# Analisis Kombinasi Algoritma Knapsack dan Run Length Enconding (RLE) pada File Teks

*Shela Dian Yunita[1], Hery Sunandar[2]*

*[1,2] STMIK Budi Darma Medan Jl. Sisingamangaraja No.338 Simpang Limun Medan, Indonesia*

ARTICLE INFORMATION

Received: February 03,2019
Revised: March 13,2019
Available online: April 03,2019

KEYWORDS

Kriptografi, Knapsack, RLE

CORRESPONDENCE

Phone: +6281397863774

E-mail: selladianyunita@gmail.com

**A B S T R A K**

Algoritma Knapsack merupakan bagian dari kriptografi asimetri yang mana kunci enkripsinya berbeda dengan kunci dekripsinya. Di samping masalah keamanan file teks, masalah ukuran dari sebuah file teks juga menjadi pertimbangan. File teks yang berukuran besar dapat dimampatkan dengan melakukan proses kompresi. Algoritma Run Length Encoding (RLE) merupakan algoritma yang mengecilkan ukuran file teks, apabila teks tersebut mengalami banyak perulangan karakter. Kombinasi algoritma Knapsack dan RLE dapat menjamin file Teks tidak dapat dilihat oleh pengguna yang tidak berhak dan dapat menjamin file teks dapat disimpan dalam media file yang berkapasitas rendah. Pada penelitian ini, penulis membuat program kombinasi algortitma knapsack dan RLE pada file teks. Pada algoritma Knapsack akan terjadi penambahan ukuran file teks, hal ini dapat dilihat pada contoh kasus yang mana ukuran plainteks (pesan asli) adalah 9 bytes, kemudian setelah dilakukan proses enkripsi ukuran file teks menjadi 7 bytes. Karena itu pengunaan kombinasi enkripsi dan kompresi data lebih baik karena file menjadi lebih kecil dibandingkan kombinasi kompresi dan enkripsi data. Plainteks yang memiliki banyak perulangan karakter akan terkompresi dengan baik.

## 1. PENDAHULUAN

Kemajuan teknologi dewasa ini menyebabkan saling ketergantungan antara komputer dan telekomunikasi semakin besar. Jaringan-jaringan komputer mempunyai andil besar untuk pengadaan sumber-sumber informasi. Dengan adanya jaringan-jaringan tersebut memudahkan orang memperoleh data serta informasi dimanapun secara terbuka. Hal ini membuka peluang bagi orang-orang yang tidak bertanggung jawab melakukan kecurangan-kecurangan. Seperti memanipulasi data, menyadap data, dan lain sebagainya pada data yang dianggap penting dan sangat pribadi.

Keamanan data diperlukan untuk mengatasi kecurangan tersebut, salah satu penelitian yang membahas tentang pengamanan data adalah Aplikasi Enkripsi dan Dekripsi Dengan Algoritma Knapsack Public Block[1], [2]. Pada penelitian ini membahas tentang pemanfaatan VB.Net sebagai salah satu bahasa pemrograman berorientasi objek untuk memperoleh rancangan yang lebih menarik, dan mendukung pembuatan aplikasi keamanan data untuk meningkatkan keotentikan data. Aplikasi enkripsi dekripsi dengan algoritma knapsack public block diharapkan dapat membantu siapapun yang ingin mengamankan data yang berupa file teks[3]. Selain keamanan data yang perlu diperhatikan juga adalah kecepatan dalam pengiriman data tersebut. Kecepatan pengiriman ini tergantung dari ukuran informasi tersebut. Salah satu solusi masalah tersebut adalah dengan melakukan pemampatan (kompresi) data sebelum melakukan pengiriman dan kemudian penerima akan merekonstruksinya kembali menjadi data aslinya (dekompresi) .

Kompresi data adalah istilah umum untuk berbagai algoritma dan program yang dikembangkan untuk mengatasi masalah pemampatan data. Tujuan dari data yang kompresi untuk mengurangi redundansi dalam data yang tersimpan atau data ditransmisikan, sehingga meningkatkan efektif data kepadatan . Salah satu penelitian yang membahas tentang masalah kompresi data yaituAnalisis Perbandingan Algoritma Kompresi Lempel Ziv Welch, Arithmetic Coding, Dan Run- Length Encoding Pada File Teks oleh Telaumbanua ,Plipus . Dalam penelitian ini, dibandingkan tiga algoritma kompresi data yaitu Lempel Ziv Welch (LZW), Arithmetic Coding, dan Run-Length Encoding(RLE), dimana algoritma RLE dibantu oleh algoritma BurrowsWheeler Transform (BWT) untuk memaksimalkan kinerjanya. Algoritma-algoritma tersebut dipilih karena semuanya dalam kategori algoritma lossless, dan mewakili masing-masing teknik pengkodean. Algoritma yang memiliki perfomansi terbaik berdasarkan rata-rata rasio dan waktu proses kompresi/dekompresi adalah algoritma LZW. Algortima RLE+BWT sangat efektif untuk file yang memilki banyak deretan karakter yang sama, namun buruk untuk file yang isinya tidak beraturan (acak). Sama halnya seperti algoritma RLE+BWT, algoritma LZW juga buruk untuk file yang isinya acak, namun efektif untuk file teks biasa, sedangkan algoritma Arithmetic Coding efektif untuk semua file uji baik dalam keadaan acak maupun tidak[4], [5].

## 2. LANDASAN TEORI

Kriptografi merupakan metode untuk mengirimkan pesan rahasia sehingga hanya penerima pesan yang dimaksud dapat menghapus, menyamarkan atau membaca pesan tersebut [1]. Kriptografi berasal dari kata bahasa Yunani yaitu kryptos yang berarti tersembunyi dan grapein yang berarti menulis. Pesan asli disebut plainteks dan pesan yang telah disandikan disebut cipherteks. Pesan yang telah dienkapsulasi dan dikirim disebut kriptogram. Proses mengubah plainteks menjadi cipherteks disebut enkripsi. Membalikkan proses cipherteks menjadi plainteks disebut dekripsi. Siapapun yang terlibat dalam kriptografi disebut kriptografer. Pada sisi lain, studi tentang teknik matematika karena berusaha untuk mengalahkan metode kriptografi disebut pembacaan sandi. Cryptanalysts adalah orang-orang berlatih pembacaan sandi[6].

### 2.2 Enkripsi dan Dekripsi

Proses menyandikan plainteks menjadi cipherteks disebut enkripsi *(encryption)* atau disebut juga *enciphering*. Sedangkan proses mengembalikan cipherteks menjadi plainteks disebut dekripsi *(decryption)* atau *dechipering*[1]. Enkripsi dan dekripsi dapat diterapkan pada pesan yang dikirim ataupun pesan yang disimpan. Istilah encryption of data in motion mengacu pada enkripsi data yang ditransmisikan melalui saluran komunikasi, sedangkan istilah decryption of data at rest mengacu pada enkripsi data yang ada didalam storage.

Enkripsi data yang sering digunakan adalah Fast Block dan Strem Cipher. Kedua teknik ini adalah contoh dari algoritma simetris. Kekurangan dari algoritma simetris adalah tingkat kesulitan dalam menentukan kunci rahasia (private key) untuk didistribusikan . Oleh karena itu kita harus mengetahui teknik dari algoritma kriptografi yang kita gunakan untuk membuat sebuah data enkripsi. Untuk mengembalikan plainteks yang telah dienkripsi, membutuhkan teknik dekripsi.
Dekripsi adalah teknik yang digunakan untuk menguraikan data rahasia atau pesan yang telah dienkripsi kembali kebentuk semula atau bentuk aslinya dan menjadi pesan yang dapat dibaca dan dimengerti maksudnya[7]. Adapun rumus untuk proses enkripsi dan dekripsi adalah sebagai berikut:
c = ek(m) )……1
dimana:
m adalah plainteks
e adalah fungsi penyamaran
k adalah kunci rahasia
c adalah cipherteks

dan untuk proses dekripsi dapat digunakan persamaan sebagai berikut:
m = dk(c)……2
dimana :
m adalah plainteks
d adalah fungsi dekripsi
k adalah kunci
c adalah cipherteks
kedua persamaan ini adalah teknik enkripsi dan dekripsi yang menggunakan kuncipada proses permutasinya. (Nigel, 2004).

### 2.3 Algoritma Knapsack

Algoritma Knapsack juga adalah algoritma kriptografi kunci-publik. Keamanan algoritma ini terletak pada sulitnya memecahkan persoalan knapsack (Knapsack Problem)[8]. Knapsack artinya karung/kantung. Karung mempunyai sekapasitas muat terbatas. Barang-barang dimasukkan ke dalam karung hanya sampai batas kapasitas maksimum karung saja[9]. Jenis-jenis Algoritma Knapsack adalah sebagai berikut:

1. 0/1 Knapsack Problem Setiap barang hanya terdiri satu unit dan boleh diambil atau tidak sama sekali
2. 0/n Knapsack Problem Setiap barang terdiri dari n buat unit dan boleh diambil atau tidak sama sekali
3. Bounded Knapsack Problem Setiap barang tersedia n buah unit dan jumlahnya terbatas
4. Unbounded Knapsack Problem Setiap barang tersediah lebih dari satu unit dan jumlahnya tidak terbatas
5. Fractional Knapsack Problem Barang boleh diambil dalam bentuk pecahan atau sebahagian. Contohnya gula, garam, tepung dan lain-lain.

Adapun permsalahan pada Algoritma Knapsack adalah Diberikan bobot knapsack adalah M. Diketahui n buah objek yang masing-masing bobotnya adalah w1, w2, …, wn. Tentukan nilai b sedemikian sehingga
M = b1w1 + b2w2 + … + bnwni (2.1).
yang dalam hal ini, bi bernilai 0 atau 1. Jika b = 1, berarti objek i dimasukkan kedalam knapsack, sebaliknya jika bii = 0, objek i tidak dimasukkan. Dalam teori algoritma, persoalan knapsack termasuk ke dalam kelompok NPcomplete. Persoalan yang termasuk NP-complete tidak dapat dipecahkan dalam orde waktu polynomial[10].

### 2.4 Kompresi

Kompresi ialah proses pengubahan sekumpulan data menjadi suatu bentuk kode untuk menghemat kebutuhan tempat penyimpanan dan waktu untuk transmisi data[11]. Saat ini terdapat berbagai tipe algoritma kompresi , antara lain: Huffman, IFO, LZHUF, LZ77 dan variannya (LZ78, LZW, GZIP), Dynamic Markov Compression (DMC), BlockSorting Lossless, Run-Length, Shannon-Fano, Arithmetic, PPM (Prediction by Partial Matching), Burrows-Wheeler Block Sorting, dan Half Byte. Berdasarkan tipe peta kode yang digunakan untuk mengubah pesan awal (isi file input) menjadi sekumpulan codeword, metode kompresi terbagi menjadi dua kelompok, yaitu :

1. Metode statik : menggunakan peta kode yang selalu sama. Metode ini membutuhkan dua fase (two-pass): fase pertama untuk menghitung probabilitas kemunculan tiap simbol/karakter dan menentukan peta kodenya, dan fase kedua untuk mengubah pesan menjadi kumpulan kode yang akan ditransmisikan. Contoh: algoritma Huffman statik.
2. Metode dinamik (adaptif) : menggunakan peta kode yang dapat berubah dari waktu ke waktu. Metode ini disebut adaptif karena peta kode mampu beradaptasi terhadap perubahan karakteristik isi file selama proses kompresi berlangsung. Metode ini bersifat onepass, karena hanya diperlukan satu kali pembacaan terhadap isi file. Contoh: algoritma LZW dan DMC.

Berdasarkan teknik pengkodean/pengubahan simbol yang digunakan, metode kompresi dapat dibagi ke dalam tiga kategori [11], yaitu:

1. Metode symbolwise : menghitung peluang kemunculan dari tiap symbol dalam file input, lalu mengkodekan satu simbol dalam satu waktu, dimana simbol yang lebih sering muncul diberi kode lebih pendek dibandingkan simbol yang lebih jarang muncul, contoh: algoritma Huffman.
2. Metode dictionary : menggantikan karakter/fragmen dalam file input dengan indeks lokasi dari karakter/fragmen tersebut dalam sebuah kamus (dictionary), contoh:algoritma LZW.
3. Metode predictive : menggunakan model finite-context atau finite-state untuk memprediksi distribusi probabilitas dari simbol-simbol selanjutnya; contoh: algoritma DMC.

Ada beberapa faktor yang sering menjadi pertimbangan dalam memilih suatu metode kompresi yang tepat, yaitu kecepatan kompresi, sumber daya yang dibutuhkan (memori, kecepatan PC), ukuran file hasil kompresi, besarnya redundansi, dan kompleksitas algoritma. Tidak ada metode kompresi yang paling efektif untuk semua jenis file. Dalam penelitian ini algoritma yang digunakan hanya Run-Length Encoding. Karena Algoritma tersebut cepat dan mudah untuk mengeksekusi .

### 2.5. Run Length Encoding (RLE)

Mengurangi ukuran karakter string yang berulang. Berulang ini biasa disebut Run. Biasanya RLE mengkodekan run dari simbol menjadi dua byte, jumlah dan symbol[11]. RLE dapat memampatkan semua jenis data terlepas dari isi informasi, tetapi isi dari( data yang akan dikompresi mempengaruhi rasio kompresi. RLE tidak dapat mencapai rasio kompresi yang tinggi dibandingkan dengan metode kompresi lainnya, tapi. mudah untuk menerapkan dan cepat untuk mengeksekusi. RLE didukung oleh sebagian besar format file bitmap seperti tiff, BMP, PCX.

Contoh: S= 11111111111111100000000000000000001111

Ini dapat direpresentasikan sebagai 1 sebanyak Lima Belas, 0 sebanyak sembilan belas, dan empat buat 1, yaitu (15,1), (19,0), (4,1). karena pengulangan jumlah maksimum adalah 19, yang dapat direpresentasikan dengan 5 bit, dapat dikodekan sebagai bit stream (01111,1), (10011,0), (00100,1) [1].

Contoh dari lain algoritma RLE:

1. KKKKKKK
   Adalah perulangan karakter “K” sebanyak 7 kali, ini dapat dikatakan sebagai run length karena mengulangi karakter yang sebanyak 7 kali
2. ABCDEFH
   Adalah contoh yang tidak akurat, karena tidak mengalami perulangan pada setiap karakternya. Ketujuh karakter diatas merupakan tujuh karakter yang berbeda-beda. Karakter ini tidak dapat dikatakan sebagai Run Length.
3. ABABBBC
   Dari karakter contoh tiga, terdapat perulangan karakter “B” sebanyak 3 kali, karakter ini sudah dpat dikatakan run length

Teknik dari algoritma run length encoding adalah untuk mempersingkat penulisan dari sebuah code, untuk penyelesaian dari contoh 1 adalah kita dapat menuliskan “KKKKKKK” menjadi (‘r’,’7’,’K’) untuk lebih singkat ditulis dengan “r7k” dengan ketentuan bahwa angka 7 didapatkan karena symbol karaker “k” mengalami perlangan sebanyak 7 kali.

Penyelesaian untuk contoh dua yaitu ABCDEFG, pada simbol ini tidak dapat perulangan karakter sehingga penulisannya dapat kita ubah menjadi (‘n’’7’,”ABCDEFG”) dan lebih singkatnya n7ABCDEFG.

## 3. HASIL DAN PEMBAHASAN

Sebelum dilakukan tahap perancangan sebuah sistem, perlu dilakukan analisis sistem yang akan dibangun. Analisis sistem merupakan istilah yang secara kolektif mendepenelitiankan fase-fase awal pengembangan sistem. Analisis sistem pada dasarnya merupakan tahapan yang ditujukan untuk menciptakan pemahaman yang menyeluruh terhadap sistem sehingga diperoleh gambaran tentang kebutuhan, cara kerja, dan alur data yang akan dikerjakan sistem. Hal ini akan membantu mempermudah dalam proses implementasi sistem. Sistem ini diharapkan dapat membantu menyelesaikan masalah dalam keamanan data, agar data tersebut tetap tidak dimodifikasi atau dibaca oleh orang yang tidak memiliki hak akses. Dalam sistem ini file teks akan dienkripsi menggunakan algoritma Knapsack. Selanjutnya untuk kompresi data teks, dilakukan dengan menggunakan algoritma RLE. Sehingga diharapkan file teks ini terjaga kemanannya serta ukuran file dapat lebih kecil dari sebelumnya. Yang menjadi masalah utama penelitian ini adalah bagaimana mengkombinasikan algoritma knapsack dan RLE pada file teks. Sehingga diharapkan file teks ini terjaga keamanannya serta ukuran file dapat lebih kecil dari sebelumnya.

### 3.1. Proses tahapan algoritma Knapsack dan RLE

Untuk menentukan barisan Kunci rahasia, harus memasukkan parameter sebagai contoh:
Parameter tersebut kemudian dimasukkan dalam rumus berikut :
w(3)=(distance)+w(1)+w(2);
w(4)=(distance+1)+w(1)+w(2)+w(3);
w(5)=(distance+2)+w(1)+w(2)+w(3)+w(4);
w(6)=(distance+3)+w(1)+w(2)+w(3)+w(4)+w(5);
w(7)=(distance)+w(1)+w(2)+w(3)+w(4)+w(5)+w(6);
dimana nilai awal itu adalah
w(1) = 2. Sehingga didapatkan :
w(3)= 4 + 2 +3 =9
w(4)= (4 + 1)+2 +3 +9 = 19
w(5)= (4 + 2) + 2 +3 +9 + 19 = 39
w(6)= (4 + 3) + 2 +3 +9 + 19 + 39 = 79
w(7)= 4 + 2 +3 +9 + 19 + 39 + 79 = 155
maka dari perhitungan tersebut didapatkan bahwa kunci rahasia adalah 2,3,9,19,39,79,155 Dibawah ini adalah tampilan source code dari proses Bangkit Kunci rahasia:
Untuk menentukan barisan Kunci publik , tentukan nilai Modulus m yaitu angka yang lebih besar dari jumlah semua elemen didalam barisan dan nilai n yaitu nilai yang tidak mempunyai persekutan terhadap m. Pada sistem ini nilai m dibangkitkan secara acak : m = 352 n = 5 dihitung menggunakan rumus:
Kp = w(Kr).n mod m
Keterangan: Kp = barisan Kunci Publik w(Kr) = nilai setiap barisan kunci rahasia Sehingga didapat: (2.5)mod 352 = 10
(3.5)mod 352 = 15
(9.5)mod 352 = 45
(19.5)mod 352 = 95
(39.5)mod 352 = 195
(79.5)mod 352 = 43
(155.5)mod 352 = 71
Barisan Kunci Publik adalah 10,15, 45, 95, 195, 43, 71. Dibawah ini adalah tampilan source code dari Bangkit Kunci Publik

### 3.2. Proses Kompresi dan Enkripsi

Sebagai contoh implementasi dalam kasus ini, misalnya untuk mengenkripsi plainteks ‘AAAAAA’. Maka kita akan mendapatkan : Kompresi menggunakaan RLE , plainteks “A” mengalami perulangan sebanyak 7 kali maka kompresi dari plainteks “AAAAAAA” adalah A7. Untuk mengenkripsi plainteks yang telah dikompresi yaitu A7:
Karakter ‘A’ dalam ASCII mempunyai nilai biner 1000001
Karakter ‘7’ dalam ASCII mempunyai nilai biner 0110111
Blok plainteks ke-1 : 1000001
Kunci publik : 10 15 45 95 195 43 71
Kriptogram : (1x10)+(0x15)+(0x45)+(0x95)+(0x195)+(0x43)+(1x71) = 81
Blok plainteks ke-2 : 0110111
Kunci publik : 10 15 45 95 195 43 71
Kriptogram :(0x10)+(1x15)+(1x45)+(0x95)+(1x195)+(1x43)+(1x71) = 369
Maka cipherteks dari plainteks A7 adalah 81, 369

### 3.3. Proses Enkripsi dan Kompresi

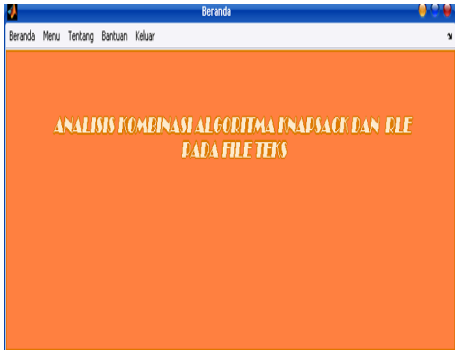
Proses awal dalam tahap ini adalah enkripsi data terlebih dahulu kemudian kompresi data. Kunci Publik, Kunci rahasia dan contoh plainteks sam dengan data diatas.
Plainteks = AAAAAA
Kunci Publik = 10 15 45 95 195 43 71
Kunci Rahasia = 2 3 9 19 39 79 155
Nilai m = 352
Nilai n = 5
Nilai n-1 = 141
Enkripsi plainteks :
Blok plainteks A : 1000001
Kunci publik : 10 15 45 95 195 43 71
Kriptogram : (1x10)+(0x15)+(0x45)+(0x95)+(0x195)+(0x43)+(1x71) = 81
Maka cipherteks adalah 81, 81, 81, 81, 81, 81, 81 Kompresi plainteks : 81 mengalami perulangan sebanyak tujuh kali maka kompresiny adalah 81 7
Untuk mengembalikan plainteks awal dilakukan dekompresi data terlebih dahulu kemudian dekripsi data.

### 3.5. Proses Dekripsi dan DeKompresi

Dekompresi nilai : 81 7
perulangannya 81 adalah objek yang berulang dan 7 adalah nilai perulangannya, maka diperoleh 81 sebanyak tujuh kali. Sehingga dapat diperoleh plainteks awal adalah "81 81 81 81 81 81 81". 81 81 81 81 81 81 81 bukan merupakan plainteks awal, untuk menentukannya diperlukan dekripsi data teks: 81. 141 mod 352 = 157 = 2 + 155 , berkoresponden dengan 1000001 yaitu A Maka diperoleh plainteks awal adalah "AAAAAAA"

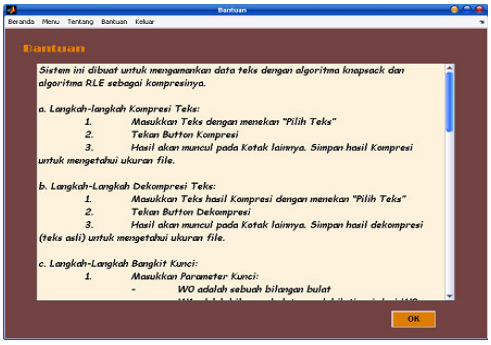
### 3.5. Implementasi

Tahap implementasi sistem merupakan tahap pembuatan perangkat lunak, tahap lanjut dari tahap perancangan sistem pengujian. Tahap yang dilakukan untuk menterjemahkan perancangan berdasarkan hasil analisis pengujian dalam bahasa yang dimengerti oleh komputer serta penerapan perangkat lunak pada keadaan yang sebenarnya.
Implementasi dari hasil tahapan analisis dan perancangan dapat dilihat dari tampilan antarmuka sistem sebagai berikut :



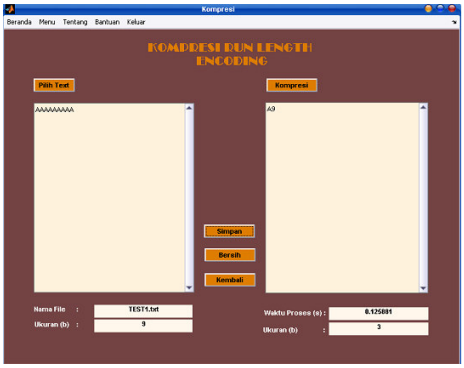

**Gambar 1 :** Tampilan Form Beranda

Form ini memiliki 8 form turunan, yaitu form Bangkit kunci, form Kompresi, form Dekompresi, form Enkripsi, form Dekripsi, form Kombinasi Algoritma Knapsack dan RLE, form Dekripsi dan Dekompresi, dan form Dekompresi dan Dekripsi.
Bantuan Form ini berisi panduan singkat dalam penggunaan software ini. Dengan adanya Form ini, diharapkan dapat membantu user yang kesulitan dalam menjalankan software ini. Tampilan Form Bantuan ditunjukan pada Gambar 2



**Gambar 2 :** Tampilan Form Bantuan

Form ini merupakan tempat user berinteraksi dengan sistem dalam melakukan kompresi teks dapat dilihat pada gambar 3

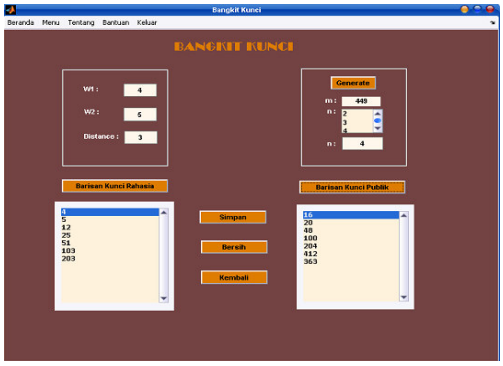



**Gambar 3 :** Tampilan pengujian Kompresi

Pada gambar 3, setelah menginputkan file teks, kemudian user menekan tombol Kompresi. Dan hasil dari kompresi akan muncul beserta waktu proses sedangkan untuk ukuran file akan muncul setelah user menekan tombol Simpan.
Form ini merupakan tempat user berinteraksi dengan sistem dalam melakukan dekompresi teks dapat dilihat pada gambar 4. Yaitu mengembalikan teks yang telah dikompresi menjadi teks asli.


**Gambar 4 :** Tampilan Pengujian Dekompresi

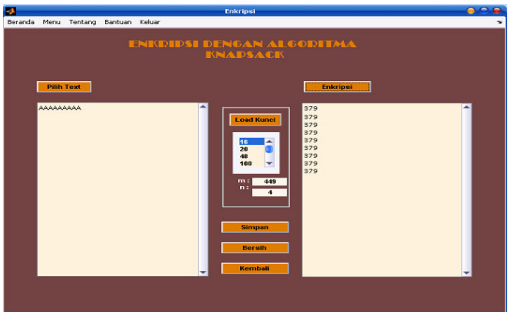
Pada gambar 4 pengujian dekompresi, user diwajibkan memasukkan file yang telah dikompresi yang bernama File Kompresi. Untuk melanjutkannya, user menekan tombol Dekompresi untuk membalikkan teks ke semula.
Form ini merupakan tempat user berinteraksi dengan sistem dalam melakukan pembangkitan kunci dengan algoritma Knapsack dapat dilihat pada gambar 5




**Gambar 5 :** Tampilan Pengujian Bangkit Kunci

Pada gambar diatas, sebelum melakukan pengujian enkripsi, sebaiknya user membangkitkan kunci terlebih dahulu dengan memasukkan parameter yang ada seperti, W1, W2, dan Distance . Setelah memasukkan parameter, user menekan tombol Barisan Kunci Rahasia untuk Kunci Rahasia. Dan untuk kunci publik, terlebih dahulu user menekan tombol Generate untuk membangkitkan nilai m secara acak dan menentukan nilai n sesuai pilihan yang telah disediakan, kemudian menekan tombol Barisan Kunci Publik.

Form ini merupakan tempat user berinteraksi dengan sistem dalam melakukan enkripsi teks dapat dilihat pada gambar 6. Yaitu mengubah plainteks menjadi cipherteks.



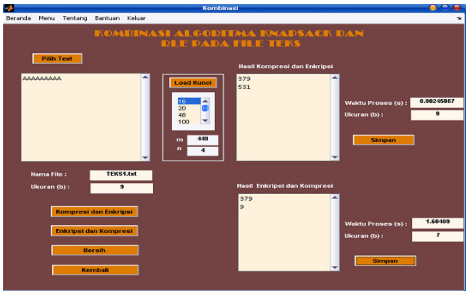

**Gambar 6 :** Tampilan Pengujian Enkripsi

Form ini merupakan tempat user berinteraksi dengan sistem dalam melakukan dekripsi teks dapat dilihat pada gambar 7. Yaitu mengembalikan cipherteks menjadi plainteks.


**Gambar 7 :** Tampilan Pengujian Dekripsi

Pada gambar di atas, user diwajibkan memasukkan file yang telah dienkripsi yang bernama File Enkripsi. Untuk melanjutkannya, user menekan load kunci untuk menampilkan kunci dekripsi dan tombol Dekripsi untuk mengembalikan teks ke semula.

Form ini merupakan tujuan dari penulisan penelitian ini. Yaitu mengkombinasikan algoritma Knapsack dan RLE pada file teks. Khusus pengujian Kombinasi Knapsack dan RLE dilakukan pengujian terhadap dua teks berbeda, TEKS1.txt berisi "AAAAAAAAA" dan TEKS2.txt berisi "ABABABABA". Pengujian terhadap TEKS1 dan TEKS2 dapat dilihat pada gambar 8




**Gambar 8 :** Tampilan Pengujian Kombinasi untuk TEKS1

Form ini merupakan tampilan untuk mengembalikan hasil Kombinasi Kompresi dan Enkripsi pada file teks. Pengujian Dekripsi dan Dekompresi juga dilakukan dengan dua teks diatas. EKS2.txt dapat dilihat pada gambar 9


**Gambar 9 :** Tampilan Pengujian Dekripsi dan Dekompresi untuk TEKS1

## 5. KESIMPULAN

Setelah melakukan pengujian, maka penulis dapat menarik kesimpulan sebagai berikut

1. Kompresi Teks dengan menggunakan algoritma Run Length Encoding dapat terkompresi dengan baik oleh sistem apabila dalam teks tersebut banyak perulangan huruf yang berurut. Sedang untuk teks yang sedikit perulangan atau tidak ada perulangan huruf sama sekali, kompresi menggunakan algoritma ini kurang terlaksana baik dikarenakan ukuran file bertambah besar dari ukuran semula.
2. Enkripsi teks dengan menggunakan algoritma Knapsack dapat mengamankan pesan dengan baik.
3. Kombinasi mendahulukan Enkripsi kemudian Kompresi teks lebih baik digunakan karena kombinasi keduanya berhasil terkompresi dengan baik dibandingkan mendahulukan Kompresi kemudian mengenkripsinya walaupun waktu untuk mengeksekusi plainteks lebih lama.
4. Kombinasi Algoritma Knapsack dan RLE pada file Teks tepat digunakan apabila dalam suatu plainteks (pesan asli) memiliki banyak perulangan karakter.

## DAFTAR PUSTAKA

[1] R. Munir, "Kriptografi," *Inform. Bandung*, 2006.
[2] H. D. M. H. Hutahaean, "Aplikasi Pembelajaran Kriptograpi berbasis Mobile menggunakan Computer Assisted Instruction," vol. 4, no. 1, pp. 2–5, 2019.
[3] R. Munir, "Algoritma Knapsack," pp. 0–18, 2004.
[4] W. Zarman and T. Pamungkas, "IMPLEMENTASI ALGORITMA KOMPRESI LZW PADA DATABASE SERVER," *J. Ilm. Komput. dan Inform.*, vol. 7, no. 1, 2013.
[5] N. J. Tuturoong, "PERBANDINGAN RASIO DAN KECEPATAN KOMPRESI MENGGUNAKAN ALGORITMA HUFFMAN, LZW DAN DMC," *TEKNO*, 2010. [Online]. Available: https://ejournal.unsrat.ac.id/index.php/tekno/article/view/4320. [Accessed: 22-Jan-2020].
[6] T. Limbong, "Pengujian Kriptografi Klasik Caesar Chipper Menggunakan Matlab," *no. Sept.*, vol. 2017, 2015.
[7] A. Fauzi, "Analisa Perancangan Aplikasi Penyandian Pesan Pada Email Menggunakan Algoritma Kriptografi Blowfish," *MEANS (Media Inf. Anal. dan Sist.*, vol. 1, no. 2, pp. 72–77, Dec. 2016.
[8] A. Ambarwari and N. Yanto, *Penerapan Algoritma Greedy Pada Permasalahan Knapsack Untuk Optimasi Pengangkutan Peti Kemas*. 2016.
[9] A. Kataria, "Algorithm for fractional knapsack problem," 2018. [Online]. Available: https://www.includehelp.com/algorithms/fractional-knapsack-problem.aspx. [Accessed: 14-Dec-2019].
[10] Paryati, "OPTIMASI STRATEGI ALGORITMA GREEDY UNTUK MENYELESAIKAN PERMASALAHAN KNAPSACK 0-1," *Semin. Nas. Inform.*, vol. 2009, no. semnasIF, pp. 101–110, 2009.
[11] D. Ariyus, "Kriptografi keamanan data dan komunikasi," *Yogyakarta Graha Ilmu*, 2006.