JURIKOM (Jurnal Riset Komputer), Vol. 9 No. 1, Februari 2022
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v9i1.3865
Hal 85−92
http://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

Optimalisasi Pemrosesan Distributed RSA Algorithm dengan Metode Penjadwalan Proses Pada Single Board Computer Cluster

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

[1]Jurusan Teknologi Informasi, Politeknik Negeri Malang
Jl. Soekarno Hatta No.9, Jatimulyo,Lowokwaru, Malang, Jawa Timur, Indonesia
Email: [1]sofyan@polinema.ac.id, [2*]vipkas@polinema.ac.id, [3]arief.prasetyo@polinema.ac.id
(* : vipkas@polinema.ac.id)
Submitted **16-02-2022**; Accepted **25-02-2022**; Published **25-02-2022**

Abstrak

Keamanan data masih menjadi masalah utama mengenai perlunya kerahasiaan data. Proses enkripsi menggunakan algoritma RSA masih merupakan metode yang paling populer digunakan dalam mengamankan data karena kompleksitas persamaan matematika yang digunakan dalam algoritma ini membuat sulit untuk hack. Namun, kompleksitas algoritma RSA masih menjadi masalah utama yang menghambat penerapannya dalam aplikasi yang lebih kompleks. Optimasi diperlukan dalam pemrosesan algoritma RSA ini, salah satunya dengan menjalankannya pada sistem terdistribusi. Dalam makalah ini, kami mengusulkan pendekatan dengan algoritma penjadwalan proses FIFO yang berjalan pada cluster komputer papan tunggal. Hasil pengujian menunjukkan bahwa alokasi sumber daya dalam sistem yang menggunakan algoritma penjadwalan proses FIFO lebih efisien dan menunjukkan penurunan waktu pemrosesan enkripsi RSA secara keseluruhan.

Kata Kunci: Distributed Computation, Encryption; RSA; FIFO

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: Distributed Computation; Encryption; RSA; FIFO

1. PENDAHULUAN

Keamanan data masih menjadi masalah populer untuk penelitian. Hal ini tidak terlepas dari kebutuhan akan kerahasiaan data. Dengan kemajuan teknologi seperti sekarang ini, data (berisi informasi penting) dapat dicuri oleh pihak yang tidak berwenang jika tidak diamankan. Salah satu cara untuk mengamankan data adalah dengan melakukan proses enkripsi. Algoritma RSA masih merupakan metode yang paling populer digunakan. Kesulitan dalam menetas algoritma ini adalah alasan utama untuk penggunaan algoritma ini dalam berbagai kasus penggunaan. Persamaan matematika yang digunakan dalam algoritma ini membuatnya sulit untuk diretas. Namun, persamaan matematika yang lebih kompleks yang digunakan, semakin besar sumber daya komputasi yang dibutuhkan untuk memprosesnya . Algoritma RSA juga memiliki karakteristik di mana ia memiliki keamanan yang baik bahkan mendekati sempurna, manajemen kunci mudah, mudah diterapkan dan mudah dimengerti. Namun dalam algoritma ini, daya modular memiliki peran besar dalam mempengaruhi kinerja algoritma dan menjadi masalah utama yang menghambat penerapannya ke aplikasi yang lebih kompleks. Oleh karena itu, pemrosesan algoritma enkripsi RSA yang cepat (termasuk) telah menjadi fokus dalam berbagai penelitian. [1][2][3][4]

Optimalisasi dalam pemrosesan algoritma RSA yang cepat dapat dilakukan dengan berbagai cara. Salah satunya dengan menjalankannya pada sistem terdistribusi. Dalam sistem terdistribusi, pekerjaan dibagi dan dieksekusi pada satu set sumber daya komputasi. Satu set sumber daya komputasi akan berkomunikasi satu sama lain dan berkolaborasi dalam memproses pekerjaan. Dalam penelitian sebelumnya, satu set komputer dalam bentuk komputer virtual digunakan untuk memproses enkripsi menggunakan algoritma RSA. Dalam studi tersebut, pekerjaan enkripsi didistribusikan ke komputer masing-masing pekerja. Pembagian pekerjaan enkripsi didistribusikan secara merata kepada setiap pekerja. Hasil penelitian membuktikan bahwa pemrosesan terdistribusi dapat meningkatkan kecepatan pemrosesan enkripsi RSA. Tetapi masalah baru muncul ketika penggunaan komputer virtual diganti dengan satu set komputer fisik. Penggunaan komputer fisik dalam jumlah besar sangat tidak efisien baik dari segi biaya pembelian alat maupun dari segi biaya operasional. Oleh karena itu, dibutuhkan sumber daya komputasi yang lebih murah dan biaya operasional yang rendah.[5][6][7][8]

Pada saat ini, kemajuan teknologi membuat pengembangan alat pemrosesan komputasi menjadi semakin berkembang. *Single Board Computer* (SBC) juga mulai digunakan untuk menggantikan alat pemrosesan komputasi klasik yang umumnya merupakan komputer pribadi dalam bentuk laptop dan workstation. Dalam sebuah penelitian yang dilakukan oleh, implementasi satu SBC Raspberry Pi 3 Model B + dalam kasus deteksi gulma di lahan pertanian mampu mengungguli kinerja komputer pribadi (intel core i5 berbasis x86) sebesar 0,04 kali lebih cepat. Selain itu, penelitian yang dilakukan oleh juga membuktikan peningkatan yang signifikan dalam implementasi arsitektur cluster SBC pada pemrosesan *machine learning* [9][10][11] bila dibandingkan dengan pemrosesan pada satu SBC . Oleh karena itu, muncul hipotesis bahwa cluster SBC dapat menjadi solusi pemrosesan enkripsi RSA yang murah dan ekonomis dalam hal biaya pembelian perangkat serta

JURIKOM (Jurnal Riset Komputer), Vol. 9 No. 1, Februari 2022
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v9i1.3865
Hal 85−92
http://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

operasional. Hipotesis ini terbukti benar dalam penelitian yang dilakukan oleh . Penggunaan cluster SBC dalam proses enkripsi RSA dapat mengungguli satu komputer pribadi *multicore* [12][8] dengan biaya pembelian perangkat yang sama. Biaya operasi cluster SBC juga jauh lebih efisien jika dibandingkan dengan satu komputer pribadi *multicore* [13][14][15].

Masalah baru muncul ketika tugas melakukan pekerjaan enkripsi datang secara bersamaan. Proses kerja yang tidak terkontrol dapat menyebabkan kerusakan enkripsi data yang mengarah pada pemrosesan tanpa akhir (pemrosesan berkelanjutan) pada komputer pekerja (baik komputer pribadi tunggal atau SBC). Pemrosesan berkelanjutan dan tidak pernah berakhir dapat menyebabkan penurunan total dalam kinerja pemrosesan enkripsi. Selain itu, dapat menyebabkan kerusakan pada perangkat komputer yang digunakan untuk memproses enkripsi [16][17][18].

Penjadwalan proses adalah pendekatan yang dapat digunakan untuk memecahkan masalah ini. Dengan penjadwalan proses, proses akan dijalankan secara terstruktur dan terkontrol. Sehingga akan meminimalisir adanya pengolahan yang tidak berujung. Ada banyak algoritma penjadwalan proses yang saat ini ada, seperti FIFO *(First In First Out),* LILO (Last *in Last Out),* LIFO (Last *in First Out)* [19]. Algoritma penjadwalan proses itu sendiri dilakukan dengan membentuk antrian (atau biasa disebut *queue).* Dalam algoritma FIFO, *queue* proses dibuat dengan meletakkan pekerjaan / proses berdasarkan waktu kedatangan. Sehingga pekerjaan/proses yang diterima pada saat (t)-1 akan diproses terlebih dahulu dibandingkan dengan pekerjaan/proses yang diterima pada saat (t). Pendekatan yang hampir sama diterapkan pada algoritma LILO. Antrian akan dibuat dari belakang pada algoritma LILO. Dengan demikian, pekerjaan/proses yang diterima pada saat (t) akan diproses setelah pekerjaan/proses diterima pada saat (t)-1. Algoritma LIFO menggunakan pendekatan yang berbeda. Antrian akan dibuat berdasarkan proses / pekerjaan yang datang terlambat. Dalam algoritma ini, kerja / proses yang diterima pada saat (t) yang kedua akan tertunda pemrosesannya ketika pekerjaan / proses baru diterima pada waktu (t) + 1. Dan itu akan dilanjutkan ketika pekerjaan atau proses yang diterima pada saat (t) + 1 selesai [20][21][22][23].

Hipotesis dikemukakan oleh para peneliti untuk menjawab masalah pemrosesan enkripsi yang datang hampir bersamaan, yaitu dengan menerapkan penjadwalan proses. Dalam kasus enkripsi RSA pada arsitektur cluster SBC, algoritma penjadwalan yang paling mungkin adalah algoritma FIFO *(First In First Out*). Pemilihan algoritma ini didasarkan pada waktu tunggu yang dihasilkan dari pemrosesan FIFO yang lebih cepat dibandingkan dengan LIFO. Dalam algoritma LIFO, kerja / proses enkripsi yang sedang berlangsung akan ditangguhkan sementara dan akan dilanjutkan setelah pekerjaan / proses terakhir yang diterima telah diproses. Sehingga waktu tunggu untuk pekerjaan/proses yang datang pada saat (t)-1 akan lebih lama dari pekerjaan/proses yang datang pada saat itu (t). Ini dapat meningkatkan waktu pemrosesan enkripsi RSA secara keseluruhan. Sementara dalam algoritma FIFO, proses yang sedang berjalan (pekerjaan yang datang tepat waktu (t)-1) akan tetap berjalan tanpa terganggu oleh pekerjaan yang datang pada saat itu (t). Pekerjaan yang datang pada saat (t) akan dimasukkan dalam antrian dan diproses setelah pekerjaan yang datang pada (t) -1 selesai. Jadi, waktu tunggu yang diperlukan untuk proses masuk (t)-1 akan lebih cepat jika dibandingkan dengan algoritma LILO. Dan dampaknya adalah penurunan waktu pemrosesan enkripsi RSA secara keseluruhan. Oleh karena itu, dalam penelitian ini algoritma FIFO akan diterapkan, diuji, dan dianalisis untuk menguji hipotesis yang telah diajukan oleh penulis [24][25].

2. METODOLOGI PENELITIAN

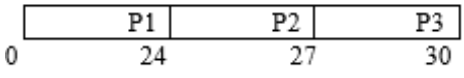
2.1 Metode

Sebuah ide diajukan untuk mengatasi masalah pemrosesan enkripsi RSA yang belum maksimal dan dekripsi dalam sistem terdistribusi yang diterapkan pada klaster SBC. Ide yang diusulkan dalam penelitian ini adalah untuk menambahkan mekanisme penjadwalan proses, sehingga dapat mengatur proses mana yang harus berjalan dan siap untuk melaksanakan pekerjaan yang datang lain kali. Algoritma penjadwalan proses yang diusulkan dalam penelitian ini adalah algoritma FIFO *(First In First Out*). Dalam algoritma penjadwalan proses FIFO, proses akan dilakukan secara berurutan sesuai dengan waktu kedatangan proses. Namun jika ada proses/pekerjaan yang datang bersamaan, akan dibuat mekanisme antrian. Misalnya, jika ada tiga pekerjaan P1, P2 dan P3 dengan lamanya waktu kerja CPU (CPU *Burst-time)* masing-masing sebagaimana pada Tabel 1 berikut :

Tabel 1. Proses waktu kerja CPU

Process	Burs-time
P1	24
P2	3
P3	3

Jika prosesnya berada dalam urutan P1, P2, P3 dan disajikan dengan algoritma FIFO maka dapat dijelaskan oleh *Bagan Gantt* sebagai berikut:



Gambat 1. Bagan Gantt

JURIKOM (Jurnal Riset Komputer), Vol. 9 No. 1, Februari 2022
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v9i1.3865
Hal 85−92
http://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

Jadi, jika waktu menunggu proses P1 adalah 0 milidetik. Sedangkan untuk proses P2, waktu tunggu adalah 24 milidetik. Dan untuk proses P3, waktu tunggu adalah 27 mili detik. Hal ini terjadi karena dalam algoritma FIFO, proses P2 dijalankan setelah proses P1 selesai selama 24 milidetik. Hal yang sama berlaku untuk proses P3. Waktu tunggu proses P3 adalah jumlah lamanya waktu proses yang sebelumnya berjalan, yaitu P1 dan P2. Jadi, dapat disimpulkan bahwa waktu tunggu rata-rata dari tiga proses adalah 17 milidetik. Selain waktu tunggu, parameter penting yang biasa dihitung dalam algoritma FIFO adalah waktu penyelesaian kerja (*Turn Around Time).* Waktu penyelesaian pekerjaan dihitung menggunakan rumus:

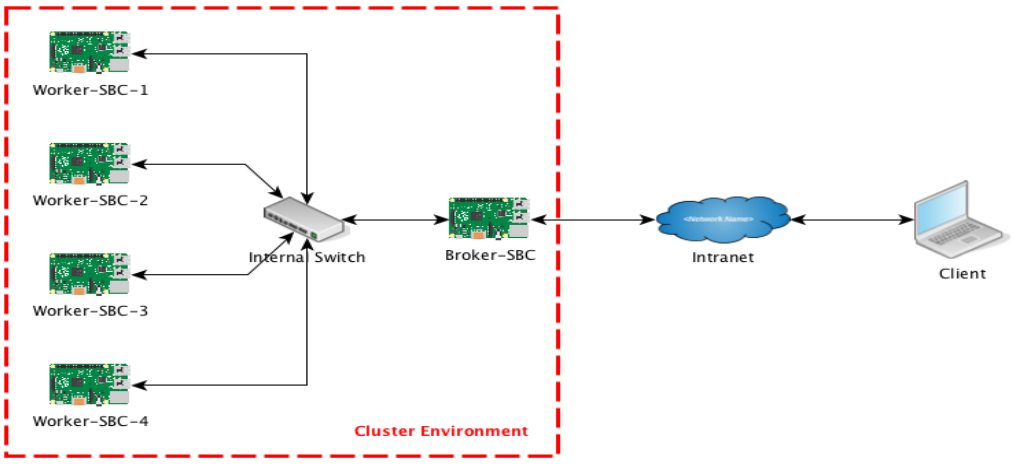
$$TA = Waiting\ Time + Length\ Of\ Execution. \quad (1)$$

Dimana *Waktu tunggu* adalah waktu tunggu proses, dan Lama eksekusi adalah lamanya waktu eksekusi proses. Jadi, dapat disimpulkan, Turn *Around Time* untuk proses P1 adalah 24 milidetik. Turn Around *Time* untuk proses P2 dan P3 masing-masing adalah 27 milidetik dan 30 milidetik. Rata-rata waktu turn-around dari tiga proses adalah 27 milidetik. Dalam kasus enkripsi dan dekripsi RSA terdistribusi dalam arsitektur cluster SBC, pekerjaan enkripsi yang akan datang akan membuat mekanisme antrian. Dimana nantinya, pekerjaan enkripsi dan dekripsi akan dilakukan dalam urutan dalam antrian. Jadi, tidak akan ada pemrosesan berkelanjutan karena kerusakan data pada saat pemrosesan.

2.2 Desain Pengujian Testbed

Metode yang diusulkan dalam penelitian ini akan diuji dalam lingkungan sistem nyata. Rincian lingkungan yang diuji dapat dijelaskan pada Gambar III.2. Setiap SBC yang digunakan dalam lingkungan pengujian ini adalah Raspberry Pi 4 Model B yang memiliki spesifikasi sebagai berikut:

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



Gambar 2. Desain Arsitektur

Tes kinerja enkripsi dan dekripsi RSA yang didistribusikan dalam arsitektur cluster SBC akan diukur berdasarkan kecepatan waktu penyelesaian serangkaian pekerjaan. Skenario akan diuji pada dua mekanisme yang berbeda. Mekanisme pertama, serangkaian pekerjaan akan dieksekusi tanpa adanya mekanisme antrian. Sedangkan pada mekanisme kedua, pekerjaan akan dijalankan dengan menerapkan sistem antrian yang diusulkan dalam penelitian ini, yaitu antrian menggunakan algoritma FIFO. Rangkaian pekerjaan yang akan diuji pada kedua mekanisme tersebut sama. Itu adalah serangkaian pekerjaan enkripsi 5 dokumen dengan ukuran berbeda mulai dari ukuran 10MB, 30MB, 50MB, 80MB, dan 100MB. File dokumen akan dikirim oleh *komputer klien*. Setiap mekanisme akan diuji dengan skenario seri pekerjaan yang ditunjukkan pada tabel di bawah ini:

Tabel 2. Skenario Pengujian

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

JURIKOM (Jurnal Riset Komputer), Vol. 9 No. 1, Februari 2022
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v9i1.3865
Hal 85−92
http://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

No.	Skenario	Urutan pekerjaan	No.	Skenario	Urutan Pekerjaan
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	Scenario 3	30 MB	38	Scenario 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	Scenario 4	30 MB	43	Scenario 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	Scenario 5	100 MB	48	Scenario 10	10 MB
24		30 MB	49		50 MB
25		50 MB	50		80 MB

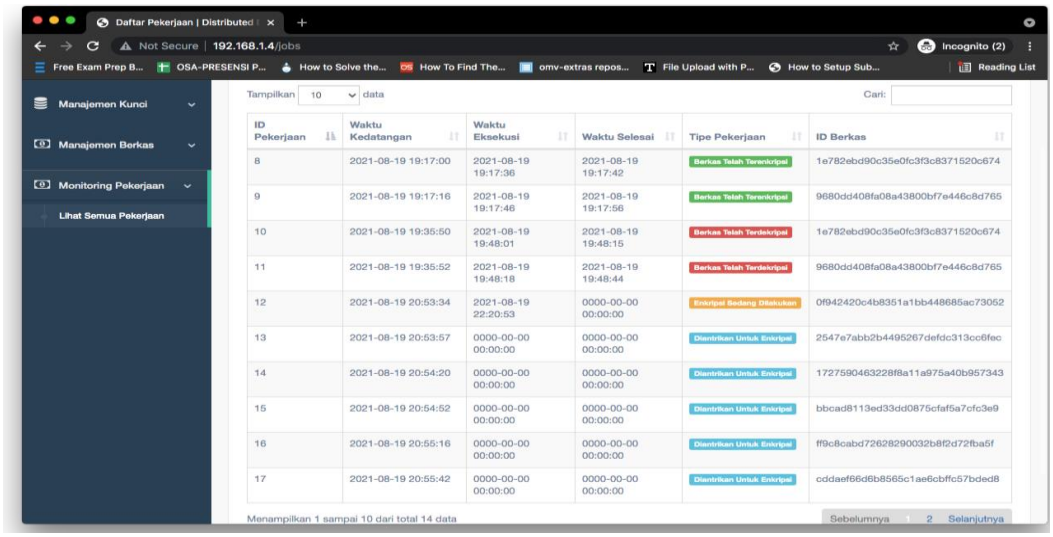
3. HASIL DAN PEMBAHASAN

Dalam studi ini, para peneliti mengembangkan aplikasi di sisi pengguna. Aplikasi ini memudahkan dan memudahkan pengguna untuk dapat sekaligus menggunakan layanan enkripsi file. Aplikasi yang dibuat mendukung penggunaan multi-user yang dipisahkan oleh sistem otentikasi pengguna. Pada halaman ini juga terdapat menu utama aplikasi ini, yaitu menu Key Management dan File Management. Menu Manajemen Kunci berfungsi untuk memberikan kontrol pengguna atas kunci yang digunakan pengguna untuk mengenkripsi dan mendekripsi file yang disimpan ke dalam sistem. Menu Manajemen Kunci memiliki submenu atau fasilitas untuk membuat kunci, melihat kunci yang ada, mengganti nama kunci dan menghapus kunci. Dalam *submenu Create New Key*, pengguna dapat membuat kunci baru yang nantinya dapat digunakan untuk mengenkripsi dan mendekripsi file yang dimiliki.

Di submenu Lihat Semua Kunci, pengguna dapat melihat kunci yang mereka miliki dan dapat melakukan beberapa tindakan pada kunci tersebut seperti mengubah nama identifikasi kunci dan menghapus kunci yang dimiliki ketika kunci tidak digunakan oleh file yang ada. Pada menu Manajemen File, pengguna dapat mengelola file yang mereka miliki. Pengguna dapat menambahkan file baru untuk dienkripsi dengan menggunakan *submenu Create New Files*. Agar setiap file baru ditambahkan, pengguna harus memilih kunci yang sudah harus digunakan dalam proses enkripsi yang berjalan secara real time pada sistem.

Setelah file ditambahkan oleh pengguna, file akan muncul di tampilan *submenu View All Files*. File yang baru saja ditambahkan oleh pengguna akan muncul di daftar file dengan status yang dikaitkan dengan enkripsi. Dan ketika file sedang dalam proses enkripsi maka statusnya akan muncul pada tampilan halaman. Setelah proses enkripsi selesai, pengguna akan diberi opsi untuk mendekripsi file ketika pengguna ingin mengunduh file dari sistem.

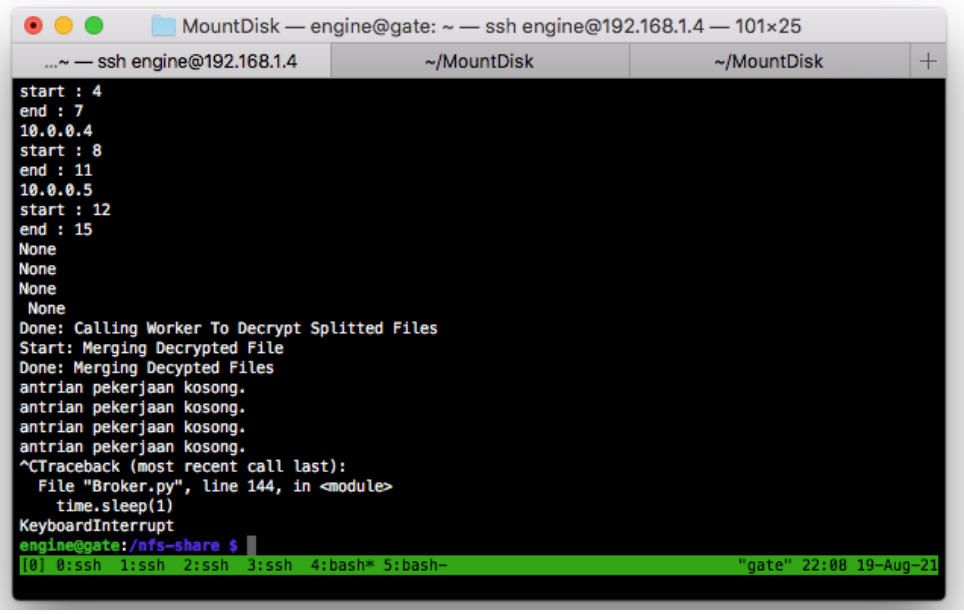
Dalam aplikasi ini, ada juga halaman khusus untuk administrator sistem, yaitu menu *Job Monitoring*. Pada halaman, informasi pekerjaan akan ditampilkan dalam sistem. Baik pekerjaan yang telah dilakukan, sedang dilakukan atau masih dalam antrian untuk diproses. Halaman ini berfungsi secara real time untuk menampilkan karya yang ditemukan pada sistem.



JURIKOM (Jurnal Riset Komputer), Vol. 9 No. 1, Februari 2022
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v9i1.3865
Hal 85−92
http://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

Gambar 3. Arsitektur Desain Testbed

Selain aplikasi untuk pengguna, dalam penelitian ini juga terdapat modifikasi aplikasi enkripsi-dekripsi terdistribusi yang telah dibuat dalam penelitian sebelumnya seperti yang ditunjukkan pada Gambar 3. Modifikasi ini dimaksudkan agar aplikasi dapat menerapkan mekanisme penjadwalan yang menjadi fokus penelitian ini. Aplikasi ini dimodifikasi dan dijalankan sebagai layanan di SBC Broker dan SBC Worker untuk melakukan pekerjaan yang datang pada saat yang sama.



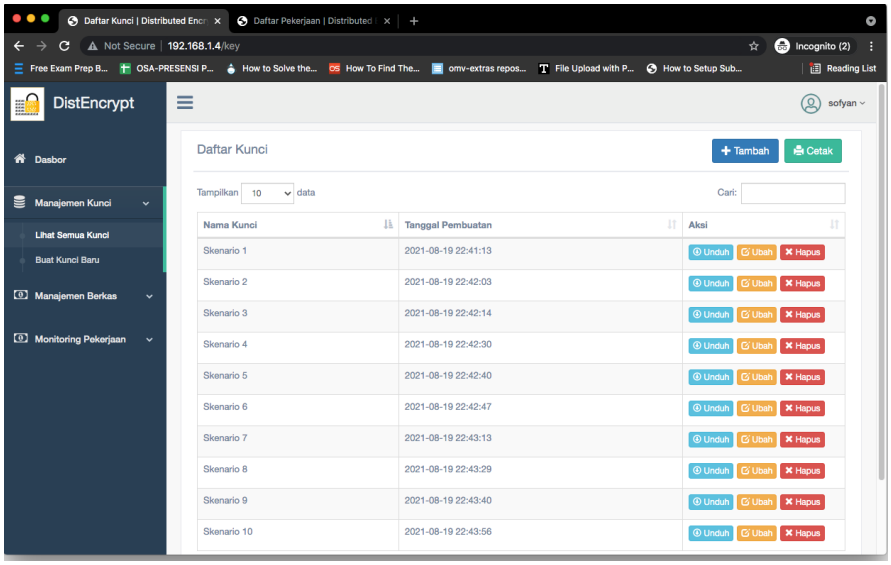
Gambar 4. Proses Dekripsi Enkripsi Terdistribusi

Modifikasi aplikasi untuk mengimplementasikan penjadwalan proses dilakukan dengan menambahkan subsistem yang memeriksa antrian pekerjaan yang direkam pada database secara terus menerus. Jika ada pekerjaan, itu akan secara otomatis dilakukan. Ketika tidak ada pekerjaan dalam database antrian maka sistem akan berhenti selama 1 detik. Dan kemudian akan melakukan pemeriksaan ulang pada database pekerjaan. Jika ada dua atau lebih pekerjaan dalam database antrian, sistem akan melakukan tugasnya berdasarkan waktu kedatangan pekerjaan. Dasar menentukan mekanisme penjadwalan kerja aplikasi ini didasarkan pada teori penjadwalan FIFO. Sehingga aplikasi yang dibuat telah mumpuni untuk dijadikan alat pengambilan data uji dalam penelitian ini. Proses pengujian dilakukan dengan membuat kunci untuk mengenkripsi dan mendekripsi file dalam setiap skenario seperti yang ditunjukkan pada Gambar 4. Kunci dibuat dengan tingkat kompleksitas dan ukuran kunci yang sama untuk memastikan kesamaan tingkat kompleksitas proses.

```
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.')
```

JURIKOM (Jurnal Riset Komputer), Vol. 9 No. 1, Februari 2022
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v9i1.3865
Hal 85−92
http://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

Gambar 5. Kode Proses Pengujian



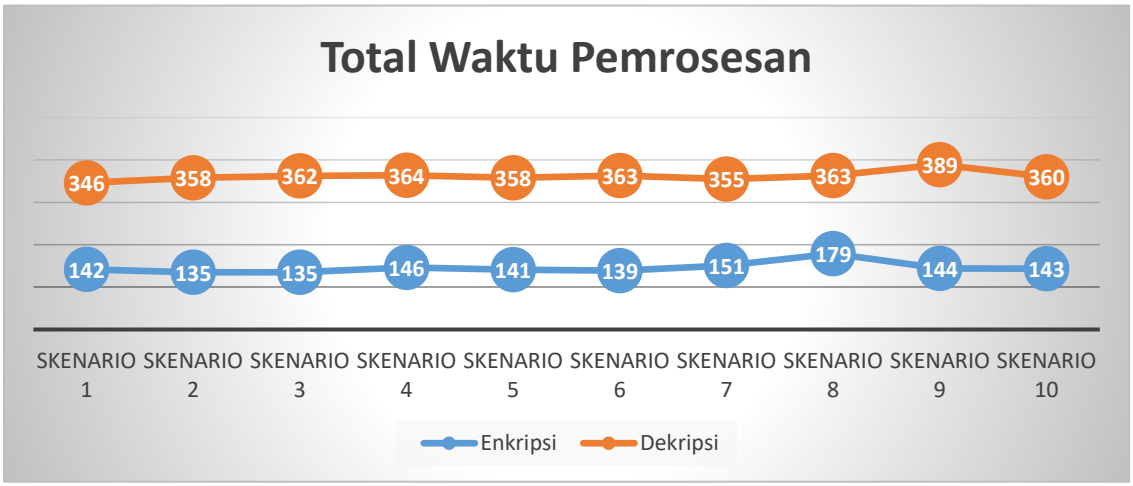
Gambar 6. Dasbor Pemantauan

Proses enkripsi dimulai saat mengunggah file. Dalam proses uploading, kunci harus dipilih sesuai dengan skenario yang akan diuji. Setelah mengunggah file, proses enkripsi dimulai dan dapat dilihat pada halaman pemantauan kerja di dasbor seperti yang ditunjukkan pada Gambar 5. Dari laman tersebut, diperoleh data waktu kedatangan kerja, waktu mulai kerja dan waktu penyelesaian pekerjaan. Semua parameter ini dihitung untuk menghasilkan data yang akan dianalisis. Dari tes yang dilakukan, hasil total waktu pemrosesan sebagaimana tercantum dalam Tabel 3 di bawah.

Tabel 3. Hasil Pengujian

Scenarios	Encryption	Decryption
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

Berdasarkan data dari hasil tes, terlihat bahwa proses enkripsi dan dekripsi memiliki perbedaan kecil-waktu 12,7 detik untuk proses enkripsi dan 10,9 detik untuk proses dekripsi. Hasil perhitungan didasarkan pada standar deviasi dari data setiap skenario yang ada. Kemudian, hasil tes dibandingkan dengan hasil tes yang diperoleh dalam penelitian sebelumnya. Dalam penelitian sebelumnya diketahui bahwa proses enkripsi dan dekripsi sama untuk total waktu pemrosesan di setiap skenario seperti yang ditunjukkan pada Gambar 6 dan hasilnya ditunjukkan pada Tabel 4.



JURIKOM (Jurnal Riset Komputer), Vol. 9 No. 1, Februari 2022
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v9i1.3865
Hal 85−92
http://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

Gambar 6. Hasil Perbandingan Proses

Tabel 4. Hasil Uji Perbandingan

Scenarios	Encryption	Decryption
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

4. KESIMPULAN

Hasil tes menunjukkan jika dibandingkan dengan penelitian saat ini, penambahan mekanisme penjadwalan proses dapat mempercepat waktu pemrosesan. Hal ini dibuktikan dengan rata-rata waktu proses enkripsi dalam penelitian sebelumnya yang lebih lama dari 30 detik jika dibandingkan dengan penelitian saat ini. Sedangkan untuk proses dekripsi, waktu pemrosesan jauh lebih cepat yaitu 4 kali. Hal ini terjadi karena dalam penelitian sebelumnya terjadi penumpukan pekerjaan dekripsi yang dilakukan pada saat yang bersamaan. Proses dekripsi yang membutuhkan sumber daya yang lebih besar sangat dipengaruhi oleh pemrosesan simultan. Oleh karena itu dapat disimpulkan bahwa penambahan mekanisme penjadwalan proses dapat meningkatkan efisiensi waktu pemrosesan.

REFERENCES

[1] R. Bhanot and R. Hans, "A review and comparative analysis of various encryption algorithms," International Journal of Security and its Applications, 2015, doi: 10.14257/ijsia.2015.9.4.27.
[2] G. Singh and S. Supriya, "A Study of Encryption Algorithms (RSA, DES, 3DES and AES) for Information Security," International Journal of Computer Applications, 2013, doi: 10.5120/11507-7224.
[3] N. Khanezaei and Z. M. Hanapi, "A framework based on RSA and AES encryption algorithms for cloud computing services," 2014. doi: 10.1109/SPC.2014.7086230.
[4] F. Meneses et al., "RSA Encryption Algorithm Optimization to Improve Performance and Security Level of Network Messages," 2016.
[5] P. N. Hemanth, N. Abhinay Raj, and N. Yadav, "Secure message transfer using RSA algorithm and improved playfair cipher in cloud computing," in 2017 2nd International Conference for Convergence in Technology, I2CT 2017, 2017, vol. 2017-January. doi: 10.1109/I2CT.2017.8226265.
[6] S. Lee and E. Lee, "Distributed adaptation system for quality assurance of web service in mobile environment," in Journal of Information Science and Engineering, Mar. 2009, vol. 25, no. 2, pp. 403–417.
[7] S. N. Arief, V. A. H. Firdaus, and A. Prasetyo, "Optimization of RSA encryption and decryption process with distributed computing method," in IOP Conference Series: Materials Science and Engineering, 2020, vol. 830, no. 2. doi: 10.1088/1757-899X/830/2/022090.
[8] Arief Prasetyo, Sofyan Noor Arief, and Rokhimatul wakhidah, "OPTIMASI PEMROSESAN ENKRIPSI DAN DEKRIPSI RSA PADA SINGLE BOARD COMPUTER (SBC) DENGAN PEMBAGIAN BEBAN KOMPUTASI DALAM SISTEM TERDISTRIBUSI," Jurnal Informatika Polinema, vol. 7, no. 4, 2021, doi: 10.33795/jip.v7i4.523.
[9] X. Zhong and Y. Liang, "Raspberry Pi: An effective vehicle in teaching the internet of things in computer science and engineering," Electronics (Switzerland), 2016, doi: 10.3390/electronics5030056.
[10] M. I. Suriansyah, "Perbandingan Kinerja Pemrosesan Paralel Pada PC dan Raspberry Pi Untuk Pendeteksian Gulma Pada Lahan Pertanian Menggunakan Fraktal," 2015.
[11] E. Irvi, "Rancang Bangun dan Evaluasi Kinerja Raspberry Pi Cluster sebagai Platform Penerapan Pembelajaran Mesin," 2019.
[12] V. A. H. Firdaus, P. Y. Saputra, and D. Suprianto, "Intelligence chatbot for Indonesian law on electronic information and transaction," in IOP Conference Series: Materials Science and Engineering, 2020, vol. 830, no. 2. doi: 10.1088/1757-899X/830/2/022089.
[13] B. Qureshi and A. Koubaa, "On energy efficiency and performance evaluation of single board computer based clusters: A hadoop case study," Electronics (Switzerland), vol. 8, no. 2, 2019, doi: 10.3390/electronics8020182.
[14] B. Sukesh, K. Venkatesh, and L. N. B. Srinivas, "A Custom Cluster Design With Raspberry Pi for Parallel Programming and Deployment of Private Cloud," in Role of Edge Analytics in Sustainable Smart City Development, 2020. doi: 10.1002/9781119681328.ch14.
[15] S. Parra et al., "Development of Low-Cost Point-of-Care Technologies for Cervical Cancer Prevention Based on a Single-Board Computer," IEEE Journal of Translational Engineering in Health and Medicine, vol. 8, 2020, doi: 10.1109/JTEHM.2020.2970694.
[16] P. J. Basford et al., "Performance analysis of single board computer clusters," Future Generation Computer Systems, vol. 102, 2020, doi: 10.1016/j.future.2019.07.040.
[17] C. Baun, "Mobile clusters of single board computers: an option for providing resources to student projects and researchers," SpringerPlus, vol. 5, no. 1, 2016, doi: 10.1186/s40064-016-1981-3.

[18] S. J. Johnston et al., "Commodity single board computer clusters and their applications," Future Generation Computer Systems, vol. 89, 2018, doi: 10.1016/j.future.2018.06.048.
[19] P. L. Ching, J. E. Mutuc, and J. A. Jose, "Assessment of the quality and sustainability implications of FIFO and LIFO inventory policies through system dynamics," Advances in Science, Technology and Engineering Systems, vol. 4, no. 5, 2019, doi: 10.25046/aj040509.
[20] M. Baclet and C. Pagetti, "Around hopcroft's algorithm," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2006, vol. 4094 LNCS. doi: 10.1007/11812128_12.
[21] H. J. Herrmann and H. E. Stanley, "On the growth of percolation clusters: The effects of time correlations," Zeitschrift für Physik B Condensed Matter, vol. 60, no. 2–4, 1985, doi: 10.1007/BF01304435.
[22] H. Putra Rajagukguk, M. Zarlis, and Sutarman, "Analysis of Priorities Queuing Model (N-P) System with Multiple Channel Multiple Phase and Simple Additive Weighting," in Journal of Physics: Conference Series, 2019, vol. 1339, no. 1. doi: 10.1088/1742-6596/1339/1/012033.
[23] "Operating systems: design and implementation," Microprocessors and Microsystems, vol. 11, no. 5, 1987, doi: 10.1016/0141-9331(87)90310-3.
[24] G. Dorr, D. Hagen, B. Laskowski, E. Steinmetz, and D. Vo, "Introduction To Parallel Processing With Eight Node Raspberry Pi Cluster," Metaheuristics International Conference, 2017.
[25] H. I. M. Al-Salman, P. Ehkan, and M. H. Al-Doori, "FPGA-based Design of Multiple Models for Industry 4.0 Cyber Security," in Journal of Physics: Conference Series, 2021, vol. 1997, no. 1. doi: 10.1088/1742-6596/1997/1/012031.