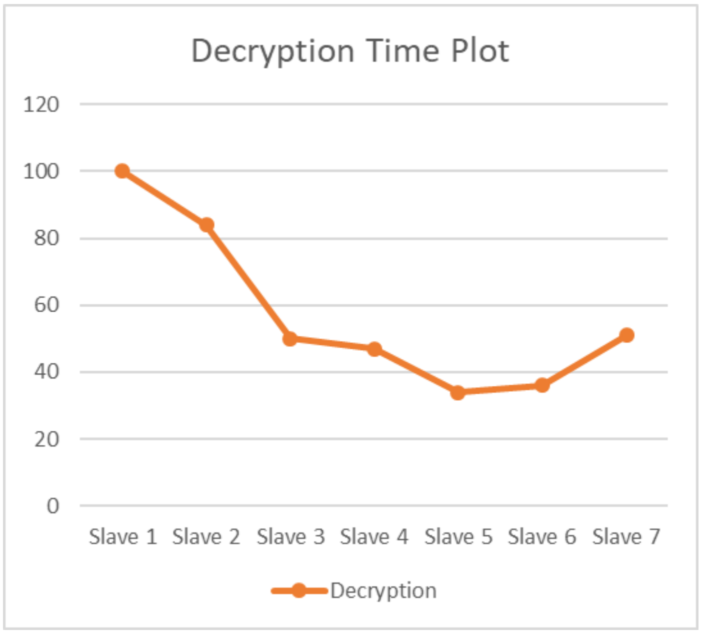
```
mpi@ubuntuMaster:~$ mpiexec -hosts master:2,slave:7 ./a.out
YAY
1

18400382086761070589

time taken : 31.290196
YAY
3367900313

5463458053

time taken : 51.822167
```

Figure 8: Slaves = 7

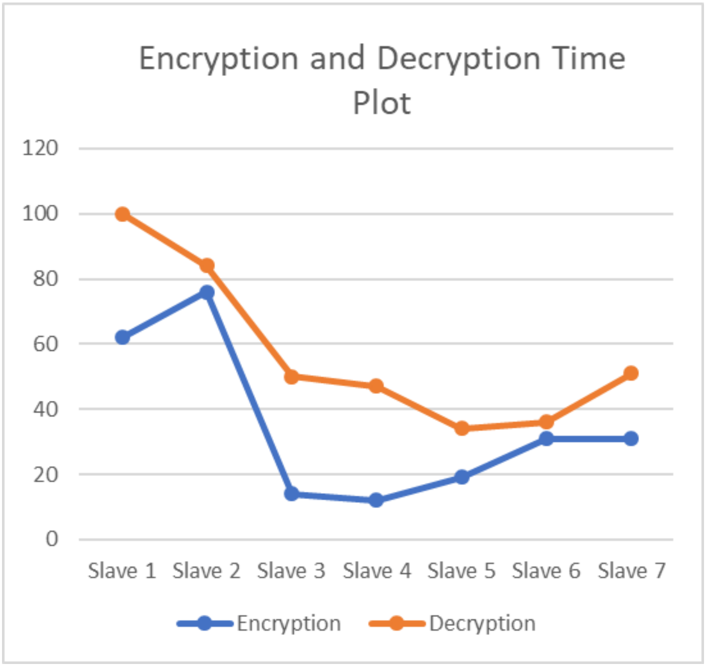
We observe that after we increase the number of slaves after 2, there is a significant decrease in the processing time of encryption. Also, the subsequent execution time for decryption also decreases.
On the other hand we see that as we reach the number of slaves at 6 and 7, there is again a jump in time. This is because as we increase the number of slaves, the time for dividing the processes takes up significant time even though the execution time is less.
We can make our point more clear by visualization through the following graphs :



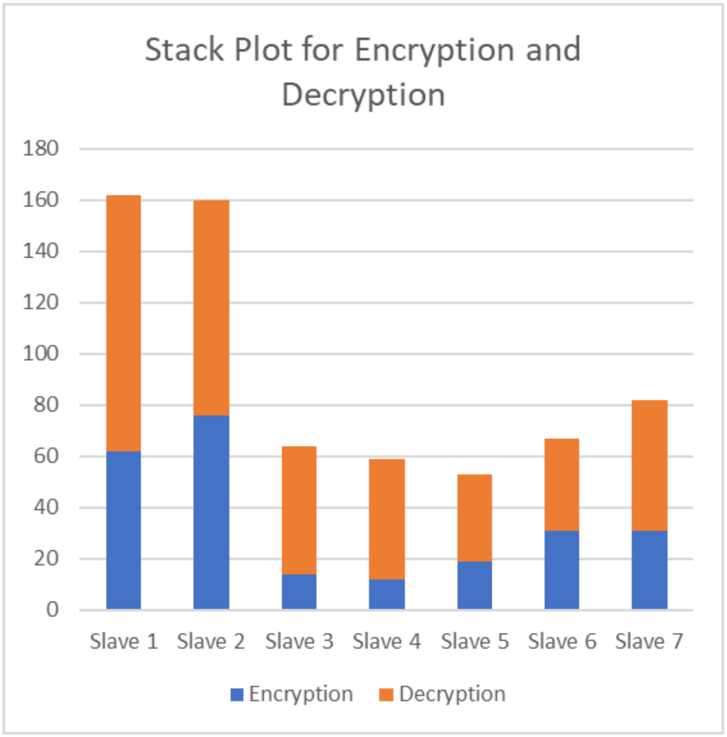
The above graph shows the decrease in encryption time as we increase the number of slaves from 1 to 5, but it again increases after 6 up to 7.



The above graph shows the decrease in decryption time as we increase the number of slaves from 1 to 5, but it again increases after 6 up to 7.



From the above graph we can compare that both – the encryption and decryption times are brought to a significant decrease. But after 6th and 7th slave, they increase again as explained earlier.



This graph helps us compare not only the decrease in encryption and decryption time, but also the difference between the both. We see that this also decreases along with increasing the number of slaves.

Chapter 5

Conclusion and Future Work

We were successful in enhancing the efficiency of parallelization of RSA Algorithm and executed it in less time using MPI Cluster.

Advantages of parallelization

- Less execution times
- Efficient code
- Distribution of instructions to threads automated

Disadvantages

- Overhead cost outweighs the time saved in case of small strings
- Memory usage multiplies by the number of threads/processors used
- Weight of performance on processor increases exponentially with the increase in number of processors

Future Study

With the hope of better parallelization algorithms and improvement in the encryption algorithms, this process can be accelerated much faster than its usual speed. Data dependency among the steps must be made a little less complicated so that they are easy to be modularized into various components.

Certain algorithms must be developed that actually parallelize the core components of these algorithms.

References

[1] Wenjun Fan, and Xudong Chen, “Parallelization of RSA Algorithm Based on Compute Unified Device Architecture”, Conference: GCC 2010, The Ninth International Conference on Grid and Cloud Computing, Nanjing, Jiangsu, China, 1-5 November 2010.

[2] Mohammed Issam Younis, Heba Mohammed Fadhil, and Zainab Nadhim Jawad, “Acceleration of the RSA Processes based on Parallel Decomposition and Chinese Reminder Theorem”, International Journal of Application or Innovation in Engineering & Management (IJAIEM) Web Site: www.ijaiem.org Email: editor@ijaiem.org Volume 5, Issue 1, January 2016 ISSN 2319 - 4847.

[3] Rahul Saxena, Dushyant Singh, Ashutosh Kushwah, and Monika Jain, “An enhanced parallel version of RSA public key crypto based algorithm using openMP”, SIN '17: Proceedings of the 10th International Conference on Security of Information and Networks, October.

[4] Balkees Mohamed Shereek, Zaiton Muda, and Sharifah Yasin, “Improve Cloud Computing Security Using RSA Encryption WithFermat’s Little Theorem”, IOSR Journal of Engineering (IOSRJEN) ISSN (e): 2250-3021, ISSN (p): 2278-8719 Vol. 04, Issue 02 (February. 2014), V6 PP 01-08.

[5] Manisha N. Kella and Sohil Gadhiya, “A Survey on AES (Advanced Encryption Standard) and RSA Encryption-Decryption in CUDA”, 2018 IJSRSET | Volume 4 | Issue 4 | Print ISSN: 2395-1990 | Online ISSN : 2394-4099.

[6] Ahsan Ayub, Zishan Ahmed Onik and Steven Smith, “Parallelized RSA Algorithm: An Analysis with Performance Evaluation using OpenMP Library in High Performance Computing Environment”, Conference: 22nd International Conference of Computer and Information Technology (ICCIT) At: Dhaka, Bangladesh.

[7] Heba Mohammed Fadhil and Mohammed Issam Younis, “Parallelizing RSA Algorithm on Multicore CPU and GPU”, International Journal of Computer Applications (0975 – 8887) Volume 87 – No.6, February 2014.

[8] Yunefi Li, Qing Liu and Tong Li, “Design and Implementation of an Improved RSA Algorithm”, 2010 International Conference on E-Health Networking, Digital Ecosystems and Technologies.

[9] Deepali and Namita Kakkar, “Hybrid Parallel Multithreading Encryption”, International Journal of Computer Sciences and Engineering Volume-3, Issue-6 E-ISSN: 2347-2693.

[10] Dr. Prerna Mahajan and Abhishek Sachdeva, “A Study of Encryption Algorithms AES, DES and RSA for Security”, Global Journal of Computer Science and Technology Network, Web & Security Volume 13 Issue 15 Version 1.0 Year 2013 Type: Double Blind Peer Reviewed International Research Journal Publisher: Global Journals Inc. (USA) Online ISSN: 0975-4172 & Print ISSN: 0975-4350.

[11] M. Preetha and M. Nithya, “A study and performance analysis of RSA algorithm”, International Journal of Computer Science and Mobile Computing A Monthly Journal of Computer Science and Information Technology ISSN 2320–088X IJCSMC, Vol. 2, Issue. 6, June 2013, pg.126 – 139.

[12] Kalpesh S. Prajapati and Prof. B. V. Buddhdev, “Overview of Improvements and Modifications in RSA Algorithm”, Journal of Emerging Technologies and Innovative Research (JETIR) Dec 2014 (Volume 1 Issue 7) JETIR (ISSN-2349-5162).

[13] Rohit Minni, Kaushal Sultania, Saurabh Mishra and Prof Durai Raj Vincent, “An Algorithm to Enhance Security in RSA”, 4th ICCCNT 2013 July 4-6, 2013, Tiruchengode, India.

[14] Shaheen Saad Al-Kaabi, Hamad Bin Khalifa University and Samir Brahim Belhaouari, “Methods toward Enhancing RSA Algorithm: A Survey”, International Journal of Network Security & Its Applications (IJNSA) Vol. 11, No.3, May 2019.

[15] Israt Jahan, Mohammad Asif and Liton Jude Rozario, “Improved RSA cryptosystem based on the study of number theory and public key cryptosystems”, American Journal of Engineering Research (AJER) e-ISSN : 2320-0847 p-ISSN : 2320-0936 Volume-4, Issue-1, pp-143-149.

[16] Devashish Vaghela and Prof.Rajyalakshmi Jaiswal, “Modified Key Based Image Encryption using RSA algorithm”, International Journal of Innovative Research in Technology (An International Open Access Journal & and ISSN Approved) IJIRT**E**XPLORE- Search Thousands of research papers Call For Paper October 2020 Last Date 25 - October 2020.

[17] Shaheen Saad Al-Kaabi and Samir Brahim Belhaouari, College of Science and Engineering, Hamad Bin Khalifa University (HBKU), Doha, Qatar, “A survey on enhanced RSA algorithms”, Dhinaharan Nagamalai et al. (Eds) : WIMO, ICAIT, ICDIPV, NC, CRYPIS - 2019 pp. 123-142, 2019. © CS & IT-CSCP 2019 DOI: 10.5121/csit.2019.90411.

[18] P. Saveetha and S. Arumugam, “Study on improved RSA algorithm and its implementation”, International Journal of Computer and Communication Technology: Vol. 7 : Iss. 3 , Article 9.

[19] Ahmed Eskander Mezher, “Enhanced RSA Cryptosystem based on Multiplicity of Public and Private Keys”, International Journal of Electrical and Computer Engineering (IJECE)Vol.8, No.5, October2018, pp. 3949~3953ISSN: 2088-8708, DOI: 10.11591/ijece.v8i5.pp3949-3953 .

[20] Fausto Meneses, Walter Fuertes, José Sancho, Santiago Salvador, Daniela Flores, Hernán Aules, Fidel Castro, Jenny Torres Alba Miranda and Danilo Nuela, "RSA Encryption Algorithm Optimization to Improve Performance and Security Level of Network Messages", IJCSNS International Journal of Computer Science and Network Security, VOL.16 No.8, August 2016.

[21] Shaheen Saad Al-Kaabi and Samir Brahim Belhaouari, "Methods toward Enhancing RSA Algorithm: A Survey", International Journal of Network Security & Its Applications (IJNSA) Vol. 11, No.3, May 2019.

[22] Israt Jahan, Mohammad Asif and Liton Jude Rozario, "Improved RSA cryptosystem based on the study of number theory and public key cryptosystems", American Journal of Engineering Research (AJER) e-ISSN : 2320-0847 p-ISSN : 2320-0936 Volume-4, Issue-1, pp-143-149.

[23] Bodake Vijay and Gawande R.M., "A Review on An Encryption Engines For Multi Core Processor Systems", IOSR Journal of Electronics and Communication Engineering (IOSR-JECE) e-ISSN: 2278-2834, p-ISSN: 2278-8735 PP 38-46.

[24] Shweta Kumari and Abhishek Kumar, "Encryption Algorithms - Parallelization", International Journal of Computer Science Trends and Technology (IJCST) – Volume 2 Issue 4, Jul-Aug 2014.

[25] Sonam Mahajan and Maninder Singh, "Analysis of RSA Algorithm using GPU Programming".

[26] https://ieeexplore.ieee.org/document/7164703 - The cryptography algorithms are divided into two parts symmetric and asymmetric. There are many different challenges to implement cryptography algorithm specially throughput in terms of time execution. In this paper, we study and analyze the performance of different cryptographic algorithm on multicore processors and also we explore the performance in sequential and parallel implementation of cryptography algorithm on multi core processors.

[27] U. Thirupalu, Dr. E. Kesavulu Reddy and E. Spandhana, "Security Analysis of Cryptographic Algorithms in Cloud Computing", Paper ID : IJERTV7IS100089 Volume & Issue : Volume 07, Issue 10 (October – 2018) Published (First Online): 05-01-2019 ISSN (Online) : 2278-0181 Publisher Name : IJERT.

[28] Sapna Saxena, "Parallel algorithms for public key infrastructure based security techniques", International Journal of Security and Networks. (ISSN print: 1747-8405) Publisher: Inderscience.

Appendix

```
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <gmp.h>
#include "mpi.h"
int main (int argc, char *argv[]){
    int rank,size;
  unsigned long int  i;
  int  m;
  char  res [300], sho [300];
  MPI_Init(&argc, &argv);
  double start = MPI_Wtime();
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Comm_size(MPI_COMM_WORLD, &size); /* initializing MPI, collecting info
about environment*/
  mpz_t   N;
  mpz_t   D,S,I,ds,Ss,Se;
  mpz_init(D);
  mpz_init(S);
  mpz_init(Ss);
  mpz_init(Se);
  mpz_init(ds);
    mpz_init(I);
    mpz_init_set_str (N, "18400382086761070589", 10);
  mpz_sqrt(S,N);
  mpz_cdiv_q_ui(ds,S,size); /* divvy up lims for loop! */
  mpz_mul_si(Se,ds,rank);
  mpz_add(Ss,Se,ds);
    if ((mpz_divisible_ui_p(Ss,2)) != 0){
        mpz_add_ui(Ss,Ss,1);
    }

  for (mpz_set(I,Ss);mpz_cmp(I,Se)>0;mpz_sub_ui(I,I,2)) {
    m = mpz_divisible_p(N,I);
    if ((mpz_divisible_ui_p(I,100000001)) != 0){
    mpz_get_str(sho,10,I);
    //printf("%s\n",sho);
    }
    if (m != 0){
         mpz_get_str(sho,10,I);
    printf("YAY\n%s\n",sho);
    mpz_cdiv_q(D,N,I)    ;
    mpz_get_str(res,10,D);
    printf("\n%s\n", res);
    double end = MPI_Wtime()-start;
    printf("\ntime taken : %f\n",end);
    MPI_Finalize(); /* Tell all threads to exit(1)*/
    }
  }
  return 0;
}
```