p-ISSN : 2406-7733
e-ISSN : 2597-9922

ANALISIS ALGORITMA KRIPTOGRAFI BACONS CIPHER DAN ALGORITMA KOMPRESI EVEN RODEH RODE UNTUK OPTIMASI KEAMANAN PESAN FILE TEKS

Daffa Zain Shahriza[1], Sayuti Rahman[2], Arif Budiman[3]
[1,2,3]Program Studi Teknik Informatika, Fakultas Teknik dan Komputer, Universitas Harapan Medan
Jl. HM Joni No 70 C, Kec. Medan Kota, Kota Medan, Sumatera Utara
E-mail: *daffashahriza01@gmail.com[1], sayutirahman@staff.uma.ac.id[2], ariefbudiman@unhar.ac.id[3]

Abstrak - Keamanan dan kerahasiaan sebuah informasi pada *file* teks merupakan hal yang sangat penting untuk dilakukan, terutama informasi sensitif atau pribadi yang hanya boleh diakses oleh pihak yang berhak saja. Selain aspek keamanan, maka hal yang perlu diperhatikan juga adalah tentang memori penyimpanan. Di era *modern* yang serba digital saat ini, pertukaran informasi dapat dilakukan secara nirkabel melalui media digital dimana saja dan kapan saja, hal ini mengharuskan pengguna untuk memiliki ruang penyimpanan (*storage*) yang memadai dan waktu pengiriman yang singkat. Semakin besar *file* teks yang akan dikirimkan maka semakin lama juga waktu yang dibutuhkan. Oleh karena itu, diperlukan langkah tambahan untuk mengefisiensikan media penyimpanan dengan melakukan kompresi agar ukurannya menjadi lebih kecil. Penelitian ini menggabungkan perpaduan teknik kriptografi Bacons Cipher dan teknik kompresi Even Rodeh Code. Penelitian ini menghasilkan sebuah aplikasi yang dapat dijadikan sebagai alternatif solusi dalam menjaga kerahasiaan *file* teks sehingga hanya dapat diakses oleh pemilik data dan dapat menghemat kebutuhan akan ruang penyimpanan (*storage*) data menjadi lebih efisian. Penerapan kompresi Even Rodeh Code juga dapat meningkatkan keamanan dari algoritma Bacons Cipher yang merupakan algoritma kriptografi klasik karena setelah dikompresi akan menghasilkan teks yang lebih acak serta tidak memperlihatkan pola-pola keterhubungannya dengan teks asli.

Kata Kunci: Bacons Cipher, Even Rodeh Code, Kriptografi, Kompresi

I. PENDAHULUAN

Keamanan dan kerahasiaan sebuah informasi pada *file* teks merupakan hal yang sangat penting untuk dilakukan, terutama informasi sensitif atau pribadi yang hanya boleh diakses oleh pihak yang berhak saja, baik pada saat disimpan dalam media penyimpanan (*hard drive*) ataupun pada saat akan dikirimkan. Hal tersebut terlebih lagi jika pengirimannya dilakukan melalui jaringan publik, maka sangat rentan disadap dan diketahui isi informasinya oleh pihak-pihak yang tidak memiliki wewenang. Salah satu cara yang dapat dilakukan untuk mengamankan informasi tersebut adalah dengan teknik kriptografi.

Selain aspek keamanan, maka hal yang perlu diperhatikan juga adalah tentang memori penyimpanan. Di era *modern* yang serba digital saat ini, pertukaran informasi dapat dilakukan secara nirkabel melalui media digital dimana saja dan kapan saja, hal ini mengharuskan pengguna untuk memiliki ruang penyimpanan (*storage*) yang memadai dan waktu pengiriman yang singkat. Semakin besar *file* teks yang akan dikirimkan maka semakin lama juga waktu yang dibutuhkan. Oleh karena itu, diperlukan langkah tambahan untuk mengefisiensikan media penyimpanan serta mempercepat proses transmisi dengan melakukan kompresi terlebih dahulu supaya ukurannya menjadi lebih kecil.

Penelitian ini, akan menggabungkan perpaduan teknik kriptografi dan teknik yang memanfaatkan dua metode kedalam satu proses, artinya pesan pada *file* teks akan di enkripsi terlebih dahulu dengan algoritma Bacons Cipher untuk mengamankan pesan pada *file* teks, kemudian dikompresi menggunakan metode Even Rodeh Code guna memperkecil ukuran *file* teks. Perpaduan kedua teknik ini juga dapat memperkuat keamanan dari algoritma Bacons Cipher karena hasil enkripsi (*ciphertext*) setelah dikompresi akan menghasilkan teks yang benar-benar acak serta tidak memperlihatkan pola keterhubungannya dengan teks asli (*plaintext*).

Terdapat banyak algoritma kriptografi yang masing-masing mempunyai kelebihan dan kelemahannya tersendiri. Penelitan ini akan menggunakan algoritma Bacons Cipher. Bacons Cipher merupakan salah satu teknik steganografi yang klasik dan unik, namun tidak terlalu umum diketahui orang. Walaupun namanya menyiratkan kriptografi (cipher), namun Bacons Cipher merupakan teknik steganografi (Adrian, 2009). Pada proses kompresi, penelitian ini menggunakan algoritma Even Rodeh Code. Alasan peneliti menggunakan metode kompresi Even Rodeh Code berdasarkan hasil kajian penelitian yang dilakukan oleh (Ihsan & Utomo, 2020) menjelaskan bahwa algoritma Even Rodeh Code (50%) lebih baik dari algoritma Subexponential Code (42%). Nilai rasio kompresi dipengaruhi oleh isi dari *file* yang

p-ISSN : 2406-7733
e-ISSN : 2597-9922

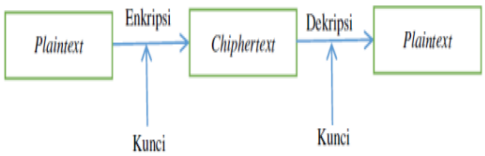
dikompresi. Semakin banyak pengulangan karakter pada suatu *file* yang dikompresi maka semakin tinggi pula rasio kompresinya. Pada penelitian yang dilakukan oleh (Auliyah, 2020), jika dibandingkan dari segi urutan proses algoritma antara enkripsi-kompresi dan kompresi-enkripsi, maka yang memberikan hasil lebih efisien adalah urutan enkripsi-kompresi, hal ini dikarenakan ukuran file hasil proses bergantung dari proses kompresi bukan proses enkripsi yang dalam hal ini malah memperbesar ukuran data. Dalam penelitian (Mesran & Nasution, 2020) menyimpulkan bahwa kombinasi dari teknik enkripsi dan teknik kompresi dapat meningkatkan keamanan data. Hasil enkripsi (*ciphertext*) yang telah dikompresi terlihat seperti hasil enkripsi dari algoritma kriptografi *modern*, karena setelah dikompresi akan menghasilkan teks yang lebih acak serta tidak memperlihatkan pola-pola keterhubungannya dengan teks asli.

II. TINJAUAN PUSTAKA

Kriptografi dan Kompresi

Kriptografi (*cryptography*) merupakan ilmu dan seni untuk menjaga pesan agar aman. *Crypto* berarti *secret* yang artinya rahasia dan *graphy* berarti *writing* yang artinya tulisan. Kriptografi dapat diartikan sebagai tulisan atau pesan rahasia. Pesan yang dirahasiakan dinamakan *plaintext*, sedangkan pesan hasil penyandian disebut *ciphertext* (Sukmawati et al., 2021. Kriptografi banyak digunakan untuk pengamanan data atau informasi saat ini (Susanti, 2020) seperti mengamankan pesan pada *file* teks.

Penggunaan kriptografi untuk mengamankan data merupakan hal yang mutlak diperlukan. Adanya kriptografi juga menjamin bukan hanya keamanan atas data-data yang dilindungi atau disembunyikan dari entitas tidak berkepentingan, melainkan juga menjaga otentifikasi daripada pengguna dalam hal ini pengirim dan penerima atas data tersebut. Sebelum munculnya *username*, *password*, data transaksi, dan email, enkripsi sangat dibutuhkan, terutama dalam hal berbagi informasi untuk keperluan militer (Ridho et al., 2022). Terdapat dua proses dalam kriptografi yaitu proses enkripsi dan dekripsi yang dapat dijelaskan pada gambar 1.



Gambar 1. Skema Proses Enkripsi dan Deskripsi
Sumber : (Darmayanti et al., 2018)

Pada gambar 1, proses kriptografi terdiri dari enkripsi dan dekripsi, dimana enkripsi adalah proses mengubah pesan asli (*plaintext*) menjadi pesan yang disandikan (*ciphertext*). Proses dekripsi adalah proses mengembalikan pesan yang disandikan (*ciphertext*) menjadi pesan asli (*plaintext*) sehingga informasi tersebut terjaga kerahasiaannya pada saat sampai ke tujuan yang mana proses dekripsi bekerja dalam urutan terbalik. Algoritma kriptografi merupakan langkah-langkah logis bagaimana menyembunyikan pesan dari orang-orang yang tidak berhak atas pesan tersebut.

Algoritma kriptografi terdiri dari tiga fungsi dasar (Surbakti, 2019) yaitu sebagai berikut:

1. Enkripsi, merupakan hal yang sangat penting dalam kriptografi, merupakan pengamanan data yang dikirim agar terjadi kerahasiaannya. Pesan asli disebut *plaintext*, yang diubah menjadi kode-kode yang tidak dimengerti. Enkripsi bisa diartikan sebagai *cipher* atau kode dengan menggunakan algoritma yang untuk mengkodekan data yang kita inginkan.
2. Dekripsi, merupakan kebalikan dari proses enkripsi. Pesan yang telah dienkripsi dikembalikan ke bentuk asalnya (teks asli). Algoritma yang digunakan untuk depenelitian tentu berbeda dengan algoritma yang digunakan untuk enkripsi.
3. Kunci, adalah yang dipakai untuk melakukan enkripsi dan dekripsi. Kunci terbagi menjadi dua bagian, kunci rahasia (*private key*) dan kunci umum (*public key*). Peran kunci yang digunakan pada metode kriptografi adalah suatu informasi yang mengendalikan jalannya sebuah metode kriptografi (Susanti, 2020).

Kompresi data merupakan cabang ilmu komputer yang bersumber dari teori informasi. Teori informasi mempelajari tentang *redundancy* (informasi tidak berguna) pada pesan. Semakin banyak *redundancy* semakin besar pula ukuran pesan dan upaya mengurangi *redundancy* inilah yang akhirnya melahirkan subjek ilmu tentang kompresi data (Auliyah, 2020). Pada kompresi data terdapat dua buah tipe teknik kompresi yang pertama teknik *lossless* dan yang kedua adalah teknik *lossy* (Pujianto et al., 2020). Perbedaan utama antara teknik kompresi *lossy* dan *lossless* terletak pada kualitasnya.

Pada kompresi, terdapat beberapa faktor penting yang perlu diperhatikan sebagai bahan pertimbangan untuk mengukur kualitas dari metode kompresi (Nabila, 2019), yaitu *Compression Ratio* (CR), *Space Savings* (SS), dan Running Time (RT).

1. *Compression Ratio* (CR)
   Compression Ratio atau rasio kompresi adalah menghitung kinerja dari repersentase data yang sudah dikompresi dan sebelum dikompresi. Persamaan (1) digunakan untuk mencari nilai *Compression Ratio*.
   $$CR = \frac{Compressed\ bits}{Uncompressed\ bits} \dots\dots\dots\dots\dots\dots\dots\dots (1)$$
2. *Space Savings* (SS)
   Space Savings adalah persentase penghematan ruang (memori) setelah file dikompresi dengan

p-ISSN : 2406-7733
e-ISSN : 2597-9922

mencari persentase selisih antara data awal sebelum dikompresi dengan hasil data yang telah dikompresi. Persamaan (2) digunakan untuk mencari nilai *Space Savings*.

$$SS = \left(1 - \frac{Compressed\ bits}{Uncompressed\ bits}\right) x\ 100 \dots\dots\dots (2)$$

3. *Running Time* (RT)
   Running Time adalah lama waktu yang dibutuhkan untuk melakukan proses kompresi dan dekompresi dari mulai pembacaan data hingga proses *encoding* pada data tersebut.

Algorima Bacons Cipher

Bacons Cipher merupakan salah satu teknik steganografi yang klasik dan unik, namun tidak terlalu umum diketahui orang. Walaupun namanya menyiratkan kriptografi (*cipher*), namun Bacons Cipher merupakan teknik steganografi. Bacons Cipher diciptakan oleh Sir Francis Bacon pada sekitar abad 16. Namun keberadaannya serta pembahasannya baru mencuat pada akhir abad 19. Bacons Cipher telah menyembunyikan banyak rahasia pada masanya (Adrian, 2009). Pada dasarnya, algoritma Bacons Cipher menggunakan cara baconian berpusat pada tabel 1.

Tabel 1. Konversi Alphabet pada Bacons Cipher

Karakter	Kode	Karakter	Kode	Karakter	Kode
a	AAAAA	j	ABAAB	s	BAABA
b	AAAAB	k	ABABA	t	BAABB
c	AAABA	l	ABABB	u	BABAA
d	AAABB	m	ABBAA	v	BABAB
e	AABAA	n	ABBAB	w	BABBA
f	AABAB	o	ABBBA	x	BABBB
g	AABBA	p	ABBBB	y	BBAAA
h	AABBB	q	BAAAA	z	BBAAB
i	ABAAA	r	BAAAB	spasi	BBBAA

Sumber : (Adrian, 2009)

Dasar dari cara kerja Bacons Cipher adalah dengan menggantikan atau melambangkan sebuah alphabet dengan sebuah deretan huruf yang dapat juga di analogikan dengan bilangan *biner* 0 dan 1. Satu kelompok bilangan *biner* yang terdiri dari 5 digit angka ini akan merepresentasikan bagaimana huruf akan digambarkan, apakah huruf kapital atau huruf kecil (Adrian, 2009)

Algorima Even Rodeh Code

Algoritma Even Rodeh Code adalah algoritma kompresi data yang pengkodeannya menggunakan karakter-karakter dengan beberapa rangkaian bit yang mewakili karakter yang dibuat berdasarkan frekuensi setiap karakter. Kode Even Rodeh hampir sama dengan kode Omega, perbedaannya adalah bahwa panjang kode ditopang sampai panjang 3-bit tercapai dan menjadi grup kode paling kiri (Pramadi et al., 2019). Algoritma Even Rodeh Code ini bersifat *lossless*, dimana hal ini merupakan suatu kelebihan karena dengan kompresi *lossless* dapat meminimalisir adanya kehilangan atau kerusakan data yang terjadi pada sumber data asli yang sangat penting (Nabila, 2019).

Kode algoritma Even Rodeh Code dapat disajikan pada tabel 2.

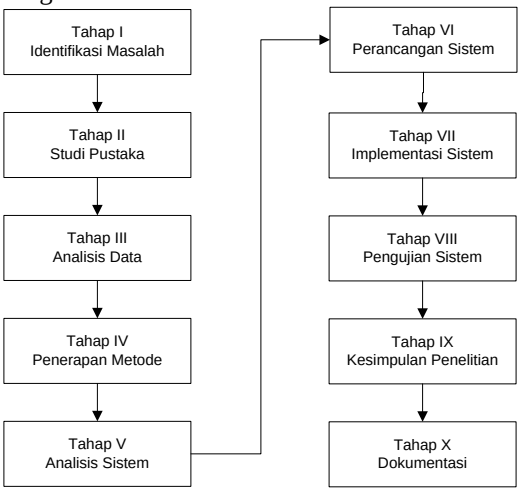
Tabel 2. Kode Algoritma Even Rodeh Code

N	Even Rodeh Code
0	000
1	001
2	010
3	011
4	100 0
7	111 0
8	100 1000 0
15	100 1111 0
16	100 10000 0
32	110 100000 0
100	111 1100100 0
1000	110 1100100 0

Sumber : (Pramadi et al., 2019)

III. METODE PENELITIAN

Kerangka kerja pada penelitian ini terdiri dari beberapa tahapan, mulai dari identifikasi masalah sampai dokumentasi penelitian yang dapat disajikan pada gambar 2.



Gambar 2. Tahapan Penelitian

Pada gambar 2, tahapan penelitian dimulai dengan melakukan identifikasi masalah, studi pustaka, analisis data, penerapan metode, analisis sistem, perancangan sistem, pengujian sitem, membuat kesimpulan dari hasil penelitian, kemudian tahapan terakhir melakukan dokumentasi dalam pembuatan laporan penelitian.

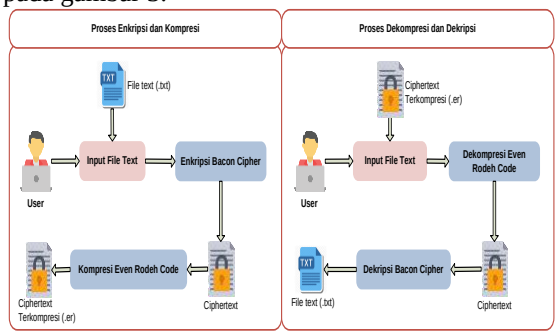
IV. HASIL DAN PEMBAHASAN

Penelitian ini bertujuan untuk mengamankan pesan pada *file* teks dengan menggabungkan perpaduan teknik kriptografi dan teknik kompresi, yang memanfaatkan dua metode kedalam satu proses, artinya pesan pada *file* teks akan dienkripsi terlebih dahulu dengan algoritma Bacons Cipher untuk mengamankan pesan pada *file* teks, kemudian dikompresi menggunakan algoritma Even Rodeh

p-ISSN : 2406-7733
e-ISSN : 2597-9922

Code guna memperkecil ukuran pesan apada file teks hasil enkripsi. Perpaduan kedua teknik ini juga dapat memperkuat keamanan dari algoritma Bacons Cipher. Hal ini karena hasil enkripsi setelah dikompresi akan menghasilkan teks yang benar-benar acak serta tidak memperlihatkan pola-pola keterhubungannya dengan teks asli.

Adapun gambaran umum sistem yang akan diterapkan dalam penelitian ini dapat ditampilkan pada gambar 3.

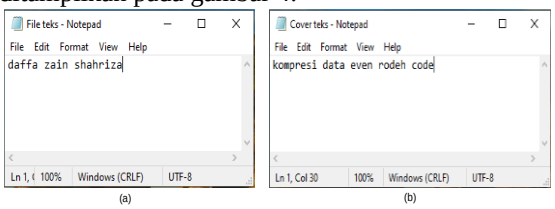


Gambar 3. Gambaran Umum Sistem

Gambar 3 merupakan gambaran umum dari proses sistem yang akan diterapkan dalam penelitian ini. Terdapat dua proses utama yang dilakukan dalam tahap ini, yaitu proses enkripsi dan kompresi serta proses dekompresi dan dekripsi.

1. Proses Enkripsi dan Kompresi

Proses enkripsi dan kompresi dalam skema sistem seperti yang telah dijelaskan pada gambar 4 dimulai dengan mengenkripsi pesan pada *file* teks menggunakan algoritma Bacons Cipher. Adapun sampel data dari *file* teks dan *cover file* teks yang digunakan pada proses perhitungan enkripsi dengan menggunakan algoritma Bacons Cipher dapat ditampilkan pada gambar 4.



Gambar 4. Sampel (a) Pesan (b) Cover File Teks

Gambar 4 merupakan sampel data yang akan digunakan dalam mensimulasikan proses enkripsi pesan pada *file* teks dengan menggunakan algoritma Bacons Cipher. Berdasarkan gambar 3 maka dapat diketahui bahwa pesan (*palintext*) dan *cover* teks yang dijadikan untuk menyembunyikan pesan yaitu:

Pesan teks : daffa zain syahriza

Cover teks : kompresi data Even Rodeh Code

Proses enkripsi dengan algoritma Bacons Cipher pada langkah pertama dilakukan dengan mengkonversi *alphabet* berdasarkan tabel 1. Guna mempermudah dalam proses enkripsi, maka jumlah karakter yang diambil dari *plaintext* hanya 5 karakter pertama saja. Adapun Proses enkripsi, yaitu dengan mengkodekan setiap huruf dari pesan (*plaintext*) yang diganti dengan sekelompok lima huruf 'A' atau 'B' sehingga menjadi:

d = AAABB
a = AAAAA
f = AABAB
f = AABAB
a = AAAAA

Dari hasil konversi pesan (*plaintext*) yang diganti dengan sekelompok lima huruf 'A' atau 'B' akan diperoleh hasil enkripsi (*ciphertext*) yang menjadi:

Ciphertext = AABBB AAAAA AABAB AABAB AAAAA

Hasil enkripsi (*ciphertext*) yang sudah dienkripsi kemudian akan disamarkan dengan cara menyisipkan *ciphertext* kedalam *cover* pesan (*cover message*) dengan menggunakan aturan modifikasi huruf kapital, dimana:

'A' = huruf kecil dan 'B' = huruf kapital

Pesan hasil enkripsi setelah disisipkan kedalam *cover* pesan (*cover message*), maka diperoleh hasil akhir proses enkripsi (*ciphertext*) dari algoritma Bacons Cipher, yaitu:

Tabel 3. Proses Enkripsi Bacons Cipher

Plaintext	daffa				
Konversi	d	a	f	f	a
	AAABB	AAAAA	AABAB	AABAB	AAAAA
	⬇	⬇	⬇	⬇	⬇
Cover Message	kompr	esida	taeve	nrode	hcode
Ciphertext	komPR	esida	taEvE	nrOdE	hcode

Plaintext = daffa (5 karakter)

Ciphertext = komPR esida taEvE nrOdE hcode (29 karakter termasuk spasi)

Berdasarkan hasil proses enkripsi algoritma Bacons Cipher pada tabel 3, dapat diketahui bahwa panjang karakter hasil enkripsi (*ciphertext*) setelah disamarkan kedalam *cover* pesan (*cover message*) akan mengalami peningkatan lima kali lebih besar dari pesan asli (*plaintext*). Dimana pesan asli (*plaintext*) mempunyai jumlah karakter sebanyak 5 karakter dan jumlah karakter *ciphertext* sebanyak 29 karakter. Oleh karena itu diperlukan metode kompresi untuk memperkecil hasil enkripsi, dimana pada penelitian ini akan menerapkan algoritma Even Rodeh Code dalam proses kompresi.

Setelah hasil enkripsi (*ciphertext*) didapatkan maka tahap selanjutnya dilakukan proses kompresi untuk memperkecil ukuran data pada *ciphertext*. Metode kompresi yang digunakan dalam penelitian ini yaitu metode *lossless compression* dengan menggunakan algoritma Even Rodeh Code. Tahap awal dalam melakukan kompresi adalah akan dilakukan pengelompokan nilai karakter hasil enkripsi (*ciphertext*) berdasarkan nilai frekuensi. Urutan karakter nilai *ciphertext* yang hasilnya dapat ditunjukkan pada tabel 4.

p-ISSN : 2406-7733
e-ISSN : 2597-9922

Tabel 4. Frekuensi Karakter Ciphertext

No.	Karakter	Frekuensi
1.	k	1
2.	o	2
3.	m	1
4.	P	1
5.	R	1
6.	spasi	4
7.	e	2
8.	s	1
9.	i	1
10.	d	2
11.	a	2
12.	t	1
13.	E	3
14.	v	1
15.	n	1
16.	r	1
17.	O	1
18.	h	1
19.	c	1

Karakter pada *ciphertext* akan diurutkan berdasarkan frekuensi kemunculannya mulai dari karakter dengan frekuensi kemunculan terbesar ke karakter dengan frekuensi kemunculan terkecil. Jika terdapat lebih dari satu karakter dengan frekuensi kemunculan yang sama maka diurutkan berdasarkan abjad. Untuk mempermudah perhitungan dalam proses kompresi, maka jumlah karakter pada *ciphertext* yang akan dikompresi hanya mengambil 10 digit karakter saja. Adapun hasil pengurutan dari 10 karakter dan frekuensi karakter pada *ciphertext* dapat disajikan pada tabel 5.

Tabel 5. String Ciphertext Sebelum Dikompresi

Karakter	ASCII (Desimal)	ASCII (Biner)	Bit	Frekuensi	Bit x Frekuensi
spasi	32	00100000	8	4	32
E	69	01000101	8	3	24
d	100	01100100	8	3	24
a	97	01100001	8	2	16
e	101	01100101	8	2	16
o	111	01101111	8	2	16
O	74	01001010	8	1	8
P	80	01010000	8	1	8
R	82	01010010	8	1	8
c	99	01100011	8	1	8
Total Bit					**160**

Berdasarkan tabel 5, satu karakter bernilai 8 bit bilangan *biner* dalam kode ASCII. Sehingga 10 karakter dari *ciphertext* mempunyai nilai *biner* sebanyak 160 *bit*, artinya *string ciphertext* sebelum dikompresi adalah 10 *byte*, dimana 1 karakter adalah 1 *byte*, dan 1 *byte* adalah 8 *bit* maka total ukuran *ciphertext* yang akan di kompresi adalah 80x8=160 *bit* atau 10 *byte*. Berdasarkan tabel 5, maka didapat *string bit* dengan uraian sebagai berikut:

Spasi	E	d	a	e
00100000	01000101	01100100	01100001	01100101
o	**O**	**P**	**R**	**c**
01101111	01001010	01010000	01010010	01100011

Tahap selanjutnya adalah dilakukan proses pengkodean yang dimulai dengan jumlah frekuensi karakter terbesar hingga yang paling kecil sehingga didapat *n* (panjang *bit* dalam karakter yang telah diurutkan). Kompresi nilai dari *string* dengan nilai kode Even Rodeh didapat dari aturan yang ada pada tabel 1. Adapun hasil kompresinya dapat disajikan pada tabel 6.

Tabel 6. String Ciphertext Setelah Dikompresi

n	Karakter	Even Rodeh Code	Bit	Frekuensi	Bit x Frekuensi
0	spasi	000	3	4	12
1	E	001	3	3	9
2	d	010	3	3	9
3	a	011	3	2	6
4	e	1000	4	2	8
5	o	1010	4	2	8
6	O	1100	4	1	4
7	P	1110	4	1	4
8	R	10010000	8	1	8
9	c	10010010	8	1	8
Total Bit					**76**

Setelah proses kompresi selesai, berdasarkan kode-kode Even Rodeh Code string pada tabel 6, maka dilakukan penyusunan kode-kode tersebut sesuai dengan posisi karakter *string* asli pada *ciphertext* dengan rincian sebagai berikut:

Spasi	E	d	a	e
000	001	010	011	1000
o	**O**	**P**	**R**	**c**
1010	1100	1110	10010000	10010010

Setelah *string bit ciphertext* disusun berdasarkan kode-kode Even Rodeh Code sesuai dengan karakternya masing-masing, sehingga akan menjadi *string bit* berikut ini:

000001010011100010101100111010010000 10010010

Dalam tabel ASCII satu karakter dipresentasikan sebanyak 8 *bit* dengan bilangan *biner*. Namun, jika *bit* tersebut bukan kelipatan 8, maka dilakukan penambahan *padding bit* dan *flag bit*. *Padding* yaitu menambahkan *bit* 0 pada hasil kompresi yang bukan merupakan kelipatan 8. Sedangkan *flag* yaitu untuk menjelaskan berapa jumlah *bit* yang ditambahkan dalam melakukan *padding*. Terdapat 76 *bit* dari *string bit* hasil kompresi, maka dilakukan penambahan *bit* 0 sebanyak 4 kali agar jumlah *bit* data tersebut dapat habis bila dibagi dengan 8. Sehingga *bit-bit* data tersebut setelah diberikan *padding* menjadi:

000001010011100010101100111010010000 10010010000

Karena terdapat 4 *bit* penambahan *padding* maka *flag bits*-nya adalah bilangan *biner* dari 4 dengan panjang 8 bit yaitu 00000100, sehingga *bit-bit* datanya setelah diberikan *flag bits* menjadi:

000001010011100010101100111010010000 1001001000000000100

Selanjutnya akan dilakukan perhitungan untuk mengetahui kualitas dari algoritma kompresi yang digunakan. Adapun parameter yang digunakan dalam proses kompresi pada penelitian ini adalah *Compression Ratio* (CR) dan *Space Savings* (SS). Sebagaimana diketahu bahwa ukuran data sebelum dan setelah dilakukan kompresi yaitu:

Ukuran data sebelum di kompresi (*uncompressed bits*) = 160 *bit*

Ukuran data setelah di kompresi (*compressed bits*) = 88 *bit*

Adapun hasil perhitnungan kinerja sistem kriptografi dan kompresi berdasarkan parameter

p-ISSN : 2406-7733
e-ISSN : 2597-9922

Compression Ratio (CR) dan *Space Savings* (SS) dapat diuraikan sebagai berikut:

$$CR = \frac{Compressed\ bits}{Uncompressed\ bits} = \frac{88}{160} = 0.55$$

$$SS = \left(1 - \frac{Compressed\ bits}{Uncompressed\ bits}\right) x\ 100$$

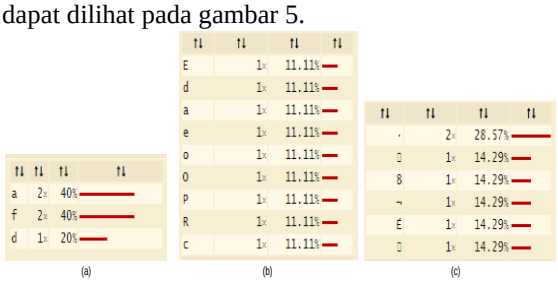
$$SS = \left(1 - \frac{88}{160}\right) x100\% = 45$$

Berdasarkan hasil perhitungan kinerja dari metode kompresi menggunakan algoritma kompresi Even Rodeh Code, nilai dari *Compression Ratio* (CR) mencapai 0.55 atau 55% dan Space Savings (SS) yang dihasilkan cukup besar mencapai 45%, hal ini membuktikan bahwa kinerja dari metode kompresi yang digunakan cukup efektif dan baik untuk kompresi pesan pada *file* teks. Adapun perbandingan data sebelum dan setelah diproses dapat disajikan pada tabel 7.

Tabel 7. Hasil Enkripsi dan Kompresi

File Teks		Hasil Enkripsi		Hasil Kompresi		
Plaintext	Size	Ciphertext	Size	CR	SS	Size
daffa	40 *bit*	EdaeoOPRc	160 bit	0.55	45	88 bit

Sesuai tabel 7, maka dapat disimpulkan bahwa *size* ata ukuran *file* teks sebelum di enkripsi dan dikompresi yaitu 40 *bit*. Setelah di enkripsi dan disisipkan kedalam *cover* pesan (*cover message*) menggunakan algoritma Bacons Cipher akan menghasilkan *ciphertext* yang lebih besar dari teks aslinya, yaitu 160 bit, oleh karena itu perlu dikompresi untuk memperkecil ukurannya. Setelah dikompresi dengan algoritma Even Rodeh Code, maka menjadi lebih kecil yaitu 88 *bit*, artinya terdapat selisih 72 *bit* yang dapat diperkecil. Adapun persentase frekuensi karakter antara *plaintext* dengan *ciphertext* setelah di enkripsi dan dikompresi dapat dilihat pada gambar 5.



Gambar 5. Analisis Frekuensi (a) Plaintext (b) Hasil Enkripsi dan (c) Hasil Kompresi

Hasil analisis frekuensi pada gambar 5 diperoleh dengan memanfaatkan *tools* yang diakses secara *online* dari *https://www.dcode.fr/frequency-analysis*. Dari gambar 5, *plaintext* menunjukkan jumlah karater (N=3), jumlah karakter hasil enkripsi (*ciphertext*) adalah (N=9). Sedangkan jumlah karakter hasil kompresi sebanyak (N=6). Dari hasil analisis frekuensi pada gambar 1 dapat disimpulkan bahwa terdapat 2 karakter pada *plaintext* yang berkorelasi dengan hasil enkripsi (*ciphertext*), yaitu karakter "a" dengan persentase sebesar 1% dan karakter "d" persentase sebesar 1%. Hal ini menunjukkan adanya pola keterhubungannya dengan *plaintext* namun dengan jumlah yang sedikit.

Jika frekuensi kemunculan karakter pada *plaintext* yang telah di enkripsi semakin sering atau semakin tinggi, maka tingkat keamanan informasi pada pesan (*plaintext*) lebih rendah dan lebih mudah dipecahkan dengan menggunakan metode analisis frekuensi. Sedangkan persentase frekuensi karakter antara *plaintext* setelah dilakuka proses kompresi pada *ciphertext*, maka tidak memperlihatkan adanya pola keterhubungan antara *plaintext* dengan hasil kompresi. Hal ini membuktikan bahwa hasil kompresi dengan menggunakan algoritma Even Rodeh Code dapat meningkatkan keamanan dari algoritma Bacons Cipher, karena setelah dikompresi akan menghasilkan teks yang lebih acak serta tidak memperlihatkan pola keterhubungannya dengan teks asli (*plaintext*).

2. Proses Dekompresi dan Dekripsi

Proses dekompresi dimulai dengan membaca karakter yang terdapat pada *file ciphertext* terkompresi. Selanjutnya menentukan panjang string *bit* yang harus dibaca dengan menghitung *padding* dan *flagging*. Berdasarkan hasil kompresi sebelumnya, *string bit* yang dihasilkan adalah:

0000010100111000101011001110100100001
0010010000000000100

Untuk proses dekompresi terhadap *string bit* yang telah dikompresi adalah dengan menentukan indeks terkahir untuk proses pembacaan *string*, yaitu total panjang *string bit* seluruhnya dikurang dengan *flag* ditambah padding atau dapat ditulis:

$n = panjang\ string\ bit - (flagging + padding)$
$n = 88 - (flagging + padding)$

Flagging pada *string bit* tersebut adalah 00000100 (8 *bit* terakhir pada *string bit)* yang bila dikonversi ke dalam desimal akan bernilai 4, maka *padding*-nya adalah 0000. Sehingga panjang *string bit* yang harus dibaca adalah panjang *string bit* (76 *bit*) terkompresi dikurang total jumlah panjang *padding* dan *flagging* (4+8). Maka diperoleh nilai panjang *string bit* yang harus dibaca adalah 88 - (4+8) = 76, sehingga akan menghasilkan *string bit* berikut:

0000010100111000101011001110100100001
0010010

Untuk melakukan dekompresi, maka pembacaan *string bit* dimulai dari indeks paling kecil (dilakukan dari indeks pertama sampai indeks terakhir) yang dapat diuraikan yaitu:

0000010100111000101011001110100100001
0010010
00000101001110001010110011101001000010010010

Indeks ke-0 adalah **0**, pada tabel 6 kode **0** tidak mewakilkan karakter apapun.

000000101001110001010110011101001000010010010

p-ISSN : 2406-7733
e-ISSN : 2597-9922

Maka dikuti oleh indeks ke-1 yaitu **0**, pada tabel 6 kode **00** tidak mewakilkan karakter apapun.

0000010100111000101011001110100100001
0010010

Maka dikuti oleh indeks ke-2 yaitu **0**, pada tabel 6 kode **000** mewakili karakter "spasi". Pembacaan selanjutnya dimulai pada indeks ke-3 yaitu **0**

00001010011100010101100111010010000 1
0010010

Maka dikuti oleh indeks ke-3 yaitu **0**, pada tabel 6 kode **0** tidak mewakilkan karakter apapun.

00000101001110001010110011101001000 01
0010010

Maka dikuti oleh indeks ke-4 yaitu **0**, pada tabel 6 kode **00** tidak mewakilkan karakter apapun.

00000101001110001010110011101001000 01
0010010

Maka dikuti oleh indeks ke-5 yaitu **1**, pada tabel 6 kode **001** mewakili karakter "E". Pembacaan selanjutnya dimulai pada indeks ke-6 yaitu **0**

Langkah tersebut akan dilakukan terus berlangsung sampai semua *string bit* habis sehingga didapatkan hasil dekompresi yaitu "spasi EdaeoOPRc" yang merupakan hasil dari proses enkripsi (*ciphertext*) sebelumnya. *Ciphertext* yang diperoleh dari proses dekompresi, selanjutnya akan dilakukan proses dekripsi yang bertujuan untuk mengembalikan pesan sudah di enkripsi pada langkah sebelumnya.

Proses dekripsi merupakan kebalikan dari proses enkripsi yaitu untuk mentransformasi *ciphertext* ke dalam bentuk yang dapat dimengerti (*plaintext*). Adapun untuk mengekstrak pesan tersembunyi (*hidden message*) yang terdapat dalam *ciphertext*, yaitu dengan mengkodekan setiap huruf dari *ciphertext* yang diganti dengan sekelompok lima huruf 'A' atau 'B', seperti pada tabel 8.

Tabel 8. Proses Dekripsi Bacons Cipher

Ciphertext	komPR	esida	taEvE	nrOdE	hcode
	⬇	⬇	⬇	⬇	⬇
Konversi	AAABB	AAAAA	AABAB	AABAB	AAAAA
	d	a	f	f	a
Plaintext	daffa				

Setelah mengkodekan setiap huruf dari *ciphertext* yang diganti dengan sekelompok lima huruf 'A' atau 'B', maka tahap selanjutnya dilakukan konversi ke dalam *alphabet* Bacons Cipher sesuai pada tabel 1 sehingga menjadi:

AAABB = d
AAAAA = a
AABAB = f
AABAB = f
AAAAA = a

Maka diperoleh kembali *plaintext* yang sama persis dengan teks aslinya (*plaintext*).

3. Implementasi Sistem

Implementasi dilakukan dengan tujuan untuk mengetahui bahwa program aplikasi yang dibuat sesuai dengan hasil analisis dan perancangan serta menghasilkan satu kesimpulan apakah program yang dibuat sesuai dengan yang diharapkan dan berhasil menjalankan fungsi-fungsinya dengan benar.

Gambar 6. Implementasi Hasil Enkripsi

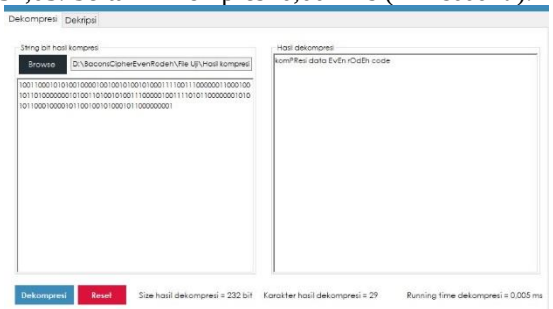
Pada kasus pengujian ini, *file* teks yang akan enkripsi yaitu format *file .txt* dengan rincian:

File teks (*plaintext*) : daffa (5 karakter)
Cover teks (*cover message*) : kompresi data Even Rodeh Code (29 karakter)
Hasil enkripsi (*ciphertext*) : komPR esida taEvE nrOdE hcode (29 karakter)
Running time enkripsi : 0 ms

Hasil enkripsi kemudian akan di kompresi, gambar 7 menampilkan hasil implementasi sistem.

Gambar 7. Implementasi Hasil Kompresi

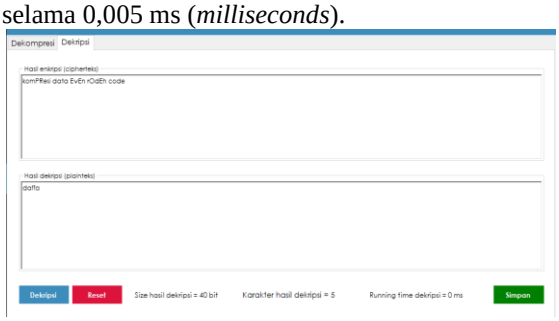
Berdasarkan gambar 7, dapat disimpulkan bahwa *size* ata ukuran *file* teks hasil enkripsi (*ciphertext*) sebelum dikompresi yaitu 232 *bit* dan setelah dikompresi menjadi lebih kecil yaitu 160 *bit*, artinya terdapat selisih 72 *bit* yang dapat diperkecil dengan nilai CR sebesar 68,97 dan nilai SS sebesar 31,03. Serta RT kompresi 0,001 ms (*millisecond*).



Gambar 8. Implementasi Hasil Dekompresi

p-ISSN : 2406-7733
e-ISSN : 2597-9922

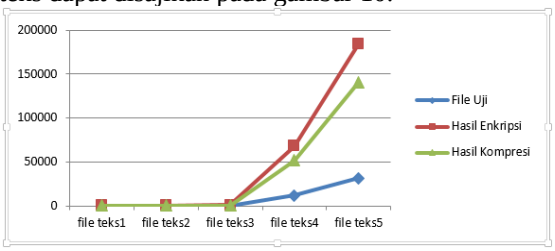
Gambar 8 merupakan hasil dari proses dekompresi dengan menggunakan algoritma Even Rodeh Code. *Size* dari hasil dekompresi adalah 232 *bit* dengan jumlah karakter sebanyak 29 karakter, yang merupakan hasil dari proses enkripsi dan kompresi sebelumnya. *Running time* yang dibutuhkan untuk melakukan proses dekompresi selama 0,005 ms (*milliseconds*).



Gambar 9. Implementasi Hasil Dekripsi

Gambar 9 merupakan hasil dari proses dekripsi yang terdapat pada halaman *decoding*. Dari hasil pengujian yang dilakukan, sistem dapat melakukan proses dekripsi sehingga diperoleh kembali pesan teks aslinya. *Size* hasil dekripsi adalah 40 *bit* yang merupakan size dari *file* teks aslinya. *Running time* yang dibutuhkan untuk melakukan proses dekripsi selama 0 ms (*milliseconds*).

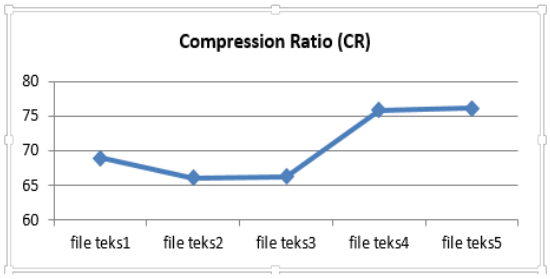
Adapun hasil pengujian mengenai perbandingan antara hasil enkripsi dan hasil kompresi dengan menggunakan 5 sampel data *file* teks dapat disajikan pada gambar 10.



Gambar 10. Diagram Hasil Pengujian

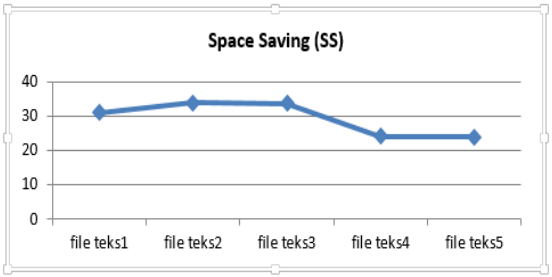
Berdasarkan gambar 10, *size* hasil enkripsi (*ciphertext*) akan mengalami peningkatab secara konstan seiring dengan perubahan *size* pada *file* uji yang digunakan. Hal ini menunjukkan bahwa *size file* uji berbanding lurus dengan *size* hasil enkripsi (*ciphertext*). Artinya semakin besar *size file* uji yang akan dienkripsi maka semakin besar juga nilai *size* hasil enkripsi (*ciphertext*) yang dihasilkan, begitu juga sebaliknya. Sedangkan korelasi antara *size* hasil enkripsi (*ciphertext*) juga berbanding lurus dengan *size* hasil kompresi.

Berdasarkan gambar 10, memperlihatkan hasil kompresi untuk *file* uji yang memiliki *size* (dalam satuan *bit*) yang berbeda menghasilkan nilai CR, SS, dab RT yang berbeda juga. Namun agar lebih memudahkan untuk melihat perbandingan efektifitas dari hasil kompresi untuk parameter CR, maka dibuatkan dalam bentuk diagram seperti pada gambar 11.



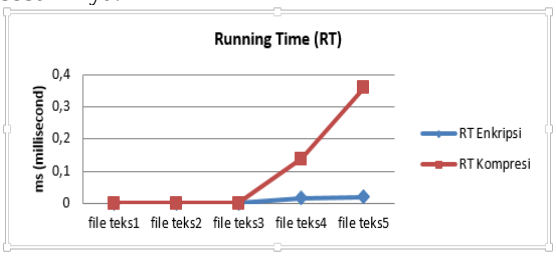
Gambar 11. Diagram Perbandingan Nilai CR

Gambar 11 memperlihatkan perbandingan nilai *Compression Ratio* (CR) untuk setiap *file* uji. CR mengalami penuruan nilai secara konstan seiring dengan perubahan *size file* uji yang digunakan. Hal ini menunjukkan bahwa *size file* uji berbanding lurus dengan nilai CR. Artinya semakin besar *size file* uji yang akan dikompresi maka semakin besar juga nilai CR yang dihasilkan, begitu juga sebaliknya.



Gambar 12. Diagram Perbandingan Nilai SS

Gambar 12 memperlihatkan perbandingan nilai *Space Saving* (SS) untuk setiap file uji. SS mengalami peningkatan nilai secara konstan seiring dengan perubahan *size file* uji yang digunakan. Hal ini menunjukkan bahwa *size file* uji berbanding lurus dengan nilai SS. Semakin besar *size file* uji yang akan dikompresi maka semakin kecil nilai SS yang dihasilkan, artinya semakin sedikit ruang penyimpanan yang dapat di hemat, begitu juga sebaliknya.



Gambar 13. Diagram Perbandingan Nilai RT

Gambar 13 memperlihatkan perbandingan nilai *Running Time* (RT) untuk setiap *file* uji. RT mengalami peningkatan nilai secara konstan seiring dengan perubahan *size file* uji yang digunakan. Hal ini menunjukkan bahwa *size file* uji berbanding lurus dengan nilai RT. Artinya semakin besar *size file* uji yang akan dikompresi maka semakin lama juga waktu yang dibutuhkan untuk melakukan proses enkripsi dan kompresi, begitu juga sebaliknya.

p-ISSN : 2406-7733
e-ISSN : 2597-9922

V. KESIMPULAN DAN SARAN

Kesimpulan

Kesimpulan yang dapat diambil setelah melakukan implementasi dan pengujian sistem dalam optimasi keamanan pesan pada *file* teks menggunakan kombinasi algoritma kriptografi Bacons Cipher dan algoritma kompresi Even Rodeh Code, yaitu sebagai berikut:

1. Prosedur dalam proses enkripsi dan kompresi bekerja dengan cara mengenkripsi pesan yang terdapat dalam *file* teks menggunakan algoritma Bacons Cipher sehingga hanya dapat diakses oleh pemilik data. Hasil enkripsi lalu dikompresi dengan algoritma Even Rodeh Code agar didapatkan size yang lebih kecil sehingga dapat menghemat kebutuhan akan ruang penyimpanan (storage) jadi lebih efisien.
2. Kompresi Even Rodeh Code dapat memperkecil ukuran data pada *file* teks setelah dienkripsi serta dapat meningkatkan keamanan dari algoritma Bacons Cipher yang merupakan algoritma kriptografi klasik karena setelah dikompresi akan menghasilkan teks yang lebih acak serta tidak memperlihatkan pola-pola keterhubungannya dengan teks asli.
3. Sistem enkripsi dan kompresi dapat di implementasikan kedalam sebuah aplikasi berbasis *desktop* dengan menggunakan bahasa pemrograman *Visual C#.NET* yang dapat digunakan untuk menjaga kerahasiaan data serta memperkecil ukuran data pada *file* teks.

Saran

Adapun beberapa saran untuk pengembangan dalam penelitian selanjutnya adalah sebagai berikut:

1. Penelitian ini hanya fokus untuk melakukan enkripsi dan kompresi berupa *string* pada *file* teks dengan format *.txt*, maka pada penelitian berikutnya diharapkan dapat mengenkripsi dan kompresi semua komponen yang terdapat pada *file* teks, seperti tabel dan gambar serta format file lainnya seperti audio dan video.
2. Implementasi dari hasil penelitian ini berupa aplikasi berbasis *desktop*, maka untuk penelitian selanjutnya dapat menerapkannya pada aplikasi berbasis *web* atau *mobile* sehingga penerapannya akan lebih beragam.

DAFTAR PUSTAKA

Adrian, A. (2009). Bacons Cipher. Makalah Kriptografi (setengah semester pertama, pengganti UTS). https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2008-2009/Makalah1-2009.html

Auliyah, A. I. (2020). Implementasi Kombinasi Algoritma Enkripsi Rivest Shamir Adleman (RSA) dan Algoritma Kompresi Huffman Pada File Document. Indonesian Journal of Data and Science, 1(1), 23–28. https://doi.org/10.33096/ijodas.v1i1.6

Darmayanti, I., Astrida, D. N., & Ariyus, D. (2018). Penerapan Keamanan Pesan Teks Menggunakan Modifikasi Algoritma Caisar Chiper Kedalam Bentuk Sandi Morse. Jurnal Ilmiah IT CIDA, 4(1), 39–47. https://doi.org/10.55635/jic.v4i1.78

Ihsan, I., & Utomo, D. P. (2020). Analisis Perbandingan Algoritma Even-Rodeh Code Dan Algoritma Subexponential Code Untuk Kompresi File Teks. KOMIK (Konferensi Nasional Teknologi Informasi Dan Komputer), 4(1), 223–227. https://doi.org/10.30865/komik.v4i1.2684

Mesran., & Nasution. S. D. (2021). Peningkatan Keamanan Kriptografi Caesar Cipher dengan Menerapkan Algoritma Kompresi "Stout Codes". Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi), 4(6), 1209-1215. DOI: https://doi.org/10.29207/resti.v4i6.2730

Sukmawati, R. A., Riski, A., & Kamsyakawuni, A. (2021). Perbandingan Playfair Cipher Dengan 3D Playfair Cipher Pada Pengamanan Citra. Majalah Ilmiah Matematika Dan Statistika, 21(1), 15–24. https://doi.org/10.19184/mims.v21i1.23116

Susanti, D. (2020). Analisis Modifikasi Metode Playfair Cipher Dalam Pengamanan Data Teks. Indonesian Journal of Data and Science, 1(1), 11–18. https://doi.org/10.33096/ijodas.v1i1.4

Pramadi, A. A., Nasution, S. D., & Purba, B. (2019). Penerapan Algoritma Even-Rodeh Pada Aplikasi Kompresi File Gambar. KOMIK (Konferensi Nasional Teknologi Informasi Dan Komputer), 3(1), 73–84. https://doi.org/10.30865/komik.v3i1.1570

Nabila, G. A. (2019). Analisis Perbandingan Algroitma Boldi Vigna ζ1 Code dan Alagoritma Even-Rodeh Code Pada Kompresi File Teks. Repsitory USU, 82–91

Pujianto, Mujito, Prasetyo, B. H., & Prabowo, D. (2020). Perbandingan Metode Huffman dan Run Length Encoding Pada Kompresi Document. InfoTekJar: Jurnal Nasional Informatika Dan Teknologi Jaringan, 5(1), 216–223. https://doi.org/10.30743/infotekjar.v5i1.2892

Ridho, A., Mutia, C., & Sinaga, A. P. (2022). Analisis Enkripsi dan Dekripsi Cipher Teks Menggunakan Kombinasi Gronsfeld Cipher Dengan Reverse Cipher. Jurnal Teknik Informatika Kaputama (JTIK), 6(1)

p-ISSN : 2406-7733
e-ISSN : 2597-9922