Optimization of Distributed RSA Encryption and Decription Processing Using Process Scheduling Method In Single Board Computer Cluster Architecture (SBC)

Sofyan Nur Arief[1], Vipkas Al Hadid Firdaus[2*], Arief Prasetyo[3]

[1,2,3]*Department of Information Technology, Politeknik Negeri Malang*
[*]*vipkas@polinema.ac.id*

ABSTRACT

Data security is still a major issue regarding the need for data confidentiality. The encryption process using the RSA algorithm is still the most popular method used in securing data because the complexity of the mathematical equations used in this algorithm makes it difficult to hack. However, the complexity of the RSA algorithm is still a major problem that hinders its application in a more complex application. Optimization is needed in the processing of this RSA algorithm, one of which is by running it on a distributed system. In this paper, we propose an approach with a FIFO process scheduling algorithm that runs on a single board computer cluster. The test results show that the allocation of resources in a system that uses a FIFO process scheduling algorithm is more efficient and shows a decrease in the overall processing time of RSA encryption.

Keywords: Encryption, RSA, FIFO.

1. INTRODUCTION

Data security is still a popular issue to research. This is inseparable from the need for data confidentiality. With the advancement of technology as it is today, a data (containing important information) can be stolen by unauthorized parties if it is not secured. One way to secure a data is to perform an encryption process. The RSA algorithm is still the most popular method used. Difficulty in hatching this algorithm is the main reason for the use of this algorithm in various use cases [1]. Themathematical equations used in this algorithm make it difficult to hack. However,the more complex mathematical equations used, the greater the computational resources needed to process them. Rsa algorithm also has characteristics where it has good security even near perfect,key management is easy, easy to implement and easy to understand. But in this algorithm, modular power has a large part in influencing the performance of the algorithm, and becomes the main problem that inhibitsits applicationto a more complex application Therefore, the rapid processing of rsa encryption algorithms (included) has been the focus in various studies [2].

Optimization in the rapid processing of RSA algorithms can be done in a variety of ways. One of them is by running it on a distributed system. In a distributed system, a job is shared and executed on a set of computing resources. A set of computing resources will communicate with each other and collaborate in processing a job [3] . In previous research [4], a set of computers in the form of virtual computers was used to process encryption using rsa algorithms. In the study, an encryption work was distributed to each worker's computer. The division of encryption work is distributed evenly to each worker. The results of the study prove that distributed processing can increase the processing speed of RSA encryption [12]. But a new problem arises when the use of a virtual computer is replaced with a set of physical computers. The use of physical computers in large quantities is very inefficient both in terms of the cost of purchasing tools and in terms of operational costs. Therefore, it requires a cheaper computing resource and low operational costs.

At this time, technological advances are making the development of computational processing tools become increasingly developed. *Single Board Computer* (SBC) also began to be used to replace classical computing processing tools that are generally a personal computer in the form of laptops and workstations [9]. In a study conducted by Sukoco [5], the implementation of a single SBC Raspberry Pi 3 Model B+ in the case of weed detection on farmland was able to outperform personal computer performance (x86-based intel core i5) by 0.04 times faster [11]. In addition, research conducted by Irfi [7] also proved a significant improvement in the implementation of the SBC cluster architecture on *machine-learning* processing when compared to processing on a single SBC. Therefore, a hypothesis arises that the SBC cluster can be a cheap and economical RSA encryption processing solution in terms of device purchase costs as well as operational. The hypothesis proved correct in research conducted by [6]. The use of the SBC cluster in the RSA encryption process can outperform a single *multicore* personal computer at the same device purchase cost. The operating costs of the SBC cluster are also much more efficient when compared to a single *multicore* personal computer.

A new problem arises when the task of doing encryption work comes along simultaneously. Uncontrolled work processes can cause data encryption damage that leads to endless processing (continuous processing) on workers' computers (either single personal computers or SBC). Continuous and never-ending processing can lead to a complete decline in encryption processing performance. In addition, it can cause damage to the computer device used to process the encryption.

Process scheduling is an approach that can be used to solve these problems. With process scheduling, a process will be run in a structured and controlled manner. So that will minimize the existence of endless processing. There are many process scheduling algorithms that currently exist, such as FIFO *(First In First Out),*LILO (Last*In Last Out),*LIFO (Last*In First Out)* [10]. The process scheduling algorithm itself is done by forming a queue (or commonly called *a queue).* In the FIFO algorithm, the queue process is created by laying the work / process based on the arrival time. So that the work / process received at the time (t)-1 will be processed first compared to the work / process received at the time (t). Almost the same approach is applied to the LILO algorithm. A queue will be created from behind on the LILO algorithm. Thus, the work / process received at the time (t) will be processed after the work / process is received at the time (t)-1. The LIFO algorithm uses a different approach. Queues will be created based on the process / work that comes in late. In this algorithm, the work/process received at the time (t) of the second will be delayed processing when a new work/process is received at time (t)+1. And it will be resumed when the work or process received at the time (t)+1 is completed [8].

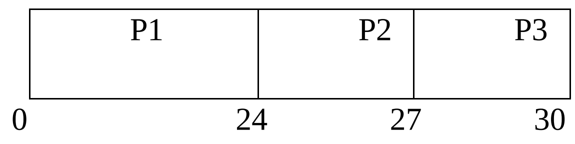
A hypothesis was put forward by researchers to answer the problem of encryption processing that comes almost simultaneously, namely by applying process scheduling. In the case of RSA encryption on the SBC cluster architecture, the most likely scheduling algorithm is the FIFO*(First In First Out)*algorithm. The selection of this algorithm is based on the waiting time resulting from faster FIFO processing compared to LIFO. In the LIFO algorithm, the ongoing encryption work/process will be temporarily suspended and will resume after the last received work/process has been processed. So that the waiting time for work / process that comes at the time (t)-1 will be longer than the work / process that comes at the time (t). This can increase the overall rsa encryption processing time. While in the FIFO algorithm, the process that is running (work that comes at time (t)-1) will still run without being disturbed by the work that comes at the time (t). Work that comes at the time (t) will be entered in the queue and processed after the work that comes on (t)-1 is completed. So the waiting time needed for the process to come in (t)-1 will be faster when compared to the LILO algorithm. And the impact is a decrease in the overall rsa encryption processing time. Therefore, in this study the FIFO algorithm will be applied, tested, and analyzed to test the hypothesis that has been put forward by the author.

2. METHODS

An idea was put forward to address the problem of not yet maximal processing of RSA encryption and decryption in distributed systems applied to SBC clusters. The idea proposed in this study is to add a process scheduling mechanism, so that it can set which processes should run and be ready to carry out the work that comes next time. The process scheduling algorithm proposed in this study is the FIFO*(First In First Out)*algorithm. In the FIFO process scheduling algorithm, a process will be done sequentially according to the time of arrival of the process. But if there is a process / work that comes simultaneously, a queue mechanism will be created. For example, if there are three P1, P2 and P3 jobs with the length of CPU work time (CPU *Burst-time)* each:

Process	***Burs-time***
P1	24
P2	3
P3	3

If the process comes in the order of P1, P2, P3 and isserved with a FIFO algorithm then it can be described by its *Gantt Chart* as follows:



So if wacttu wait for the process P1 is 0 milliseconds. As for the P2process, the waiting time is 24 milliseconds. And for the P3process, the waiting time is 27 milli seconds. This happens because in the FIFO algorithm, the P2 process is executed after the P1 process is completed for 24 milliseconds. The same is true of the P3 process.

The waiting time of the P3 process is the sum of the length of time of the previously running processes, namely P1 and P2. So it can be concluded that the average waiting time of the three processes is 17 milliseconds. In addition to waiting time, an important parameter commonly calculated in the FIFO algorithm is the work completion time (Turn*Around Time).* The completion time of the work is calculated using the formula:

$$\boldsymbol{TA = Waiting\,Time + Length\,Of\,Execution} \tag{1}$$

Where *Waiting time* is the process waiting time, and Length Of Execution is the length *of* process execution time. So it can be concluded, *Turn Around Time* for the P1 process is 24 milliseconds. Turn *Around Time* for P2 and P3 processes is 27 milliseconds and 30 milliseconds, respectively. The turn-around *time* average of the three processes is 27 milliseconds. In the case of distributed RSA encryption and decryption in the SBC cluster architecture, the upcoming encryption work will create a queue mechanism. Where later, encryption and decryption work will be carried out in the order in the queue. So there will be no continuous processing due to data damage at the time of processing.

2.1 DESIGN OF TESTING TESTBED

The methods proposed in this study will be tested in a real system environment. The details of the testbed environment can be described in Figure 1. Each SBC used in this testing environment is a Raspberry Pi 4 Model B that has the following specifications:

• *Processor*	: Broadcom BCM2711, Quad core Cortex-A72 SoC @ 1.5GHz
• *RAM*	: 4GB LPDDR4-3200 SDRAM
• *Storage*	: 16 GB
• *NIC*	: 10/100/1000 Mbps Ethernet
• Sistem Operasi	: RaspiOS

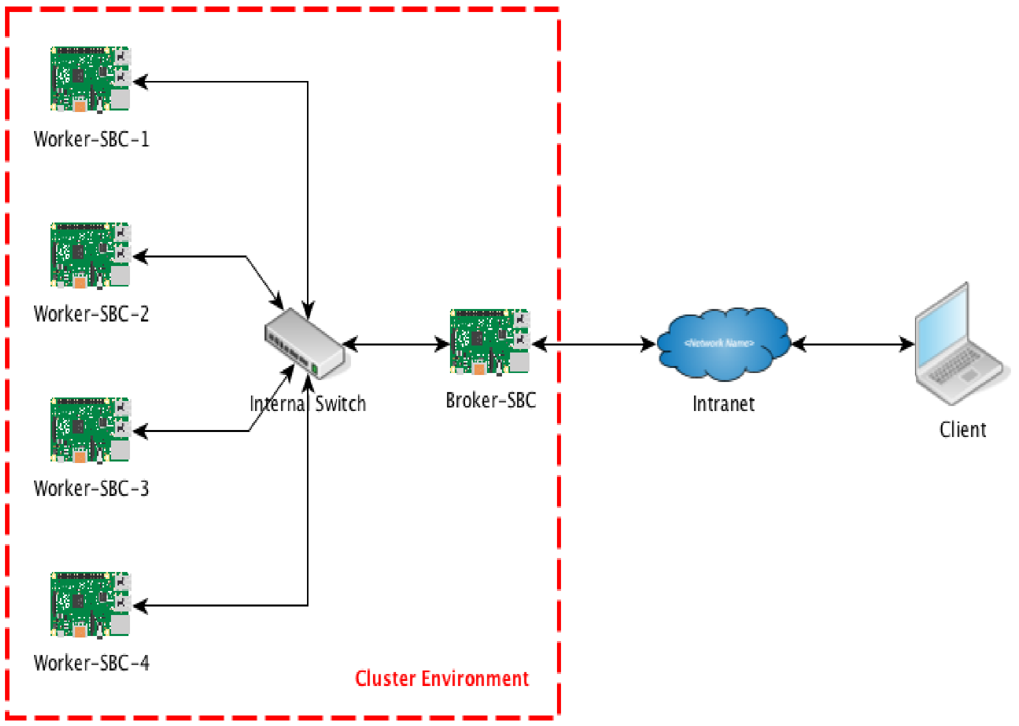


FIGURE 1. Desain of *Testbed* Architecture

Rsa encryption and decryption performance tests distributed in the SBC cluster architecture will be measured based on the speed of completion time of a series of jobs. The scenario will be tested on two different mechanisms. The first mechanism, a series of jobs will be executed in the absence of a queue mechanism. While in the second mechanism, the work will be executed by applying the queue system proposed in this study, namely queues using the FIFO algorithm. The series of jobs that will be tested on both mechanisms are the same. That is a series of encryption work of 5 documents with different sizes ranging from sizes 10MB, 30MB, 50MB, 80MB, and 100MB. The document file will be sent by the *client* computer. Each mechanism will be tested with the job series scenario shown in the table below:

TABEL 1.
Testing Scenario

No.	Skenario	Urutan Pekerjaan	No.	Skenario	Urutan Pekerjaan
1		10 MB	26		10 MB
2		30 MB	27		30 MB
3	Skenario 1	50 MB	28	Skenario 6	100 MB
4		80 MB	29		80 MB
5		100 MB	30		50 MB
6	Skenario 2	100 MB	31	Skenario 7	30 MB
7		80 MB	32		10 MB

8		50 MB	33		100 MB
9		30 MB	34		50 MB
10		10 MB	35		80 MB
11		10 MB	36		30 MB
12		50 MB	37		50 MB
13	Skenario 3	30 MB	38	Skenario 8	100 MB
14		80 MB	39		10 MB
15		100 MB	40		80 MB
16		100 MB	41		100 MB
17		80 MB	42		50 MB
18	Skenario 4	30 MB	43	Skenario 9	10 MB
19		50 MB	44		30 MB
20		10 MB	45		80 MB
21		10 MB	46		100 MB
22		80 MB	47		30 MB
23	Skenario 5	100 MB	48	Skenario 10	10 MB
24		30 MB	49		50 MB
25		50 MB	50		80 MB

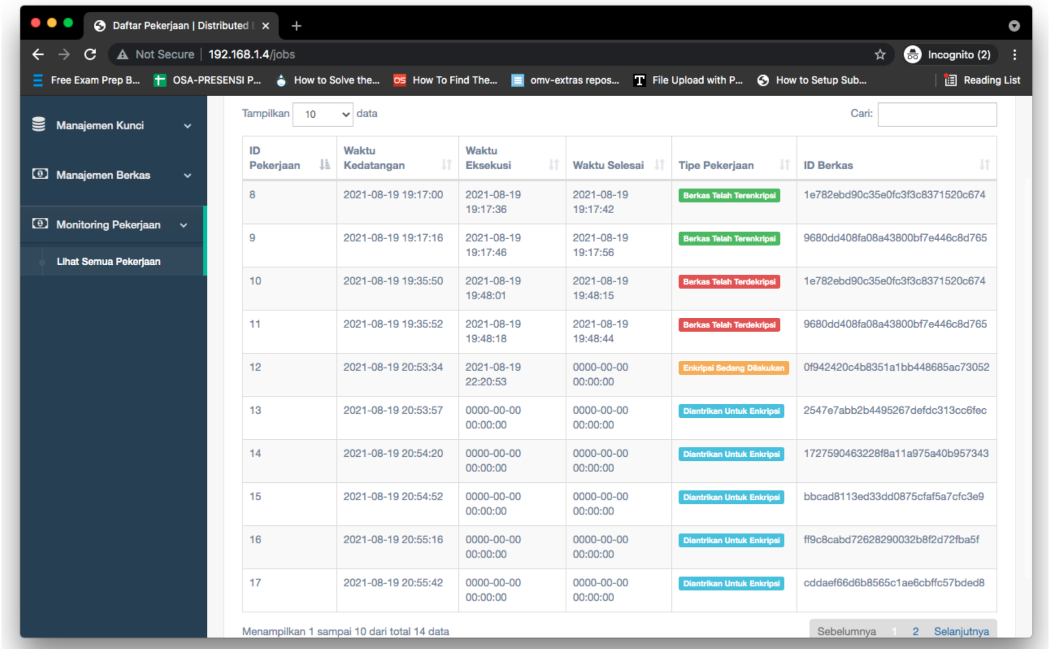
2. RESULT AND DISCUSSION

In this study, researchers developed an app on the user side. This application makes it easy and facilitates users to be able to simultaneously use file encryption services. The created application supports the use of multi-user separated by the user authentication system. On this page there is also the main menu of this application, namely the Key Management and File Management menu. The Key Management menu serves to provide user control over the keys that the user uses to encrypt and decrypt files stored into the system. The Key Management menu has submenu or facilities for creating keys, viewing existing keys, renaming keys and removing keys. In the Create New Key submenu, users can create a new key that can later be used to encrypt and decrypt files owned.

In the See All Keys submenu, users can see the keys they have and can perform some actions on those keys such as changing the name of the key identification and removing the key that is owned when the key is not used by the existing files. On the File Management menu, users can manage the files they have. Users can add new files to be encrypted by using the Create New Files submenu. For each new file to be added, the user must select the key he already has to be used in the encryption process that runs in real time on the system.

After the file is added by the user, it appears in the View All Files submenu view. Files that have just been added by the user will appear on the list of files with the status attributed to encryption. And when the file is in the process of encryption then its status will appear on the page view. After the enkrpsi process is complete, the user will be given the option to decrypt the file when the user wants to download the file from the system.

In this application, there is also a special page for system administrators, namely the Job Monitoring menu. On the page, the job information will be displayed in the system. Both the work that has been done, is being done or is still in the queue for processing. This page works in real time to display the work found on the system.



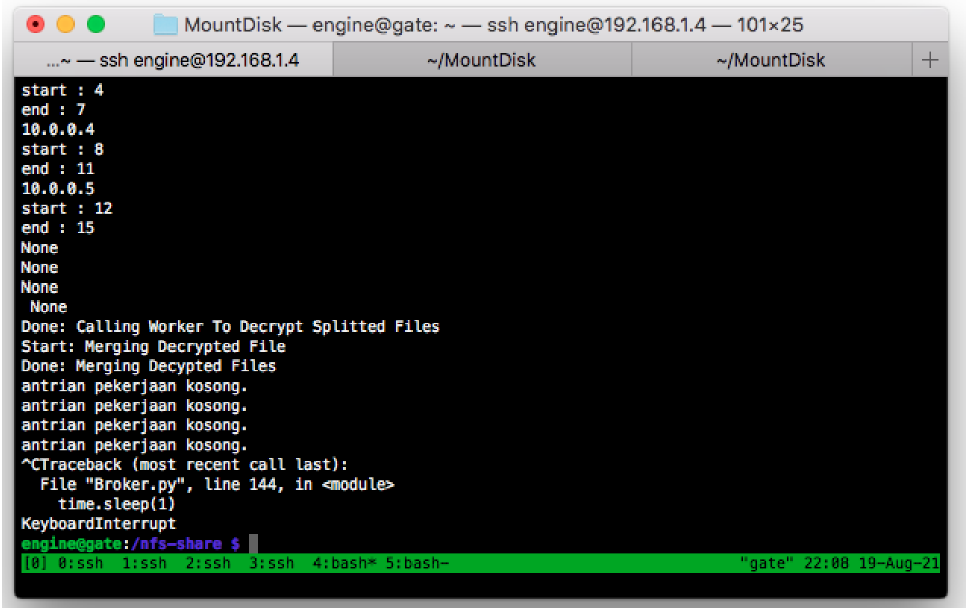
Tampilkan 10 data   Cari:

ID Pekerjaan	Waktu Kedatangan	Waktu Eksekusi	Waktu Selesai	Tipe Pekerjaan	ID Berkas
8	2021-08-19 19:17:00	2021-08-19 19:17:36	2021-08-19 19:17:42	Berkas Telah Terenkripsi	1e782ebd90c35e0fc3f3c8371520c674
9	2021-08-19 19:17:16	2021-08-19 19:17:46	2021-08-19 19:17:56	Berkas Telah Terenkripsi	9680dd408fa08a43800bf7e446c8d765
10	2021-08-19 19:35:50	2021-08-19 19:48:01	2021-08-19 19:48:15	Berkas Telah Terdekripsi	1e782ebd90c35e0fc3f3c8371520c674
11	2021-08-19 19:35:52	2021-08-19 19:48:18	2021-08-19 19:48:44	Berkas Telah Terdekripsi	9680dd408fa08a43800bf7e446c8d765
12	2021-08-19 20:53:34	2021-08-19 22:20:53	0000-00-00 00:00:00	Enkripsi Sedang Dilakukan	0f942420c4b8351a1bb448685ac73052
13	2021-08-19 20:53:57	0000-00-00 00:00:00	0000-00-00 00:00:00	Diantrikan Untuk Enkripsi	2547e7abb2b4495267defdc313cc6fec
14	2021-08-19 20:54:20	0000-00-00 00:00:00	0000-00-00 00:00:00	Diantrikan Untuk Enkripsi	1727590463228f8a11a975a40b957343
15	2021-08-19 20:54:52	0000-00-00 00:00:00	0000-00-00 00:00:00	Diantrikan Untuk Enkripsi	bbcad8113ed33dd0875cfaf5a7cfc3e9
16	2021-08-19 20:55:16	0000-00-00 00:00:00	0000-00-00 00:00:00	Diantrikan Untuk Enkripsi	ff9c8cabd72628290032b8f2d72fba5f
17	2021-08-19 20:55:42	0000-00-00 00:00:00	0000-00-00 00:00:00	Diantrikan Untuk Enkripsi	cddaef66d6b8565c1ae6cbffc57bded8

Menampilkan 1 sampai 10 dari total 14 data   Sebelumnya 1 2 Selanjutnya

FIGURE 2. Desain of *Testbed* Architecture

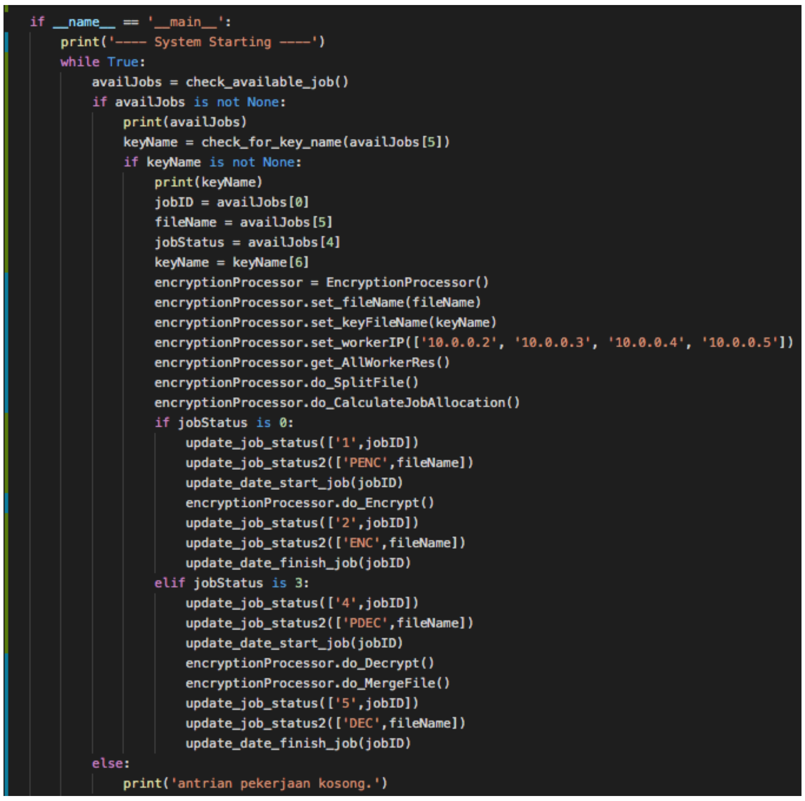
In addition to applications for users, in this study there are also modifications of distributed encryption-decryption applications that have been made in previous research. This modification is intended so that the application can implement the scheduling mechanism that is the main focus of this research. The application is modified and run as a service on SBC Broker and SBC Worker in order to do the work that comes at the same time.


MountDisk — engine@gate: ~ — ssh engine@192.168.1.4 — 101×25
...~ — ssh engine@192.168.1.4
~/MountDisk
~/MountDisk

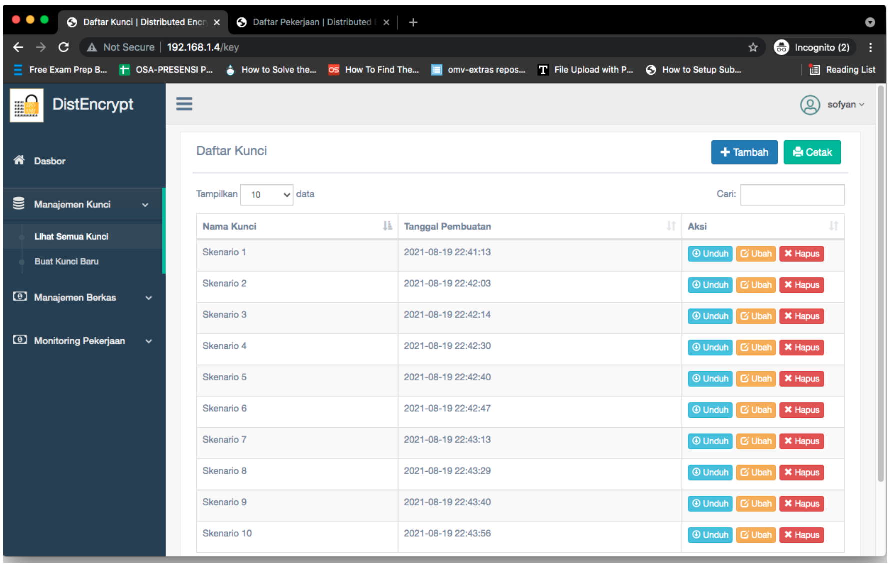
```
start : 4
end : 7
10.0.0.4
start : 8
end : 11
10.0.0.5
start : 12
end : 15
None
None
None
 None
Done: Calling Worker To Decrypt Splitted Files
Start: Merging Decrypted File
Done: Merging Decypted Files
antrian pekerjaan kosong.
antrian pekerjaan kosong.
antrian pekerjaan kosong.
antrian pekerjaan kosong.
^CTraceback (most recent call last):
  File "Broker.py", line 144, in <module>
    time.sleep(1)
KeyboardInterrupt
engine@gate:/nfs-share $
[0] 0:ssh  1:ssh  2:ssh  3:ssh  4:bash* 5:bash-                  "gate" 22:08 19-Aug-21
```

Application modification to implement process scheduling is done by adding a subsystem checking the queue of work recorded on the database continuously. If there is work, it will automatically be done. When there is no work in the queue database then the system will stop for 1 second. And then it will do a recheck on the job database. If there are two or more jobs in the queue database, the system will do its job based on the arrival time of the job. The basis of determining the working scheduling mechanism of this application is based on the FIFO scheduling theory. So that the application made has been qualified to be used as a test data retrieval tool in this study. The testing process is done by creating a key to encrypt and decrypt files in each scenario. The key is made with the same level of complexity and key size to ensure the similarity of the complexity level of the process.

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)



```
if __name__ == '__main__':
    print('---- System Starting ----')
    while True:
        availJobs = check_available_job()
        if availJobs is not None:
            print(availJobs)
            keyName = check_for_key_name(availJobs[5])
            if keyName is not None:
                print(keyName)
                jobID = availJobs[0]
                fileName = availJobs[5]
                jobStatus = availJobs[4]
                keyName = keyName[6]
                encryptionProcessor = EncryptionProcessor()
                encryptionProcessor.set_fileName(fileName)
                encryptionProcessor.set_keyFileName(keyName)
                encryptionProcessor.set_workerIP(['10.0.0.2', '10.0.0.3', '10.0.0.4', '10.0.0.5'])
                encryptionProcessor.get_AllWorkerRes()
                encryptionProcessor.do_SplitFile()
                encryptionProcessor.do_CalculateJobAllocation()
                if jobStatus is 0:
                    update_job_status(['1',jobID])
                    update_job_status2(['PENC',fileName])
                    update_date_start_job(jobID)
                    encryptionProcessor.do_Encrypt()
                    update_job_status(['2',jobID])
                    update_job_status2(['ENC',fileName])
                    update_date_finish_job(jobID)
                elif jobStatus is 3:
                    update_job_status(['4',jobID])
                    update_job_status2(['PDEC',fileName])
                    update_date_start_job(jobID)
                    encryptionProcessor.do_Decrypt()
                    encryptionProcessor.do_MergeFile()
                    update_job_status(['5',jobID])
                    update_job_status2(['DEC',fileName])
                    update_date_finish_job(jobID)
        else:
            print('antrian pekerjaan kosong.')
```



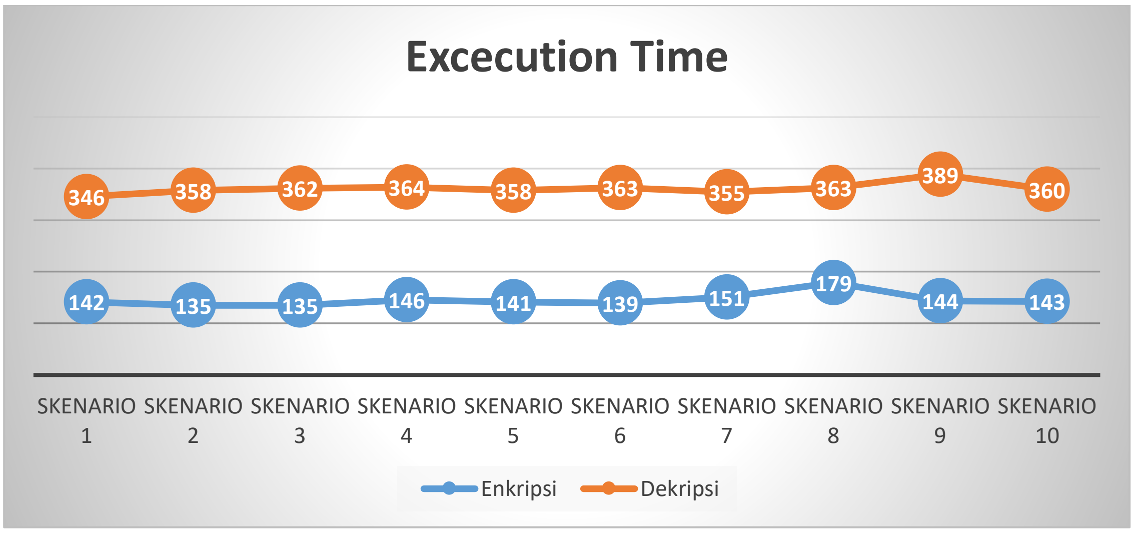
Daftar Kunci

Nama Kunci	Tanggal Pembuatan	Aksi
Skenario 1	2021-08-19 22:41:13	Unduh Ubah Hapus
Skenario 2	2021-08-19 22:42:03	Unduh Ubah Hapus
Skenario 3	2021-08-19 22:42:14	Unduh Ubah Hapus
Skenario 4	2021-08-19 22:42:30	Unduh Ubah Hapus
Skenario 5	2021-08-19 22:42:40	Unduh Ubah Hapus
Skenario 6	2021-08-19 22:42:47	Unduh Ubah Hapus
Skenario 7	2021-08-19 22:43:13	Unduh Ubah Hapus
Skenario 8	2021-08-19 22:43:29	Unduh Ubah Hapus
Skenario 9	2021-08-19 22:43:40	Unduh Ubah Hapus
Skenario 10	2021-08-19 22:43:56	Unduh Ubah Hapus

Proses encryption begins when uploading a file. In the uploading process, the key must be selected according to the scenario to be tested. After uploading the file, the encryption process begins and can be viewed on the work monitoring page. From the page, obtained data on the arrival time of work, the start time of work and the time of completion of work. All these parameters are calculated to produce the data to be analyzed. From the tests conducted, the results of the total processing time as stated in the table below.

Scenario	Encription	Description
Scenario 1	142	346
Scenario 2	135	358
Scenario 3	135	362
Scenario 4	146	364
Scenario 5	141	358
Scenario 6	139	363
Scenario 7	151	355
Scenario 8	179	363
Scenario 9	144	389
Scenario 10	143	360

Based on the data from the test results, it is seen that the encryption and decryption process has a small time difference of 12.7 seconds for the encryption process and 10.9 seconds for the decryption process. The results of the calculation are based on the standard deviation from the data of each existing scenario. Then, the test results were compared with the test results obtained in the previous study. In previous research it was known that the encryption and decryption processes were the same for the total processing time in each scenario.



Scenario	Encription	Description
Scenario 1	175	1767
Scenario 2	175	1767
Scenario 3	175	1767
Scenario 4	175	1767
Scenario 5	175	1767
Scenario 6	175	1767

Scenario 7	175	1767
Scenario 8	175	1767
Scenario 9	175	1767
Scenario 10	175	1767

3. CONCLUSION

The test result showed when compared to current research, the addition of process scheduling mechanisms can speed up processing time. This is evidenced by the average encryption process time in the previous research is longer than 30 seconds when compared to the current study. As for the decryption process, the processing time is much faster which is 4 times. This happens because in previous studies there was a buildup of decryption work done at the same time. Decryption processes that require larger resources are greatly affected by simultaneous processing. It can therefore be concluded that the addition of process scheduling mechanisms can improve the efficiency of processing time.

ACKNOWLEDGEMENTS

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] Meneses, F., Fuertes, W., Salvador, S., Flores, D., Aules, H., Castro, F., Torres, J., Miranda, A., Nuela, D., 2016. RSA Encryption Algorithm Optimization to Improve Performance and Security Level of Network Messages 8.

[2] Andrews, G.R., 1999. Foundations of Multithreaded, Parallel, and Distributed Programming, 1st ed. Addison-Wesley

[3] Lee, S., Lee, E., 2009a. Distributed Adaptation System for Quality Assurance of Web Service in Mobile Environment. JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 25, 403–417

[4] Arief, S.N., Firdaus, V.A.H., Prasetyo, A., 2020. Optimization of RSA encryption and decryption process with distributed computing method. IOP Conference Series: Materials Science and Engineering 830. https://doi.org/10.1088/1757-899X/830/2/022090

[5] Sukoco, H., Solahudin, M., Iqbal, M., 2015. Perbandingan Kinerja Pemrosesan Paralel Pada PC dan Raspberry Pi Untuk Pendeteksian Gulma Pada Lahan Pertanian Menggunakan Fraktal

[6] Prasetyo, A., Arief, S.N., Wakhidah, Rokhimatul, 2021. OPTIMASI PEMROSESAN ENKRIPSI DAN DEKRIPSI RSA PADA SINGLE BOARD COMPUTER (SBC) DENGAN PEMBAGIAN BEBAN KOMPUTASI DALAM SISTEM TERDISTRIBUSI. JIP (Jurnal Informatika Polinema).

[7] Irvi, E., 2019. Rancang Bangun dan Evaluasi Kinerja Raspberry Pi Cluster sebagai Platform Penerapan Pembelajaran Mesin.

[8] Tanenbaum, A.S., Woodhull, A.S., 2011. Operating Systems Design and Implementation, 3rd ed. Pearson Education

[9] Basford, P. J., Johnston, S. J., Perkins, C. S., Garnock-Jones, T., Tso, F. P., Pezaros, D., … Cox, S. J. (2020). Performance analysis of single board computer clusters. *Future Generation Computer Systems*, *102*. https://doi.org/10.1016/j.future.2019.07.040

[10] Ching, P. L., Mutuc, J. E., & Jose, J. A. (2019). Assessment of the quality and sustainability implications of FIFO and LIFO inventory policies through system dynamics. *Advances in Science, Technology and Engineering Systems*, *4*(5). https://doi.org/10.25046/aj040509

[11] Zhong, X., & Liang, Y. (2016). Raspberry Pi: An effective vehicle in teaching the internet of things in computer science and engineering. *Electronics (Switzerland)*. https://doi.org/10.3390/electronics5030056

[12] Hazay, C., Mikkelsen, G. L., Rabin, T., Toft, T., & Nicolosi, A. A. (2019). Efficient RSA Key Generation and Threshold Paillier in the Two-Party Setting. *Journal of Cryptology, 32*(2). https://doi.org/10.1007/s00145-017-9275-7