Optimization of RSA encryption and decryption process with distributed computing method

S N Arief*, V A H Firdaus, and A Prasetyo
Information Technology Department, State Polytechnic of Malang, Indonesia

*sofyan@polinema.ac.id

Abstract. Mathematics equation of the RSA is the complicated one between the other asymmetric encryption methods. It makes the RSA method needs more resources than the others. Most existing studies are trying to make another encryption method that need less computational resources, but it is less secure than using the RSA algorithm. In this paper, we proposed the Distributed Computing method to accommodate the limited computing resources in a single worker. We will distribute the computational process of RSA encryption and decryption to numerous workers. The test results showed that the implementation of Distributed Computing method made the encryption and decryption process finished faster. In conclusion, it increased completion time, made the encrypted data serve faster, and maximized the computational resources.

1. Introduction
In this digital era, document storing are no longer uses the media of each personal computer. But using a centralized storage area. Because it uses a centralized store area, security issue of these documents is most discussed widely. If the contents of the document can be read by an unauthorized parity, confidentiality of the document will be disturbed. So, the document needs some encryption mechanism that can protect a document that made it is not easily to read by unauthorized parties [1].

There are many methods that can be applied to encrypt documents. When viewed from the type of key used to encrypt and decrypt, it can be divided into two types, namely symmetric key encryption and asymmetric key encryption. Encryption with a symmetric key uses the same key to encrypt and decrypt. Whereas encryption with asymmetric key uses different keys to encrypt and decrypt [2]. Encryption with asymmetric keys is safer when compared to symmetric key encryption. Encryption with asymmetric keys itself has many kinds of algorithm. The most popular asymmetric key encryption algorithm is RSA [3]. This algorithm is the most popular method because it is the most difficult to hack. The difficulty in hacking this algorithm is due to the complexity of the mathematical equations used in the algorithm [4]. If more complicated mathematical equations are used, much computational resources are used to process it. Comparing with the other algorithm, RSA algorithm is the greediest computational power needs [5]. Because of that, RSA algorithm is the slowest one on completion time comparing with the other encryption and decryption algorithm [6].

In this paper we proposed new method to accommodate computational problem of an RSA algorithm. As we know that computing RSA in single computer need more time to complete. So, we proposed to use distributed computing method to compute RSA algorithm in some part of their sequence. Our focus work is in encryption and decryption process in RSA algorithm. The main idea of this method is to
distribute computational task of encryption and decryption process in a set of computers. Then, that computer workers will running the task parallelly [7]. So, with this method encryption and decryption completion time will be faster.

2. Methods

![Diagram of Encryption Process](image1)

**Figure 1.** Proposed method steps.

In this paper we proposed new method (Figure 1) to accommodate computational problem of an RSA algorithm that made this algorithm need more time to complete [8]. We proposed to use distributed computing method to compute RSA algorithm in some part of their sequence [9]. Our focus work is only in encryption and decryption process of RSA algorithm. The main idea is distributing computational task of encryption and decryption process to a set of computers [10].

![Diagram of Decryption Process](image2)

**Figure 2.** Illustration of proposed method.

File that will be encrypt or decrypt are will be splitted into some part depending of used key size. This process is handled by a single computer that we called broker. After that, splitted file (later we called pieces) and the generated key, are distributed to a set of computers (later we called worker). Then every worker encrypts or decrypts that pieces parallelly with the same generated key. After that, broker merge encrypted or decrypted pieces into a completed file. Illustration of our proposed method are shown on the Figure 2 above.

We test our proposed method using isolated environment that running on the top of VMware vSphere Hypervicro (EXSi) 6.7 bare metal. This bare metal are hosted on our server that have 1,6Ghz Six Cores Processor with 20GB of RAM and equipped with 2TB SATA 3Gb/s 7200RPM HDD. We used six virtual machines to simulate one broker and five workers. Although using virtual machine, we can guarantee that it is same with using the real PC. Because we have enabling hardware virtualization and input/output virtualization features on every virtual machine. Every virtual machine has an identical specification to simplify the number of resources they had. Every virtual machine has 1,6Ghz Single Core CPU with 1GB of RAM. Not only that, this virtual machine is equipped with 32GB of network attached storage (NAS) to make distributing working files easily. Figure 3 below describe the testbed that we build.
We create a simple Python program to implement our method. We used PythonXMLRPC for calling workers to do their jobs. For the RSA algorithm, we modified Python Encryption Library (Crypto). This Python program is running on every PC used as a daemon. Many sizes of files were used for testing, which are 1MB, 5MB, 10MB, 50MB, and the last 100MB. Every testing file must be tested for encryption and decryption processes. From the testing, we can get information about the completion time for each tested file. After that, we compare that completion time with the completion time of a single computer method. But we made this single computer equally have a same specification with the total specification that the distributed computer owned.

3. Results and discussion

Testing on the testbed, carried out to get the results of research. The code implementation of our proposed method has three crucial parts. First parts are splitting the input file process. In this process, we split the input file by the number of available workers. We implement this process by using standard Unix splitting command and store its file into a shared directory (we used NFS for this). After it, we move to the second parts. The second parts are distributing pieces of files to encryption/decryption processes. This process is implemented using PythonXMLRPC for calling the worker to do their job. After called by the broker, the worker will do encryption/decryption processes using the defined key and their pieces of files. Figure 4 below shown the code implementation for this process.

![Testbed Diagram](image-url)
The last crucial parts of our proposed method is merge output encrypted/decrypted file from the worker. For this parts, we use a standard Unix merge command to merge the file. The Implementation of this part are shown on the Figure 5 below.

We run our script to get the result. And we have made a log file of our processing steps to get the processing. Figure 6 below describe how our code running and generating the log file.
Figure 6. Running scripts and generated log.

From the log of testing process we get information about completion time for every testing file. The result of each process are shown on the Table 1 and Table 2 below.

Table 1. Encryption testing result.

| File Size | 1MB | 5MB | 10MB | 50MB | 100MB |
|-----------|-----|-----|------|------|-------|
| Processing Time | 2   | 6   | 12   | 58   | 115   |
| Proposed Method | Old Method | 6   | 28   | 55   | 277   | 552   |
| Diff | 4   | 22  | 43   | 219  | 437   |

Table 2. Decryption testing result.

| File Size | 1MB | 5MB | 10MB | 50MB | 100MB |
|-----------|-----|-----|------|------|-------|
| Processing Time | 8   | 35  | 69   | 346  | 691   |
| Proposed Method | Old Method | 35  | 171  | 341  | 1708  | 3441  |
| Diff | 27  | 136 | 272  | 1362 | 2750  |

4. Conclusion

The test results (Figure 7) showed that the implementation of Distributed Computing method made the encryption process finished 3 times faster at our minimum tested files. In 1MB files, our proposed method has finishing the job in 2 seconds while the single method needs 6 seconds. Not only that, encryption process finish rate is increased on a larger file size. For 100MB files, our proposed method has finishing the job in 115 seconds while the single method needs 552 seconds (it’s about 4,8 times faster). For the decryption process, our proposed method finished their job 4,4 times faster at minimum tested files. In 1MB files, our proposed method has finishing the job in 8 seconds while the single method needs 35 seconds. Decryption process finish rate are increased too on a larger file size. For 100MB files, our proposed method has finishing the job in 691 seconds while the single method needs 3441 seconds (it’s about 5 times faster). In conclusion, our proposed method has been increased completion time and made the encrypted data serve faster.
Figure 7. Graph of testing result.

Acknowledgment
The authors would like to thank to the Indonesian Directorate General of the Higher Education (DIKTI) and State Polytechnic of Malang who have supported this research project.

References
[1] Buchanan W J 2017 Cryptography
[2] Munir R 2008 Pengantar Ilmu Kriptografi Penerbit Andi
[3] Rivest R L, Shamir A and Adleman L 1983 A Method for Obtaining Digital Signatures and Public-Key Cryptosystems Communications of the ACM
[4] Singh G and Supriya S 2013 A Study of Encryption Algorithms (RSA, DES, 3DES and AES) for Information Security International Journal of Computer Applications
[5] Hercigonja Z 2016 Comparative Analysis of Cryptographic Algorithms International Journal of DIGITAL TECHNOLOGY & ECONOMY
[6] Bhanot R and Hans R 2015 A review and comparative analysis of various encryption algorithms International Journal of Security and its Applications
[7] Diaz J, Muñoz-Caro C and Niño A 2012 A survey of parallel programming models and tools in the multi and many-core era IEEE Transactions on Parallel and Distributed Systems
[8] Amalarethinam I G and Leena H M 2017 Enhanced RSA Algorithm with Varying Key Sizes for Data Security in Cloud Proceedings - 2nd World Congress on Computing and Communication Technologies, WCCCT 2017
[9] Khanzaei N and Hanapi Z M 2014 A framework based on RSA and AES encryption algorithms for cloud computing services Proceedings - 2014 IEEE Conference on System, Process and Control, ICSPC 2014
[10] Jadeja Y and Modi K 2012 Cloud computing - Concepts, architecture and challenges 2012 International Conference on Computing, Electronics and Electrical Technologies, ICCEET 2012