**Journal of Computer System and Informatics (JoSYC)**
ISSN 2714-8912 (media online), ISSN 2714-7150 (media cetak)
Volume 3, No. 4, August 2022, Page 303−312
https://ejurnal.seminar-id.com/index.php/josyc
DOI 10.47065/josyc.v3i4.2109

# Kinerja Algoritma Yamamoto's Recursive Code dan Algoritma Fixed Length Binary Encoding pada Kompresi File PDF

**Yulrismawati Sihura, Taronisokhi Zebua*, Hukendik Hutabarat**

Ilmu Komputer dan Teknologi Informasi, Program Teknik Informatika, Universitas Budi Darma, Medan, Indonesia
Email: [1]yulrisna30@gmail.com, [2,*]taronizeb@gmail.com
Submitted: **15/08/2022**; Accepted: **24/08/2022**; Published: **30/08/2022**

**Abstrak**−Pemanfaatan *file* dalam bentuk digital saat ini semakin berkembang dan membutuhkan media untuk menyimpannya. Bila semakin banyak *file* yang disimpan, maka semakin besar ruang penyimpanan yang dibutuhkan. Hal ini mendorong perkembangan teknik pengecilan *file* atau dikenal dengan teknik kompresi data dengan tujuan agar ruang penyimpanan yang dibutuhkan menjadi lebih sedikit. Teknik kompresi memiliki beberapa algoritma yang dapat digunakan untuk memampatkan *file* seperti algoritma *yamamoto's recursive code* dan *Fixed Length Binary Encoding* (FLBE). Kedua algoritma ini memiliki kinerja yang berbeda untuk menghasilkan kualitas hasil kompresi, sehingga perlu dibandingkan. Penelitian ini menguraikan analisa dan perbandingan kinerja dari kedua algoritma berdasarkan Metode Perbandingan *Exponential* (MPE) yang diukur berdasarkan nilai-nilai parameter kualitas hasil kompresi *file* pdf. Metode *exponential* atau metode perbandingan *exponential* adalah salah satu metode dari *Decision Suport System* (DSS) untuk menentukan urutan prioritas alternatif keputusan dengan kriteria jamak. Parameter kualitas kompresi yang diukur adalah nilai *ratio of compression*, *compression ratio* dan *space saving*. Berdasarkan hasil perbandingan yang dilakukan berdasarkan Metode Perbandingan *Exponential* (MPE), didapatkan bahwa algoritma *fixed length binary encoding* memiliki kinerja yang lebih baik dengan nilai 2,55% dibandingkan dengan algoritma *yamomoto's recursive code* dengan nilai yang lebih kecil yaitu 2,22 %.

**Kata Kunci:** Kompresi; *File* PDF; Yamamoto's; FLBE; Perbandingan Algoritma

**Abstract**−The use of files in digital form is currently growing and requires media to store them. The more files that store, the more storage space need. This encourages the development of file reduction techniques or data compression techniques with the aim of reducing the required storage space. The compression technique has several algorithms that can be used to compress files such as Yamamoto's recursive code algorithm and Fixed Length Binary Encoding (FLBE). These algorithms have different performance to produce quality compressed results, so they need to be compared. This study describes the analysis and comparison of the performance of the two algorithms based on the exponential method as measured by the quality of the compressed pdf file. The exponential comparison method is one of the methods of the Decision Support System (DSS) to determine the priority order of decision alternatives with multiple criteria. Compression quality parameters that are measured as an alternative comparison are the value of the ratio of compression, compression ratio, space saving. Based on the results of the comparison with Exponential Comparison Method (MPE), it was found that the fixed length binary encoding algorithm has a better performance with a value of 2.55% compared to the Yamomoto's recursive code algorithm with a smaller value of 2.22%.

**Keywords**: Compression; PDF File; Yamamoto's; FLBE; Algorithms Compare

## 1. PENDAHULUAN

Ruang penyimpanan yang dibutuhkan dalam dunia digital saat ini semakin meningkat karena adanya kecenderungan untuk selalu menyimpan *file* atau dokumen. Semakin banyak *file* yang disimpan, maka semakin banyak penyimpanan yang dibutuhkan. Salah satu hal yang diperlukan untuk mengatasi permasalahan ini adalah metode pengecilan *file* atau dikenal dengan teknik kompresi data dengan tujuan agar ruang penyimpanan yang dibutuhkan menjadi lebih sedikit.

Ada berbagai algoritma kompresi data yang dapat digunakan dalam mengkompresi data seperti algoritma yang digunakan dalam penelitian ini yaitu algoritma *yamamoto's recursive code* dan algoritma *Fixed Length Binary Encoding* (FLBE). Algoritma *yamamoto's recursive code* merupakan kode *recursive* untuk angka positif dimana setiap urutan yang diberikan dapat digunakan sebagai delimiter, berbeda dengan *universal code* lainnya yang menggunakan bit "0" sebagai delimiter seperti *Elias Omega Code, Even-Rodeh Code, Stout Code*, dan lain-lain [1]. Algoritma FLBE merupakan algoritma kompresi yang bekerja dengan merubah simbol asli ke dalam bentuk *fixed-length code*. Salah satu kelebihan dari algoritma ini adalah bila panjang bit karakternya mencapai jumlah karakter yang terakhir, maka proses kompresi akan berhenti, sehingga algoritma ini cukup baik dalam mengkompresi *file* teks [2].

Walaupun banyak algoritma kompresi yang berkembang hingga saat ini, namun masing-masing algoritma memiliki kelebihan dan kekurangan masing-masing dalam pengukuran kinerjanya [3]. Optimalisasi kinerja algoritma kompresi dapat diketahui dengan melakukan perhitungan kinerjanya berdasarkan parameter kompresi. Parameter dalam mengukur kinerja algoritma untuk kompresi seperti *Ratio Of Compression* (RC), *Compression Ratio* (CR), dan *Space Saving* (SS ).

Berdasarkan penelitian sebelumnya yang menganalisa perbandingan algoritma *prediction by partial matching* dengan *sequitur* pada kompresi *file* teks, mengatakan bahwa alasan dialakukannya perbandingan algoritma kompresi adalah untuk mengetahui algoritma mana yang lebih baik digunakan atau algoritma mana yang lebih mudah digunakan untuk melakukan kompresi suatu *file* pdf [4].

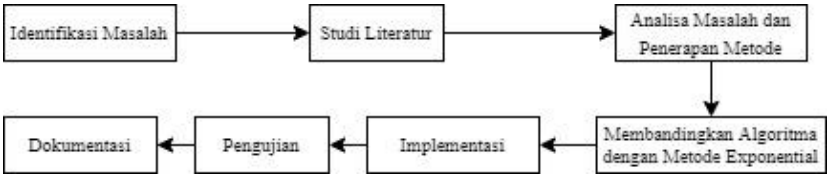
**Journal of Computer System and Informatics (JoSYC)**
ISSN 2714-8912 (media online), ISSN 2714-7150 (media cetak)
Volume 3, No. 4, August 2022, Page 303−312
https://ejurnal.seminar-id.com/index.php/josyc
DOI 10.47065/josyc.v3i4.2109

Metode Perbandingan Eksponensial (MPE) merupakan salah satu metode yang dapat digunakan menentukan urutan prioritas alternatif keputusan dengan kriteria jamak. Berdasarkan penelitian sebelumnya yang melakukan analisa terhadap perbandingan algoritma *Rice Codes* dengan algoritma *Goldbach Codes* pada kompresi *file text* menggunakan metode *exponential*, mengatakan bahwa metode eksponensial sangat cocok dan baik digunakan untuk membandingkan algoritma karena dapat mengurangi bias yang mungkin terjadi sehingga dapat menyebabkan urutan prioritas alternatif keputusan menjadi lebih nyata[5].

Penelitian ini menguraikan bagaimana mengukur kinerja kedua algoritma *yamamoto's recrusive code* dan *fixed length binary encoding* pada kompresi *file* pdf yang dilakukan berdasarkan pengukuran nilai kualitas hasil kompresi. Paramater nilai kualitas kompresi yang digunakan adalah nilai *ratio compression*, *compression ratio* dan *space saving*. Perbandigan keduanya dilakukan melalui formulasi metode eksponensial dengan tujuan dapat dijadikan sebagai referensi bagi pengguna dalam pemilihan algoritma kompresi yang lebih baik.

## 2. METODOLOGI PENELITIAN

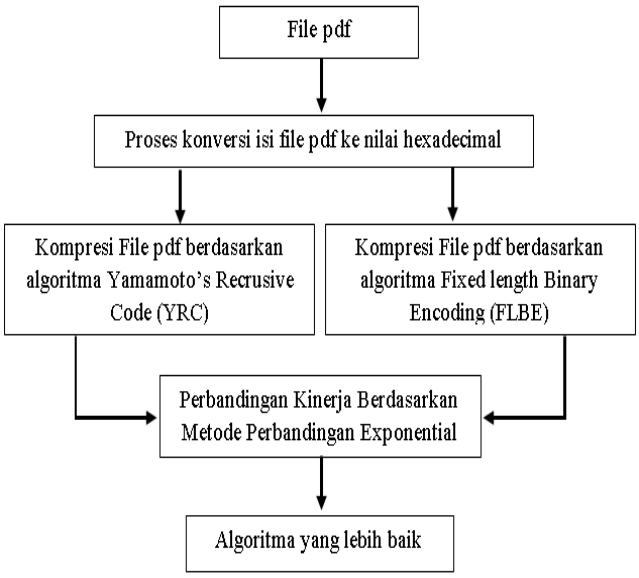
Adapun metode yang dilakukan dalam menyelesaikan penelitian ini disajikan pada gambar 1.




**Gambar 1.** Tahap Penelitian

Penelitian ini diawali dengan tahapan identifikasi terhadap permasalahan yang terkait dengan ukuran *file* pdf yang mempengaruhi kebutuhan ruang penyimpanan. Agar dasar penelitian ini kuat, maka dilakukan studi pustaka melalui referensi yang terkait dengan permasalahan dalam penelitian ini baik dalam bentuk buku maupun jurnal. Tahap selanjutnya adalah melakukan analisa serta penerapan dua algoritma pada kompresi *file* pdf. Hasil kompresi dari kedua algoritma tersebut akan dibandingkan mengunakan metode *exponential* untuk mendapatkan kinerja dari masing-masing algoritma. Tahap selanjutnya adalah mengimplementasikan solusi ke dalam bentuk aplikasi yang dibangun melalui Bahasa pemrograman *visual basic net* 2008. Setelah tahap implementasi, maka akan dilakukan pengujian untuk memastikan apakah aplikasi yang dibangun telah memberikan hasil yang sesuai dengan pengerjaan manual. Tahap akhir adalah tahap dokumentasi seluruh rangkaian kegiatan penelitian ini dalam bentuk laporan penelitian.

Penelitian ini berfokus pada proses perbandingan kinerja algoritma *yamamoto's recursive code* dan algoritma *Fixed Length Binary Encoding* (FLBE) dalam mengkompresi *file* pdf yang diukur melalui nilai-nilai parameter kualitas hasil kompresi dari masing-masing algoritma. Proses kompresi *file* pdf diawali dengan pembacaan nilai-nilai *file* pdf yang akan dikompresi, kemudian dilakukan proses pemampatan terhadap nilai-nilai biner *file* pdf tersebut berdasarkan algoritma kompresi yang digunakan. Proses ini akan menghasilkan *file* pdf baru yang terkompresi. Algoritma FLBE dan *yamamoto's recursive code* ,merupakan dua algoritma yang digunakan dalam penelitian ini untuk melakukan proses kompresi *file* pdf.




**Gambar 2.** Prosedur Kompresi *File* PDF

**Journal of Computer System and Informatics (JoSYC)**
ISSN 2714-8912 (media online), ISSN 2714-7150 (media cetak)
Volume 3, No. 4, August 2022, Page 303−312
https://ejurnal.seminar-id.com/index.php/josyc
DOI 10.47065/josyc.v3i4.2109

### 2.1 Kompresi

Kompresi adalah proses yang bertujuan untuk memampatkan atau mengecilkan ukuran suatu data atau proses mengkonversi *input* data *stream* (aliran sumber) menjadi aliran data yang lain (*output*, *bitstream*, aliran terkompresi) dengan ukuran yang lebih kecil. Sebuah *stream* dapat berupa *file* atau *buffer* di memori. Kompresi terdiri dari 2 komponen algoritma yaitu algoritma *encoding*, dimana data akan dipresentasikan ke dalam bentuk lain yang lebih kecil dan algoritma *decoding*, dimana data yang sudah terkompresi dipresentasikan ke bentuk awal data tersebut [6].

Tujuan kompresi data adalah mengurangi jumlah *bit* yang dihasilkan dari setiap simbol yang muncul dan mengurangi data yang berlebihan (Redudansi Data) sehingga ukuran data menjadi lebih kecil dan mengurangi pemakaian ruang penyimpanan [7]. Secara garis besar terdapat dua jenis kompresi data yaitu *loseless data compression* dan *losy data compression* [1]. Terdapat faktor-faktor atau parameter yang dapat digunakan untuk mengukur kualitas hasil dari teknik kompresi data yaitu *ratio of compression*, *compression ratio*, *space saving*, *redundancy* [8].

### 2.2 Algoritma Yamamoto's Recrusive Code

*Yamamoto's Recursive Code* merupakan kode *recursif* untuk angka positif yang dimana setiap urutan yang diberikan dapat digunakan sebagai delimeter. Berbeda dengan *universal code* lainnya yang menggunakan bit "0" sebagai delimeter seperti *Elias Omega Code*, *Even-Rodeh*, *Stout Code*, dan lain-lain. Delimeter yang dipakai pada algoritma *Yamamoto's Recursive Code* lebih pendek dari $log^*_2$ n dalam hampir dari semua angka bulat positif dibandingkan algoritma sebelumnya [9].

### 2.3 Algortima Fixed Length Binary Encoding

Algoritma *Fixed Length Binary Encoding* (FLBE) sering dikenal sebagai kode blok. Kemudahan dalam mengubah simbol yang asli ke dalam bentuk *fixed-length code* menjadikan algoritma ini mudah untuk diimplementasikan ke dalam *software*. Kemudahan ini dikatakan berbanding terbalik dengan proses pengubahan kode dengan panjang tetap menjadi simbol aslinya [2]. Algoritma *Fixed Length Binary Encoding* (FLBE) akan menggunakan karakter dalam *string* yang ingin dikompres dan menghitung frekuenasi karakter tersebut. *Fixed Length Binary Encoding* (FLBE) mengubah *bit* karakter menjadi *fixed-length code* sehingga menghasilkan *string bit* yang baru. *String bit* inilah yang merupakan hasil kompresi dari algoritma *Fixed Length Binary Encoding* (FLBE) [10].

### 2.4 Metode Exponential

Metode ini dirancang untuk membantu pengambil keputusan individu dalam memanfaatkan desain model yang terdefinisi dengan baik pada tahap awal proses. Metode *Exponential* akan menghasilkan nilai alternatif yang perbedaannya lebih kontras [11]. Beberapa tahap penerapan metode *exponential* [12], yaitu:

a. Menentukan alternatif analisa perbandingan kecepatan antar kedua algoritma.
b. Menentukan kriteria untuk membandingkan kedua algoritma dan menentukan kriteria dalam menganalisis proses dan cara kerjanya.
c. Pemberian nilai dari setiap kriteria yang telah ditetapkan.
d. Menentukan hasil atau prioritas keputusan berdasarkan nilai dari setiap alternatif.

Formulasi perhitungan skor untuk setiap alternatif dalam metode perbandingan eksponensial [5], sebagai berikut :

$$\textit{TotalNilai}\ (\text{TNi}) = \sum_{j=1}^{m} (V_{ij})\text{B}_\text{j} \quad (1)$$

Keterangan :

TNi = Total nilai alternatif ke-i
Vij = Derajat kepentingan relatif kriteria ke-j pada pilihan keputusan ke-i yang dapat dinyatakan dengan bobot.
Bj = Derajat kepentingan kriteria keputusan yang dinyatakan dengan bobot.

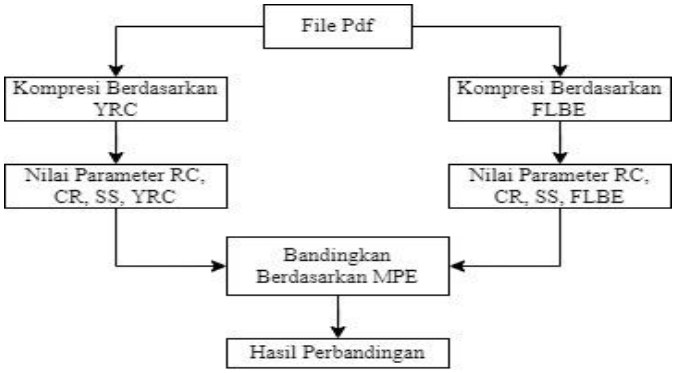
## 3. HASIL DAN PEMBAHASAN

Masalah yang dibahas dalam penelitian ini adalah pengukuran kinerja dari dua algoritma kompresi yaitu kinerja algoritma *yamamoto's recursive code* dan algoritma *Fixed Length Binary Encoding* (FLBE) dalam kompresi *file* pdf. Pengujian kinerja dari kedua algoritma menggunakan data uji yang sama yaitu sebuah *file* pdf. Pengukuran kinerja dari dua algoritma ini diukur berdasarkan kualitas hasil proses kompresi melalui nilai *Ratio of Compression* (RC), *Compression Ratio* (CR), dan *Space Saving* (SS). Agar didapatkan algoritma yang kinerjanya lebih baik, maka hasil dari parameter pengukuran kualitas kompresi dari masing-masing algoritma perlu dibandingkan. Metode yang digunakan untuk membandingkan kedua algoritma ini adalah Metode Perbandingan *Exponential* (MPE). Hasil perbandingan yang didapatkan dapat digunakan untuk mengetahui algoritma mana yang lebih baik digunakan untuk menghasilkan kualitas kompresi yang lebih baik khususnya pada kompresi *file* pdf.

**Journal of Computer System and Informatics (JoSYC)**
ISSN 2714-8912 (media online), ISSN 2714-7150 (media cetak)
Volume 3, No. 4, August 2022, Page 303−312
https://ejurnal.seminar-id.com/index.php/josyc
DOI 10.47065/josyc.v3i4.2109

### 3.1 Perbandingan Algoritma Kompresi

Perbandingan algoritma kompresi perlu dilakukan untuk mengetahui kinerja dari algoritma yang dibandingkan. Kinerja algoritma yang digunakan dalam penelitian ini dapat diketahui dengan mengukur nilai parameter kualitas hasil kompresi, yaitu nilai *ratio of compression*, *compression ratio* dan *space saving*. Metode yang digunakan dalam proses perbandingan algortima yang digunakan dalam penelitian ini adalah metode *exponential*.




**Gambar 3.** Prosedur Perbandingan Algoritma Kompresi

*File* pdf yang digunakan sebagai contoh dalam penelitian ini adalah *file* pdf dengan *size* 7,90 MB, namun untuk mempermudah proses penerapan metode, maka diambil 20 *byte* sebagai sampel. Nilai-nilai pdf yang dijadikan sampel dalam penelitian ini (dalam bentuk hexa) disajikan pada tabel 1 berikut ini.

**Tabel 1.** Nilai *file* pdf sampel

| Index | Hexa | Biner | Index | Hexa | Biner |
|---|---|---|---|---|---|
| 1 | 25 | 00100101 | 11 | F3 | 11110011 |
| 2 | 50 | 01010000 | 12 | A0 | 10100000 |
| 3 | 44 | 01000100 | 13 | D0 | 11010000 |
| 4 | 46 | 01000110 | 14 | C4 | 11000100 |
| 5 | 2D | 00101101 | 15 | C6 | 11000110 |
| 6 | 31 | 00110001 | 16 | 0A | 00001010 |
| 7 | 2E | 00101110 | 17 | 34 | 00110100 |
| 8 | 33 | 00110011 | 18 | 20 | 00100000 |
| 9 | 0A | 00001010 | 19 | 30 | 00110000 |
| 10 | 25 | 00100101 | 20 | 20 | 00100000 |
| Total *bit* sampel adalah 160 *bit* | | | | | |

### 3.2 Penerapan Algoritma Yamamoto's Recursive Code

Algoritma *yamamoto's recursive code* mengawali proses kompresi dengan mengurutkan nilai-nilai *file* pdf berdasarkan frekuensi kemunculan. Biner dari setiap nilai pdf tersebut akan dikodekan berdasarkan kode algoritma *yamamoto's recursive code* secara berurut, hingga dihasilkan *bit file* pdf terkompresi. Penentuan *codeword* dilakukan dengan mengikuti aturan nilai-nilai tabel *codeword* algoritma *yamamoto's recrusive code*.

**Tabel 2.** Pengurutan frekuensi dan *codeword* nilai pdf

| n | Hexa | Frek | ASCII Biner | *Codeword* | *Bit* | *Bit* × Frekuensi |
|---|---|---|---|---|---|---|
| 1 | 25 | 2 | 00100101 | 100 | 3 | 6 |
| 2 | 0A | 2 | 00001010 | 10100 | 5 | 10 |
| 3 | 20 | 2 | 00100000 | 11000 | 5 | 10 |
| 4 | 50 | 1 | 01010000 | 11100 | 5 | 5 |
| 5 | 44 | 1 | 01000100 | 10101000 | 8 | 8 |
| 6 | 46 | 1 | 01000110 | 10101100 | 8 | 8 |
| 7 | 2D | 1 | 00101101 | 10110000 | 8 | 8 |
| 8 | 31 | 1 | 00110001 | 10110100 | 8 | 8 |
| 9 | 2E | 1 | 00101110 | 10111000 | 8 | 8 |
| 10 | 33 | 1 | 00110011 | 10111100 | 8 | 8 |
| 11 | F3 | 1 | 11110011 | 110010000 | 9 | 9 |
| 12 | A0 | 1 | 10100000 | 110010100 | 9 | 9 |
| 13 | D0 | 1 | 11010000 | 110011000 | 9 | 9 |
| 14 | C4 | 1 | 11000100 | 110011100 | 9 | 9 |
| 15 | C6 | 1 | 11000110 | 110100000 | 9 | 9 |

**Journal of Computer System and Informatics (JoSYC)**
ISSN 2714-8912 (media online), ISSN 2714-7150 (media cetak)
Volume 3, No. 4, August 2022, Page 303−312
https://ejurnal.seminar-id.com/index.php/josyc
DOI 10.47065/josyc.v3i4.2109

| n | Hexa | Frek | ASCII Biner | *Codeword* | *Bit* | *Bit* × Frekuensi |
|---|---|---|---|---|---|---|
| 16 | 34 | 1 | 00110100 | 110100100 | 9 | 9 |
| 17 | 30 | 1 | 00110000 | 110101000 | 9 | 9 |
| | | | | **Total *Bit* × Frekuensi** | | **142 bit** |

Tahap selanjutnya adalah menyusun kembali *string bit* yang dihasilkan dari proses kompresi sesuai dengan urutan index nilai *hexadecimal* awal sebelum dikompresi (tabel 1), sehingga dihasilkan *string* biner berikut ini :

**Tabel 3.** *Codeword file* pdf Berdasarkan Algoritma YRC

| Index | Hexa | Biner | Code wods | Index | Hexa | Biner | Code words |
|---|---|---|---|---|---|---|---|
| 1 | 25 | 00100101 | 100 | 11 | F3 | 11110011 | 110010000 |
| 2 | 50 | 01010000 | 11100 | 12 | A0 | 10100000 | 110010100 |
| 3 | 44 | 01000100 | 10101000 | 13 | D0 | 11010000 | 110011000 |
| 4 | 46 | 01000110 | 10101100 | 14 | C4 | 11000100 | 110011100 |
| 5 | 2D | 00101101 | 10110000 | 15 | C6 | 11000110 | 110100000 |
| 6 | 31 | 00110001 | 10110100 | 16 | 0A | 00001010 | 10100 |
| 7 | 2E | 00101110 | 10111000 | 17 | 34 | 00110100 | 110100100 |
| 8 | 33 | 00110011 | 10111100 | 18 | 20 | 00100000 | 11000 |
| 9 | 0A | 00001010 | 10100 | 19 | 30 | 00110000 | 110101000 |
| 10 | 25 | 00100101 | 100 | 20 | 20 | 00100000 | 11000 |
| Total *bit codeword* sampel adalah 142 *bit* | | | | | | | |

100 11100 10101000 10101100 10110000 10110100 10111000 10111100 10100 100 110010000 110010100 110011000 110011100 110100000 10100 110100100 11000 110101000 11000

Total *bit* yang diperoleh ialah sampai pada tahap ini adalah 142 *bit*. Selanjutnya perlu dilakukan penambahan *bit* yaitu *padding bit* dan *flagging bit* karena 142 tidak habis dibagi 8 dan menyisakan 6 . Nyatakan hasil bagi tersebut dengan n. Apabila jumlah *bit* habis dibagi 8 atau kelipatan 8, maka tidak perlu dilakukan *padding* tetapi tetap harus menambahkan *flagging*. Rumus untuk menambahkan *padding* adalah 7-n+"1" dan untuk menambahkan *flagging* dapat menggunakan rumus 9-n.

142 *mod* 8 = 6 ; n =6

*Padding* = 7 – n + "1"

= 7 – 6 + "1" = **01** (tambahkan *bit* 0 sebanyak 1 *bit* dan gabungkan dengan *bit* 1)

*Flagging* = 9 – n

= 9 – 6 = 3 (dikonversi ke biner menjadi **00000011)**

*String bit* setelah penambahan *padding* dan *flagging* adalah :

100 11100 10101000 10101100 10110000 10110100 10111000 10111100 10100 100 110010000 110010100 110011000 110011100 110100000 10100 110100100 11000 110101000 11000 **01 00000011**

Total *bit* akhir hasil kompresi adalah 152 *bit*. *String* biner ini dikelompokkan menjadi 8 *bit* setiap kelompok seperti yang disajikan pada tabel 4.

**Tabel 4.** Karakter isi *file* pdf hasil kompresi

| No | Nilai Biner | Nilai Desimal | Karakter | No | Nilai Biner | Nilai Desimal | Karakter |
|---|---|---|---|---|---|---|---|
| 1 | 10011100 | 156 | œ | 11 | 00110011 | 51 | 3 |
| 2 | 10101000 | 168 | ¨ | 12 | 00011001 | 49 | 1 |
| 3 | 10101100 | 172 | ¬ | 13 | 11001101 | 205 | Í |
| 4 | 10110000 | 176 | ° | 14 | 00000101 | 5 | ENQ |
| 5 | 10110100 | 180 | ´ | 15 | 00110100 | 52 | 4 |
| 6 | 10111000 | 184 | ¸ | 16 | 10011000 | 152 | ˜ |
| 7 | 10111100 | 188 | ¼ | 17 | 11010100 | 212 | Ô |
| 8 | 10100100 | 164 | ¤ | 18 | 01100001 | 193 | Á |
| 9 | 11001000 | 200 | È | 19 | 00000011 | 3 | ETX |
| 10 | 01100101 | 101 | e | | | | |

Masing-masing kelompok biner pada tabel 4 di atas dikonversi menjadi karakter kemudian disimpan menjadi sebuah *file* pdf terkompresi. Proses dekompresi dilakukan dengan mengkonversi nilai *file* pdf menjadi biner, kemudian dilakukan proses pengurangan *string bit* dengan menghilangkan *padding* dan *flagging*. Langkah seterusnya adalah pencocokan *string* biner dengan *code yamamoto's recursive code* sehingga ditemukan nilai-nilai *file* pdf awal. Proses pengurangan *bit padding* dan *flagging* dilakukan dengan cara nyatakan hasil pembacaan dengan n, selanjutnya gunakan rumus 7 + n untuk mengembalikan *string bit* ke bentuk semula.

**Journal of Computer System and Informatics (JoSYC)**
ISSN 2714-8912 (media online), ISSN 2714-7150 (media cetak)
Volume 3, No. 4, August 2022, Page 303−312
https://ejurnal.seminar-id.com/index.php/josyc
DOI 10.47065/josyc.v3i4.2109

100111001010100010101100101100001011010010111000101111001010010011001000011001010011001100
011001110011010000010100110100100110001101010001100001**00000011**

8 *bit* terakhir = 00000011 = 3 = n, sehingga 7 + n = 7 + 3 = 10
Hilangkan dari *string bit* sebanyak 10 bit terakhir, menjadi :

100111001010100010101100101100001011010010111000101111001010010011001000011001010011001100
011001110011010000010100110100100110001101010001100001000

Jumlah *string bit* sampai pada proses ini adalah 142 seperti semula. Langkah selanjutnya adalah pembacaan dan pencocokkan *string bit* berdasarkan *code yamamoto's recursive code* agar diperoleh nilai hexa dari *file* pdf asli. Pembacaan *string bit* dilakukan dari indeks terkecil sampai indeks terakhir dengan terus menambahkan nilai pada indeks sebelumnya.

Indeks ke 1 adalah 1, tidak terdapat pada nilai *codeword* tabel 3. Indeks ke 2 adalah 10, tidak terdapat pada *code word* tabel 3. Indeks ke 3 adalah 100 terdapat pada *codeword* tabel 3 dan didapat bahwa biner 100 adalah nilai *hexadecimal* 25. Proses yang sama dilakukan untuk mendapatkan nilai *hexadecimal* selanjutnya, sehingga hasil keseluruhan proses dekompresi disajikan pada tabel 5 berikut ini.

**Tabel 5.** Nilai hasil dekompresi *file* pdf berdasarkan algoritma YRC

| Index | *Codeword* | Hexa | Index | *Codeword* | Hexa |
|---|---|---|---|---|---|
| 1 | 100 | 25 | 11 | 110010000 | F3 |
| 2 | 11100 | 50 | 12 | 110010100 | A0 |
| 3 | 10101000 | 44 | 13 | 110011000 | D0 |
| 4 | 10101100 | 46 | 14 | 110011100 | C4 |
| 5 | 10110000 | 2D | 15 | 110100000 | C6 |
| 6 | 10110100 | 31 | 16 | 10100 | 0A |
| 7 | 10111000 | 2E | 17 | 110100100 | 34 |
| 8 | 10111100 | 33 | 18 | 11000 | 20 |
| 9 | 10100 | 0A | 19 | 110101000 | 30 |
| 10 | 100 | 25 | 20 | 11000 | 20 |
| Total *bit codeword* = 160 *bit* | | | | | |

Nilai-nilai hexa pada tabel 4 di atas, akan dikonversikan kembali menjadi nilai-nilai *file* pdf, sehingga dihasilkan *file* pdf awal.

### 3.3 Penerapan Algoritma Fixed Length Binary Encoding

Kompresi data berdasarkan algoritma *Fixed Length Binary Encoding* (FLBE) diawali dengan proses penyusunan frekuensi kemunculan dari karakter data yang akan dikompresi, kemudian masing-masing karakter pada susunan tersebut diberi nilai biner mulai dari 0 hingga biner ke-n untuk mendapatkan kode FLBE. Kode yang diperoleh akan disusun kembali sesuai dengan posisi karakter pada *string*. Gabungan dari kode FLBE yang telah disusun ini merupakan hasil dari kompresi. *File* PDF yang digunakan pada contoh kasus ini adalah *file* PDF yang digunakan pada proses kompresi algoritma YRC (file pdf yang sama). Nilai *file* pdf yang digunakan sebagai sampel pada penerapan kompresi algoritma FLBE adalah *file* pdf yang digunakan sebagai sampel pada algorima YRC.

**Tabel 6.** Pengurutan frekuensi dan *codeword* FLBE dari nilai pdf

| No | Hexa PDF | Frek | ASCII Biner | *Codeword* | *Bit* | *Bit* × Frekuensi |
|---|---|---|---|---|---|---|
| 1 | 25 | 2 | 00100101 | 00000 | 5 | 10 |
| 2 | 0A | 2 | 00001010 | 00001 | 5 | 10 |
| 3 | 20 | 2 | 00100000 | 00010 | 5 | 10 |
| 4 | 50 | 1 | 01010000 | 00011 | 5 | 5 |
| 5 | 44 | 1 | 01000100 | 00100 | 5 | 5 |
| 6 | 46 | 1 | 01000110 | 00101 | 5 | 5 |
| 7 | 2D | 1 | 00101101 | 00110 | 5 | 5 |
| 8 | 31 | 1 | 00110001 | 00111 | 5 | 5 |
| 9 | 2E | 1 | 00101110 | 01000 | 5 | 5 |
| 10 | 33 | 1 | 00110011 | 01001 | 5 | 5 |
| 11 | F3 | 1 | 11110011 | 01010 | 5 | 5 |
| 12 | A0 | 1 | 10100000 | 01011 | 5 | 5 |
| 13 | D0 | 1 | 11010000 | 01100 | 5 | 5 |
| 14 | C4 | 1 | 11000100 | 01101 | 5 | 5 |
| 15 | C6 | 1 | 11000110 | 01110 | 5 | 5 |
| 16 | 34 | 1 | 00110100 | 01111 | 5 | 5 |
| 17 | 30 | 1 | 00110000 | 10000 | 5 | 5 |

**Journal of Computer System and Informatics (JoSYC)**
ISSN 2714-8912 (media online), ISSN 2714-7150 (media cetak)
Volume 3, No. 4, August 2022, Page 303−312
https://ejurnal.seminar-id.com/index.php/josyc
DOI 10.47065/josyc.v3i4.2109

| No | Hexa PDF | Frek | ASCII Biner | *Codeword* | *Bit* | *Bit* × Frekuensi |
|---|---|---|---|---|---|---|
| | Total *Bit* Awal | | 160 *bit* | Total *Bit* x Frekuensi | | 100 *bit* |

Tahap selanjutnya adalah menyusun kembali *string bit* yang telah rirubah dari *codeword* sesuai dengan urutan nilai *hexadecimal* awal, seperti berikut ini:

**Tabel 7.** *Codeword File* Pdf Berdasarkan Algoritma FLBE

| Index | Hexa | Biner | *Codeword* | Index | Hexa | Biner | *Codeword* |
|---|---|---|---|---|---|---|---|
| 1 | 25 | 00100101 | 00000 | 11 | F3 | 11110011 | 01010 |
| 2 | 50 | 01010000 | 00011 | 12 | A0 | 10100000 | 01011 |
| 3 | 44 | 01000100 | 00100 | 13 | D0 | 11010000 | 01100 |
| 4 | 46 | 01000110 | 00101 | 14 | C4 | 11000100 | 01101 |
| 5 | 2D | 00101101 | 00110 | 15 | C6 | 11000110 | 01110 |
| 6 | 31 | 00110001 | 00111 | 16 | 0A | 00001010 | 00001 |
| 7 | 2E | 00101110 | 01000 | 17 | 34 | 00110100 | 01111 |
| 8 | 33 | 00110011 | 01001 | 18 | 20 | 00100000 | 00010 |
| 9 | 0A | 00001010 | 00001 | 19 | 30 | 00110000 | 10000 |
| 10 | 25 | 00100101 | 00000 | 20 | 20 | 00100000 | 00010 |
| Total *bit codeword* sampel adalah 100 *bit* | | | | | | | |

Sehingga *string bit* yang dihasilkan dari proses kompresi algoritma FLBE berjumlah 100 *bit* :

00000 00011 00100 00101 00110 00111 01000 01001 00001 00000 01010 01011 01100 01101 01110 00001 01111 00010 10000 00010

Selanjutnya perlu dilakukan penambahan *bit* yaitu *padding bit* dan *flagging bit* karena 100 tidak habis dibagi 8 dan menyisakan 4 dan nyatakan hasil bagi tersebut dengan n. Apabila jumlah *bit* habis dibagi 8 atau kelipatan 8, maka tidak perlu dilakukan *padding* tetapi tetap harus menambahkan *flagging*. Formulasi *padding* adalah 7-n+"1" dan untuk menambahkan *flagging* dapat menggunakan rumus 9-n.
100 mod 8 = 4 ; n = 4
*Padding* = 7 – n + "1", sehingg 7 – 4 + "1" = **0001** (ditambah 3 *bit* 0 dan gabungkan dengan *bit* 1)
*Flagging* = 9 – n, sehingga 9 – 4 = 5 = **00000101** (representasi 8 *bit* dari nilai biner bilangan 5)
Jadi, penambahan *padding* dan *flagging* seperti berikut :

000000001100100001010011000111010000100100001000000101001011011000110101110000010111100010 100000001**0001**00000101, dengan total *bit* yaitu 112 *bit*.

Langkah berikutnya membagi *string bit* menjadi per 8 *bit*, lalu merubahnya menjadi karakter

**Tabel 8.** Nilai hasil kompresi *file* pdf berdasarkan algoritma FLBE

| No | Nilai Biner | Nilai Desimal | Karakter | No | Nilai Biner | Nilai Desimal | Karakter |
|---|---|---|---|---|---|---|---|
| 1 | 00000000 | 0 | NUL | 8 | 10110110 | 182 | ¶ |
| 2 | 11001000 | 200 | È | 9 | 00110101 | 53 | 5 |
| 3 | 01010011 | 83 | S | 10 | 11000001 | 193 | Á |
| 4 | 00011101 | 29 | GS | 11 | 01111000 | 120 | x |
| 5 | 00001001 | 9 | HT | 12 | 10100000 | 160 | |
| 6 | 00001000 | 8 | BS | 13 | 00100001 | 33 | ! |
| 7 | 00010100 | 20 | DC4 | 14 | 00000101 | 5 | ENQ |

Nilai-nilai hexa pada tabel 8 di atas, akan dikonversikan kembali menjadi nilai-nilai *file* pdf, sehingga dihasilkan *file* pdf awal.

Proses dekompresi berdasarkan algoritma FLBE diawali dengan membaca *string* biner dari *file* pdf terkompresi sebelumnya :
000000001100100001010011000111010000100100001000000101001011011000110101110000010111100010 10000000100001000010
Langkah selanjutnya adalah melakukan proses penghilangan *bit padding* dan *flagging* dengan cara yang sama seperti yang dilakukan pada algoritma YRC.
000000001100100001010011000111010000100100001000000101001011011000110101110000010111100010 10000000100001**00000101**
8 *bit* terakhir = 00000111 = 5 = n, sehingga 7 + n = 7 + 5 = 12
Hilangkan dari *string bit* sebanyak 12 *bit* terakhir, menjadi :
000000001100100001010011000111010000100100001000000101001011011000110101110000010111100010 1000000010. Berdasarkan perhitungan di atas, *string bit* berjumlah 100 seperti semula.

**Journal of Computer System and Informatics (JoSYC)**
ISSN 2714-8912 (media online), ISSN 2714-7150 (media cetak)
Volume 3, No. 4, August 2022, Page 303−312
https://ejurnal.seminar-id.com/index.php/josyc
DOI 10.47065/josyc.v3i4.2109

Selanjutnya, dilakukan pembacaan *string bit* awal dengan menggantikan kode pada *string bit* berdasarkan kode *Fixed Length Binary Encoding* (tabel 7) untuk memperoleh isi *file* pdf asli. Pembacaan string bit dilakukan dari indeks terkecil sampai indeks terakhir dengan terus menerus menambahkan nilai pada indeks sebelumnya yang terdapat pada kode *Fixed Length Binary Encoding* (FLBE).

**Tabel 9.** Nilai hasil dekompresi *file* pdf berdasarkan algoritma FLBE

| Index | *Codeword* | Hexa | Index | *Codeword* | Hexa |
|---|---|---|---|---|---|
| 1 | 00000 | 25 | 11 | 01010 | F3 |
| 2 | 00011 | 50 | 12 | 01011 | A0 |
| 3 | 00100 | 44 | 13 | 01100 | D0 |
| 4 | 00101 | 46 | 14 | 01101 | C4 |
| 5 | 00110 | 2D | 15 | 01110 | C6 |
| 6 | 00111 | 31 | 16 | 00001 | 0A |
| 7 | 01000 | 2E | 17 | 01111 | 34 |
| 8 | 01001 | 33 | 18 | 00010 | 20 |
| 9 | 00001 | 0A | 19 | 10000 | 30 |
| 10 | 00000 | 25 | 20 | 00010 | 20 |
| Total *bit codeword* = 160 bit | | | | | |

### 3.4 Pengukuran Kualitas Hasil Kompresi

Pengukuran kualitas hasil kompresi dilakukan dengan menghitung nilai parameter *ratio of compression*, *compression ratio* dan *space saving* dari masing-masing *file* pdf.

a. Algoritma *Yamamoto's Recursive Code*

1. *Ratio of Compression* ($R_C$)

   $Rc = \frac{Ukuran\ bit\ data\ sebelum\ dikompresi}{Ukuran\ bit\ data\ setelah\ dikompresi} = \frac{160}{152} = 1{,}05$

2. *Compression Ratio* ($C_R$)

   $C_R = \frac{Ukuran\ bit\ data\ setelah\ dikompresi}{Ukuran\ bit\ data\ sebelum\ dikompresi} \times 100\ \% = \frac{152}{160} \times 100\ \% = 0{,}95\ \%$

3. *Space Saving* ($S_S$)

   $S_S = 100\ \% - C_R = 100\ \% - 0{,}95\ \% = 0{,}05\ \%$

b. Algoritma *Fixed Length Binary Encoding*

1. *Ratio of Compression* ($R_C$)

   $Rc = \frac{Ukuran\ bit\ data\ sebelum\ dikompresi}{Ukuran\ bit\ data\ setelah\ dikompresi} = \frac{160}{112} = 1{,}42$

2. *Compression Ratio* ($C_R$)

   $C_R = \frac{Ukuran\ bit\ data\ setelah\ dikompresi}{Ukuran\ bit\ data\ sebelum\ dikompresi} \times 100\ \% = \frac{112}{160} \times 100\ \% = 0{,}7\ \%$

3. *Space Saving* ($S_S$)

   $S_S = 100\ \% - C_R = 100\ \% - 0{,}7\ \% = 0{,}3\ \%$

### 3.5 Penerapan Metode Exponential

Proses perbandingan kedua algoritma dalam penelitian ini dilakukan dengan memanfaatkan nilai parameter kualitas kompresi *file* pdf berdasarkan masing-masing algoritma. Algoritma *yamamoto's recursive code* dan algoritma *fixed length binary encoding* menjadi alternatif dalam penerapan metode *exponential*, sedangkan parameter kualitas hasil kompresi menjadi kriteria. Pemberian nilai bobot didasarkan pada tingkat kepentingan masing-masing kriteria dimana kriteria *ratio of compression* dan *compression ratio* masing-masing diberi bobot yang sama yaitu 0,25, sedangkan kriteria *space saving* diberi bobot 0,50. Proses perhitungannya sebagai berikut :

Algoritma YRC $= (1{,}05)^{0{,}25} + (0{,}95)^{0{,}25} + (0{,}05)^{0{,}50}$
$= 1{,}01 + 0{,}98 + 0{,}22 = 2{,}22\ \%$

Algoritma FLBE $= (1{,}42)^{0{,}25} + (0{,}7)^{0{,}25} + (0{,}3)^{0{,}50}$
$= 1{,}09 + 0{,}91 + 0{,}54 = 2{,}55\ \%$

Berdasarkan hasil perhitungan dan perbandingan algoritma di atas, maka dapat disimpulkan bahwa algoritma yang lebih baik untuk mengkompresi *file* pdf berdasarkan nilai parameter kualitas kompreasi adalah algoritma *Fixed Length Binary Encoding* (FLBE) dengan nilai MPE sebesar 2,55 karena memiliki hasil

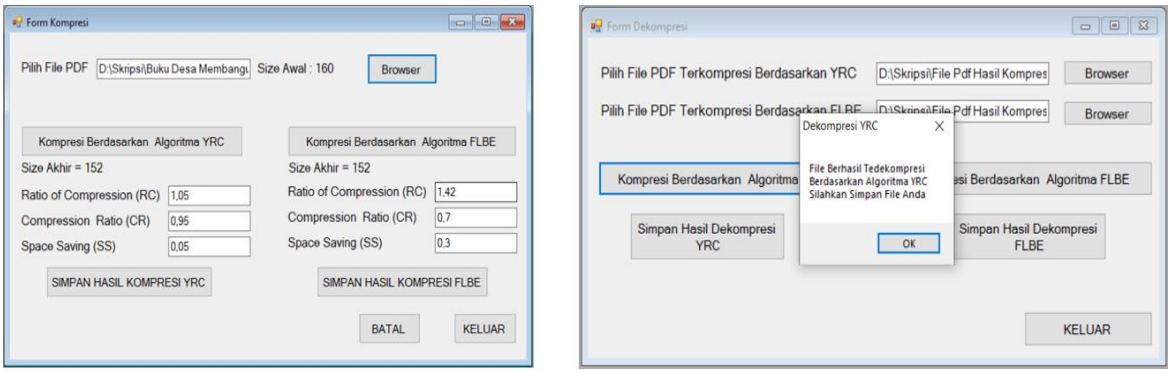
**Journal of Computer System and Informatics (JoSYC)**
ISSN 2714-8912 (media online), ISSN 2714-7150 (media cetak)
Volume 3, No. 4, August 2022, Page 303−312
https://ejurnal.seminar-id.com/index.php/josyc
DOI 10.47065/josyc.v3i4.2109

perhitungan masing-masing parameter kualitas kompresi yang lebih tinggi bila dibandingkan dengan algoritma *Yamamoto's Recursive Code*.

### 3.6 Hasil Implementasi Sistem

Implementasi dalam penelitian ini terdiri dari dua bagian yaitu, implementasi proses kompresi dan dekompresi *file* pdf serta implementasi proses perbandingan dari sistem yang dibangun.

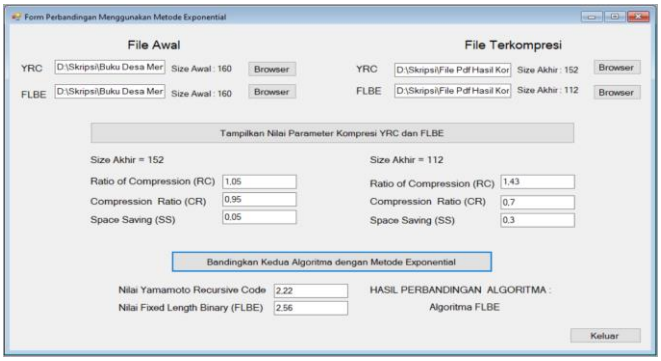
**a. Implementasi *Form* Kompresi dan Dekompresi**




**Gambar 4.** Implementasi *Form* Kompresi dan Form Dekompresi

Berdasarkan hasil implementasi yang dilakukan, disimpulkan bahwa proses kompresi dan dekompresi telah berjalan dengan baik dan menyajikan hasil yang sesuai dengan perhitungan manual.

**b. Implementasi *Form* Perbandingan**



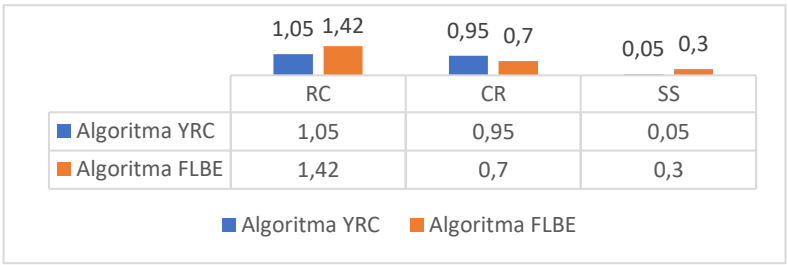

**Gambar 6.** Implementasi *Form* Perbandingan

Berdasarkan hasil implementasi yang dilakukan, disimpulkan bahwa proses perbandingan kedua algoritma telah berjalan dengan baik dan dapat menyajikan nilai perbandingan yang sesuai dengan perhitungan manual.

### 3.7 Hasil Pengujian

Pengujian dalam penelitian ini dilakukan terhadap tingkat kualitas hasil kompresi *file* pdf dari kedua algoritma serta pengujian terhadap nilai perbandingan kedua metode.

**a. Pengujian Nilai Parameter Kualitas Hasil Kompresi**



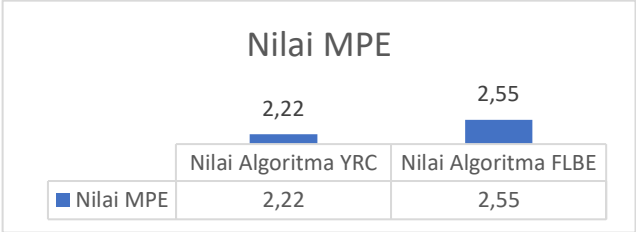

**Gambar 7.** Grafik Pengujian Nilai Parameter Kualitas Hasil Kompresi *File* pdf

Berdasarkan grafik di atas terlihat bahwa nilai parameter kualitas kompresi RC dan nilai parameter kualitas kompresi SS dari algoritma *Fixed Length Binary Encoding* memiliki nilai yang lebih besar dibandingkan dengan nilai parameter kualitas kompresi RC dan nilai parameter kualitas kompresi SS dari algoritma *Yamamoto's*

**Journal of Computer System and Informatics (JoSYC)**
ISSN 2714-8912 (media online), ISSN 2714-7150 (media cetak)
Volume 3, No. 4, August 2022, Page 303−312
https://ejurnal.seminar-id.com/index.php/josyc
DOI 10.47065/josyc.v3i4.2109

*Recursive Code*. Sedangkan nilai CR dari algoritma *Yamamoto's Recursive Code* lebih besar dari nilai CR algoritma *Fixed Length Binary Encoding*. Berdasarkan nilai tersebut dapat disimpulkan bahwa algoritma FLBE menghasilkan kualitas kompresi yang lebih baik khusunya pada kompresi *file* pdf.

**b. Hasil Perbandingan Kinerja Algoritma Berdasarkan Metode *Exponential***




**Gambar 8.** Grafik Perbandingan Kinerja Algoritma

Berdasarkan grafik pada gambar 8 di atas, terlihat bahawa setelah kedua algoritma dibandingkan berdasarkan Metode Perbandingan *Exponential* (MPE) diperoleh bahwa nilai *exponential* dari algoritma FLBE memiliki nilai yang tertinggi yaitu 2,55, sehingga dapat disimpulkan bahwa algoritma ini memiliki kinerja yang lebih baik untuk menghasilkan kualitas kompresi file pdf.

## 4. KESIMPULAN

Berdasarkan hasil perbandingan kinerja antara algoritma *Yamamoto's Recursive Code* dengan algoritma *Fixed Length Binary Encoding* (FLBE) yang dibandingkan berdasarkan Metode Perbandingan *Exponential* (MPE) melalui parameter kualitas hasil kompresi, disimpulkan bahwa algoritma *Fixed Length Binary Encoding* (FLBE) memiliki kinerja yang lebih baik untuk menghasilkan kualitas kompresi. Hal ini dapat diketahui dari nilai masing-masing parameter pengukuran kualitas kompresi (grafik pada gambar 7). Nilai-nilai parameter kualitas kompresi dari algoritma FLBE tetap memiliki nilai yang lebih tinggi dibandingkan dengan nilai parameter dari algoritma *yamamoto's recrusive code*. Hal ini juga diperkuat oleh nilai hasil perbandingan yang dilakukan menggunakan metode MPE (grafik pada gambar 8), dimana total nilai algoritma FLBE yaitu 2,55 %, sedangkan hasil dari algoritma *yamamoto's recrusive code* jauh lebih kecil yaitu 2,22%.

## REFERENCES

[1] N. Aftikasyah, "Penerapan Algoritma Yamamoto ' s Recursive Code Untuk Mengkompresi File Dokumen," vol. 5, pp. 255–263, 2022, doi: 10.30865/komik.v5i1.3716.

[2] D. Pratiwi and T. Zebua, "Analisis Perbandingan Kinerja Algoritma Fixed Length Binary Encoding Dan Algoritma Elias Gamma Code Dalam Kompresi File Teks," *KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer)*, vol. 3, no. 1, pp. 424–430, 2019, doi: 10.30865/komik.v3i1.1623.

[3] S. B. Ginting *et al.*, "Perbandingan Algoritma Yamamoto ' s Recursive Code Dan Additive Code Dalam Kompresi File Video," vol. 5, 2021, doi: 10.30865/komik.v5i1.3819.

[4] W. T. W. Simanjuntak, "Analisa Perbandingan Algortima Prediction By Partial Matching Dengan Sequitur Pada Kompresi File Teks," vol. 5, pp. 221–227, 2021, doi: 10.30865/komik.v5i1.3675.

[5] M. A. Latif, S. D. Nasution, and P. Pristiwanto, "Analisa Perbandingan Algoritma Rice Codes Dengan Algoritma Goldbach Codes Pada Kompresi File Text Menggunakan Metode Exponential," *Maj. Ilm. INTI (Informasi dan Teknol. Ilmiah)*, vol. 13, no. 1, pp. 28–33, 2018, [Online]. Available: http://ejurnal.stmik-budidarma.ac.id/index.php/inti/article/view/639.

[6] R. Y. TANJUNG, "Perancangan aplikasi kompresi file dokumen menggunakan algoritma additive code," vol. 8, no. 4, pp. 108–113, 2020, doi: 10.30865/jurikom.v8i4.3593.

[7] B. Ramadhana, "Implementasi Kombinasi Algoritma Fibonacci Codes Dan Levenstein Codes Untuk Kompresi File Pdf," vol. 8, no. 2, pp. 67–71, 2021.

[8] M. Apriyanto and H. Hutrianto, "Analisa Penerapan Algortima Goldbach Codes Dan Metode Shannon-Fano Pada Kompresi File Teks," *Bina Darma Conf. ...*, pp. 207–218, 2020, [Online]. Available: https://conference.binadarma.ac.id/index.php/BDCCS/article/download/1714/771.

[9] D. Muliadi, "Perbandingan Algoritma Yamamoto Recursive Dan Punctured Elias Code Dalam Kompresi File Teks," pp. 7–37, 2019.

[10] R. D. Pratiwi, S. D. Nasution, and F. Fadlina, "Perancangan Aplikasi Kompresi File Teks Dengan Menerapkan Algortima Fixed Length Binary Encoding (Flbe)," *J. Media Inform. Budidarma*, vol. 2, no. 1, pp. 10–14, 2018, doi: 10.30865/mib.v2i1.813.

[11] I. Lestari, "Analisa Perbadingan Algoritma Goldbach Codes Dengan Algoritma Sequitur Pada Kompresi File Text Menggunakan Metode …," *Pelita Inform. Inf. dan ...*, vol. 8, pp. 15–18, 2019, [Online]. Available: https://www.ejurnal.stmik-budidarma.ac.id/index.php/pelita/article/view/1520.

[12] S. Nainggolan, "Analisa Perbandingan Algoritma Goldbach Codes Dengan Algoritma Dynamic Markov Compression (DMC) Pada Kompresi File Teks Menggunakan Metode Eksponensial," *Maj. Ilm. INTI*, vol. 6, no. 3, pp. 395–399, 2019.