ANALISIS ALGORITMA KONSTRUKSI HIMPUNAN SUBMATRIKS MINIMAL DARI MATRIKS SIRKULAN

I Gusti Bagus Rai Nanditha[1]

[1]Universitas Mataram, rainanditha98@gmail.com

Abstract. Data security is crucial in today's technological era, and cryptography, especially the Block Cipher method, is used to safeguard computer network security by dividing messages into blocks of bits. However, attacks like differential attack and linear attack need to be vigilant against. The use of MDS matrices in Block Cipher can mitigate these attacks and enable the recovery of lost information. In the selection of efficient error recovery algorithms, time complexity becomes a key factor. Therefore, this research aims to analyze the complexity of algorithms for constructing sets of minimal submatrices in circular matrices using simulation and time complexity analysis. The analysis concludes that the program's time complexity depends on the complexity of each component, including the partition algorithm, circular shift elimination, I_u construction, $\mathcal{J}_u$ construction, and sorting algorithm. The analysis results indicate the time complexity of each component, including $O(p(n))$ for the partition algorithm, $O(p(n))$ for circular elimination, $O(\frac{p(n)}{n})$ for I_u construction, and $O(2^n)$ for $\mathcal{J}_u$ construction. In summary, the total time complexity of the program can be $O(p(n))$, depending on the parameters used and the input size.

Keywords: *Cryptography, Block Cipher, Maximum Distance Separable, Circular Matrix, Algorithm Complexity.*

Abstrak. Keamanan data di era teknologi saat ini sangat penting, dan kriptografi, khususnya metode Block Cipher, digunakan untuk menjaga keamanan jaringan komputer dengan membagi pesan menjadi blok-bit. Namun, serangan seperti differential attack dan linear attack perlu diwaspadai. Menggunakan matriks MDS dalam Block Cipher dapat mengatasi serangan ini dan memungkinkan pemulihan informasi yang hilang. Dalam pemilihan algoritma pemulihan kesalahan yang efisien, kompleksitas waktu menjadi faktor kunci. Oleh karena itu, penelitian ini bertujuan untuk menganalisis kompleksitas algoritma konstruksi himpunan dari submatriks minimal pada matriks sirkulan dengan menggunakan simulasi dan analisis kompleksitas waktu. Analisis menyimpulkan bahwa kompleksitas waktu program bergantung pada kompleksitas setiap bagian, Algoritma partisi, eliminasi sirkular shift, konstruksi I_u, konstruksi $\mathcal{J}_u$ dan algoritma pengurutan. Hasil analisis menunjukkan kompleksitas waktu setiap bagian, antara lain $O(p(n))$ untuk algoritma partisi, $O(p(n))$ untuk eliminasi siklik, $O\left(\frac{p(n)}{n}\right)$ untuk konstruksi I_u, $O(2^n)$ untuk $\mathcal{J}_u$ konstruksi. Singkatnya, kompleksitas waktu total dari program dapat $O(p(n))$ bergantung pada parameter yang digunakan dan ukuran input.

Kata kunci: *Kriptografi, Block Cipher, Maximum Distance Separable, Matriks Sirkulan, Kompleksitas Algoritma.*

1 Pendahuluan

Keamanan data sangat penting dalam era teknologi saat ini. Kebocoran data dapat dimanfaatkan oleh pihak yang tidak bertanggung jawab. Kriptografi, termasuk hash function, public key cryptography, dan *symmetric key cryptography*, adalah alat yang digunakan untuk mengamankan jaringan komputer. Kriptografi merupakan ilmu dan seni untuk menjaga keamanan pesan yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan, integritas, serta otentikasi [1]. Kriptografi sangat diperlukan dalam menjaga keamanan informasi dengan menggunakan proses enkripsi dan dekripsi untuk menjalankannya. Salah satu cara yang dapat dilakukan untuk mengenkripsi terhadap bit-bit pesan adalah menggunakan metode *Block Chiper*. Teknik kriptografi ini merupakan salah satu dari algoritma kriptografi modern yang masih dipakai sampai saat ini. Metode *block chiper* beroperasi dengan membagi bit-bit plainteks menjadi sekumpulan bit atau blok yang sama panjang. *Block Chiper* berfungsi untuk meningkatkan keamanan dari suatu pesan dengan menggabungkan operasi atau perhitungan sederhana seperti substitusi yang dilakukan dalam beberapa putaran.

Terdapat dua *attack* atau serangan yang sering ditemui pada *block chiper* yaitu *differential attack* dan *linear attack.* Untuk mencegah terjadinya serangan-serangan tersebut dapat menggunakan *block chiper* yang melibatkan matriks *Maximum Distance Separable* (MDS). Salah satu cara dalam melakukan konstruksi matriks MDS adalah dengan menggunakan matriks sirkulan. Dalam MDS, tujuan utama adalah menciptakan kode pemulihan kesalahan yang memungkinkan untuk memulihkan informasi yang hilang atau rusak dengan menggunakan jumlah minimum bit tambahan.

Analisis kompleksitas waktu menjadi krusial dalam memilih dan merancang algoritma pemulihan kesalahan yang efisien untuk sistem MDS, mengingat algoritma yang lambat dapat mengurangi efektivitas dan kegunaan dari sistem tersebut. Oleh karena itu, dalam penelitian ini pemahaman tentang kompleksitas waktu menjadi faktor kunci dalam pemilihan algoritma yang sesuai untuk tugas yang diberikan. Metode penelitian umumnya melibatkan implementasi algoritma dalam bahasa pemrograman tertentu, pengujian dengan berbagai data uji, dan analisis hasil pengukuran untuk menentukan kompleksitas waktu masing-masing algoritma.

2 Metode Penelitian

2.1 Studi Literatur

Penelitian ini akan menganalisis algoritma yang dikembangkan oleh Malakhov [2] untuk memahami kompleksitas waktu dan kinerjanya dalam kriptografi yang berjudul *Construction of a set of circulant matrix submatrices for faster MDS matrix verification*. Adapun algoritma yang akan digunakan adalah Algoritma 1 yang berhasil dibuat oleh Malakhov [2] melalui penelitiannya yaitu "*Construction of a set of all non-equivalent submatrices of a circulant matrix without equivalence classes of transposes or anti-transposes*".

2.2 Eksperimen Aplikasi Algoritma

Metode yang digunakan dalam penelitian ini adalah dilakukan dengan membuat program dari algoritma yang telah dibuat oleh Malakhov [2], proses pembuatan program menggunakan Bahasa pemrograman *Python* pada *Jupyter Notebook*. Kemudian dijalankan dengan membuat *syntax* yang polanya mengikuti dari Algoritma 1 tersebut,

2.3 Menguji Program Algoritma

Dari hasil eksperimen sebelumnya kemudian akan dilanjutkan dengan menguji *syntax* yang telah dibuat, jika berhasil maka akan dilakukan analisis yang teliti terhadap kinerja serta hasil yang diperoleh dari program tersebut dan jika gagal maka akan diulangi tahap sebelumnya sampai hasil dari uji yang dilakukan berhasil.

2.4 Analisis Algoritma

Analisis algoritma merupakan proses yang dilakukan untuk menentukan kompleksitas waktu dari algoritma tersebut. Karena hasil dari tahap pengujian sudah berhasil, yang mengindikasikan bahwa Algoritma 1 yang dibuat oleh Malakhov [2] bisa menyelesaikan masalah *Maximum Distance Separable* pada matriks sirkulan.

2.5 Penarikan Kesimpulan

Tahapan terakhir dari metode penelitian ini adalah dengan melakukan penarikan kesimpulan dari hasil dan pembahasan yang di tunjukkan pada penelitian ini.

3 Hasil dan Pembahasan

Dalam menjalankan analisis kompleksitas waktu pada *syntax* simulasi Algoritma 1, hanya bagian-bagian tertentu dalam program yang memiliki kompleksitas waktu yang signifikan dan memerlukan analisis. Pada umumnya, bagian yang memiliki pengulangan (*loop*), pemrosesan data yang besar, atau pemanggilan fungsi rekursif adalah bagian-bagian yang memerlukan analisis kompleksitas waktu. Namun, ada juga bagian program lainnya yang tidak memiliki kompleksitas waktu yang signifikan, seperti deklarasi variabel atau operasi aritmatika sederhana.

3.1 Partisi

Dengan menggunakan pendekatan dinamis, semua partisi unik yang diperlukan dari suatu angka dapat dihasilkan. Kompleksitas dari pendekatan di atas adalah $O(k)$, di mana k merupakan jumlah partisi unik yang perlu dihasilkan. Nilai maksimum untuk k adalah 2^n, karena terdapat total 2^n partisi yang mungkin. Oleh karena itu, pendekatan ini memiliki kompleksitas waktu terburuk $O(2^n)$ [3].

Selanjutnya memisahkan/partisi dengan panjang tertentu, untuk rumus umumnya tidak ada atau masih belum diketahui untuk fungsi partisi. Namun, ada ekspansi atau pendekatan asimptotik yang dapat memberikan perkiraan yang akurat dan hubungan rekurensi yang memungkinkan dihitung secara tepat. Pertumbuhan fungsi ini bersifat eksponensial terhadap akar kuadrat yang dapat di jelaskan dengan fungsi berikut,

$$p(n) \sim \frac{1}{4n\sqrt{3}} exp\left(\pi\sqrt{\frac{2n}{3}}\right) \text{ untuk } n \to \infty$$

dalam menentukan panjang partisi untuk ukuran tertentu dilakukan perulangan (*looping*) untuk mendapatkan partisi untuk setiap u sebanyak $p(n)$, sehingga kompleksitas waktunya adalah $O(p(n))$ [4].

Jadi kompleksitas pada bagian partisi ini jika dijumlahkan akan menjadi $O(2^n + p(n))$, dari penjumlahan kompleksitas waktu tersebut dapat disederhanakan menjadi $O(p(n))$ karena dalam kompleksitas waktu untuk kasus *worst case*, nilai kompleksitas waktu yang paling dominan atau yang paling tinggi yang akan menjadi nilai kompleksitas akhir.

3.2 Menghapus Pergeseran Siklik

Kompleksitas waktu untuk menghapus semua pergeseran siklik adalah dengan mengetahui total partisi untuk setiap u, Dalam konteks ini, ketika mencoba menghapus semua pergeseran siklik, perlu dipertimbangkan setiap elemen u dalam himpunan tersebut. Dengan mengetahui jumlah partisi total untuk setiap elemen u, kompleksitas waktu yang terlibat dalam operasi ini dapat dipahami. Untuk mendapatkan informasi tentang jumlah partisi total, perlu dilakukan perulangan (looping) pada setiap elemen u dalam himpunan.maka untuk mengetahuinya dilakukan perulangan (*looping*) pada setiap u sehingga kompleksitas waktunya akan sama yaitu $O(p(n))$.

3.3 Konstruksi I_u

Kompleksitas waktu dari konstruksi I_u diperoleh dari hasil menghapus pergeseran siklik yang sama. Jadi untuk setiap satu partisi kumingkanan akan memiliki perulangan sebanyak n kali, sehingga akan tersisa sebanyak $\frac{p(n)}{n}$ sehingga kompleksitas waktu dari I_u adalah $O\left(\frac{p(n)}{n}\right)$.

3.4 Konstruksi $\mathcal{J}_u$

Untuk memperoleh kompleksitas waktu total dari mencari banyaknya subset untuk ukuran tertentu dengan,

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n$$

Berdasarkan teorema binomial berikut,

$$(a+b)^n = \binom{n}{0} a^n b^0 + \binom{n}{1} a^{n-1} b^1 + \binom{n}{2} a^{n-2} b^2 + \cdots + \binom{n}{n-1} a^1 b^{n-1} + \binom{n}{n} a^0 b^n$$

Dimana setiap $\binom{n}{k}$ adalah bilangan bulat positif tertentu yang dikenal sebagai koefisien binomial. Rumus ini dikenal juga sebagai rumus binomial atau identitas binomial. Dengan menggunakan notasi penjumlahanm dapat ditulis sebagai berikut,

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}$$

Ekspresi akhir mengikuti ekspresi sebelumnya dengan menukar letak x dan y dari ekspresi pertama. Dari perbandingan keduanya, dapat diketahui bahwa urutan koefisien binomial dalam rumus tersebut bersifat simetris. Untuk mendapatkan

kompleksitas waktunya, dapat menggunakan pendekatan dinamis seperti pada kompleksitas waktu untuk partisi, sehingga kompleksitas waktunya adalah $O(2^n)$.

Terakhir adalah menghitung kompleksitas waktu untuk algoritma *sorting* (pengurutan), salah satu algoritma pengurutan kompleksitas waktu terburuk adalah algoritma *Bubble Sort*. Kompleksitas waktu *Bubble Sort* adalah $O(n^2)$, di mana n adalah jumlah item yang akan disortir. Ini berarti bahwa waktu yang diperlukan untuk mengurutkan item bertambah karena jumlah item yang diurutkan secara langsung bertambah [5]. Jadi kompleksitas untuk konstruksi $\mathcal{J}_u$ adalah $O(2^n + n^2)$, kemudian disederhanakan menjadi $O(2^n)$ karena nilai tersebut yang paling dominan.

4 Kesimpulan

Dalam pembahasan di atas telah dicatat bahwa beberapa bagian dari program ini adalah kompleks waktu. Bagian-bagian ini termasuk algoritma partisi, menghapus pergeseran siklik, konstruksi I_u , konstruksi $\mathcal{J}_u$, dan algoritma pengurutan. Kompleksitas waktu dari setiap bagian dianalisis dan disimpulkan bahwa kompleksitas waktu dari keseluruhan program bergantung pada kompleksitas waktu dari bagian-bagian tersebut. Kompleksitas waktu dari algoritma partisi adalah $O(p(n))$ dalam kasus terburuk, sedangkan kompleksitas waktu menghapus semua pergeseran siklik adalah $O(p(n))$. Kompleksitas waktu konstruksi I_u adalah $O\left(\frac{p(n)}{n}\right)$ sedangkan kompleksitas waktu konstruksi $\mathcal{J}_u$ total adalah $O(2^n)$.

Oleh karena itu, kesimpulan akhir adalah bahwa kompleksitas waktu total program dapat mencapai $O(p(n))$ dalam kasus tertentu, bergantung pada parameter yang digunakan dan ukuran input. Dengan mengetahui kompleksitas waktu algoritma Malakhov (2021), peneliti dapat memperkirakan seberapa efisien algoritma tersebut dalam menyelesaikan masalah konstruksi dan verifikasi matriks MDS pada berbagai ukuran masukan. Selain itu, analisis ini memungkinkan kita membandingkan kinerja algoritma Malakhov [2] dengan algoritma-algoritma lain yang serupa dalam konteks masalah yang sama.

5 Daftar Pustaka

[1] Lusiana, V. and Hadikurniawati, W. (2012). Kriptografi Kunci Publik (*Public KeyCryptography*).

[2] Malakhov, S. S. (2021). *Construction of a set of circulant matrix submatrices for faster MDS matrix verification*. arXiv preprint arXiv:2110.13325.

[3] Tikariha, S. (2019). *Number of Unique Partitions of an Integer* (https://iq.opengenus.org/number-of-unique-partitions-of-an-integer/). Diakses pada pukul 19:40 WITA, tanggal 26/06/2023.

[4] JayBeeEll. (2023). *Partition Function (Number Theory).* Dalam Wikipedia : Ensiklopedia (*https://en.wikipedia.org/wiki/Partition_(number_theory)*). Diakses pada pukul 20:00 WITA, tanggal 26/06/2023.

[5] McConnell, Jeffrey. (2001). *Analysis of algorithms: an active learning approach* . Sudbury: Jones and Bartlett Publishers.