Journal Global Tecnology Computer
Vol 3, No. 3, Agustus 2024, Hal 95-104
ISSN: 2809-6118 (Online - Elektronik)
DOI: 10.47065/jogtc.v3i3.6257
Website https://ejurnal.seminar-id.com/index.php/jogtc

Implementasi Algoritma Golomb Code Dan Algoritma Kriptografi RSA Untuk Kompresi Dan Pengamanan File Citra Terkompresi

Reuny Theovani Saragih

Fakultas Ilmu Komputer dan Teknologi Informasi, Program Studi Teknik Informatika, Universitas Budi Darma, Medan, Indonesia
Jl. Sisingamangaraja No.338, Siti Rejo I, Kec. Medan Kota, Kota Medan, Sumatera Utara, Indonesia
Email: theovanisaragihreuny@gmail.com

Abstrak−Pada era saat ini kebutuhan informasi sangatlah penting bagi masyarakat umum karena semakin banyak informasi yang disimpan secara digital maka, semakin besar pula media penyimpanan data yang dibutuhkan. Oleh karena itu diperlukan suatu proses penyimpanan data alternatif supaya dapat menyimpan data sebanyak-banyaknya dengan menggunakan media penyimpanan yang ada. Dalam kompresi *file* citra, dekompresi sangat diperlukan agar proses pemampatan data bekerja secara optimal. Proses kompresi digunakan untuk memperkecil ukuran *file,* sementara proses dekompresi merupakan cara untuk mengembalikan data yang terkompresi menjadi data asli. Pada dasarnya teknik kompresi citra digunakan untuk mengetahui cara transmisi data (data *transmision*) dan penyimpanan data (*storage*). Metode yang dilakukan dalam penelitian ini adalah algoritma Golomb Code proses kompresi pada *file* citra dan Algoritma RSA (*Rivest-Shamir-Adleman*) untuk mengamankan *file* citra yang sudah dikompresi. Diharapkan bahwa implementasi algoritma Golomb Code untuk kompresi file citra dengan menerapkan algoritma RSA untuk pengamanan *file* citra setelah dikompresi akan memberikan hasil yang efisien dalam hal ukuran file citra hasil kompresi, serta memberikan tingkat keamanan yang memadai untuk melindungi data terkompresi. Dalam menentukan hasil kriteria kompresi menggunakan algoritma *golomb code* tersebut adalah *Rasio Of Compression* (RC) = 1,63 %, Compression Rasio (CR) = 61,11%, *Redudancy* (Rd) = 38,88%, *Space Saving* (Ss) = 38,89%.

Kata Kunci: Teknologi; Kompresi File Citra; Golomb Code; RSA; Keamanan Data.

Abstract−*In the current era, information needs are very important for the general public because the more information is stored digitally, the greater the data storage media needed. Therefore, an alternative data storage process is needed in order to store as much data as possible using existing storage media. In image file compression, decompression is very necessary so that the data compression process works optimally. The compression process is used to reduce file size, while the decompression process is a way to restore compressed data to original data. Basically, image compression techniques are used to find out how to transmit data (data transmission) and store data (storage). The method used in this research is the Golomb Code algorithm for the compression process on image files and the RSA (Rivest-Shamir-Adleman) algorithm to secure image files that have been compressed. It is hoped that the implementation of the Golomb Code algorithm for image file compression by applying the RSA algorithm for securing image files after compression will provide efficient results in terms of the size of the compressed image file, as well as providing an adequate level of security to protect the compressed data. In determining the compression criteria results using the Golomb code algorithm, they are Compression Ratio (RC) = 1.63%, Compression Ratio (CR) = 61.11%, Redundancy (Rd) = 38.88%, Space Saving (Ss) = 38.89%.*

Keywords*: Technology; Image File Compression; Golomb Code; RSA; Data Security.*

1. PENDAHULUAN

Perkembangan teknologi di dunia digital saat ini telah membawa banyak perubahan khususnya pada data yang semakin meningkat di berbagai bidang, seperti multimedia, ilmu pengetahuan, bisnis, dan sebagainya. Data yang besar dan kompleks kerap menimbulkan tantangan dalam hal penyimpanan, transfer, dan pengolahan. Oleh karena itu, diperlukan teknik kompresi data yang efisien untuk mengurangi ukuran data tanpa mengorbankan kualitas atau informasi yang penting, dilakukan pengamanan data setelah *file* dikompresi. Hal ini dilakukan karena untuk menghindari hal-hal yang tidak diinginkan. Misalnya, penyebaran citra oleh orang-orang yang tidak bertanggung jawab, manipulasi citra serta merubah citra (pengeditan) menjadi citra yang berbau negatif dan lain sebagainya [1][2].

Kompresi merupakan salah satu ilmu pengetahuan di bidang komputer yang bertujuan untuk mengurangi ukuran *file* sebelum menyimpan kedalam media penyimpanan *(storage device)*. Salah satu algoritma kompresi yang efektif dan banyak digunakan adalah Algoritma Golomb Code. Algoritma ini dikembangkan khusus untuk data dengan pola kejadian yang berulang, seperti data citra. Golomb code memanfaatkan pengkodean delta untuk mengurangi jumlah bit yang digunakan untuk menyimpan data, sehingga menghasilkan ukuran *file* yang lebih kecil [3][4]. Menurut Bruce dalam bukunya yang berjudul "Applied Cryptography 2nd", kriptografi merupakan ilmu sekaligus seni untukmenjaga keamanan data. Namun, selain mengompresi *file* citra, penting juga untuk menjaga keamanan data tersebut. Dalam konteks ini, algoritma RSA *(Rivest-Shamir-Adleman)* dapat digunakan untuk melindungi *file* citra terkompresi dari ancaman dunia digital yang sangat buruk. Algoritma RSA *(Rivest-Shamir- Adleman)* adalah salah satu algoritma kriptografi yang terkenal dan aman, dengan menggunakan pasangan kunci publik dan kunci privat untuk mengenkripsi dan mendekripsi sebuah data [5][6].

Berdasarkan penelitian sebelumnya yang dilakukan oleh Setia Nengsih Ritonga pada tahun 2022, telah berhasil mengimplementasikan algoritma golomb code dalam melakukan kompresi file video dan dapat membuktikan bahwa dari file video yang awal mula berukuran besar dapat dikompresi menjadi file yang berukuran kecil. Hal ini dapat dibuktikan berdasarkan pengujian sistem bahwa ukuran file video dapat berkurang dengan dilakukannyan kompresi, dan mengasilkan rasio rata-rata diatas 50,5% [7].

Journal Global Tecnology Computer
Vol 3, No. 3, Agustus 2024, Hal 95-104
ISSN: 2809-6118 (Online - Elektronik)
DOI: 10.47065/jogtc.v3i3.6257
Website https://ejurnal.seminar-id.com/index.php/jogtc

Lalu pada tahun 2017 dilakukan penelitian oleh Mohammad Hamdani dan Katika Putri telah berhasil dalam pengamanan file citra terkompresi. Sistem kompresi pada penelitian mampu mengkompresi dan mengirimkan data citra digital sampai ke penerima tanpa diketahui oleh pihak lain. Hal ini menguntungkan dari segi keamanan. Dan apabila terjadi penyadapan pada saat pengiriman, data kompresi tetap aman selama pihak lain tersebut tidak mengetahui kode pseudonoise yang digunakan [8].

Lalu pada tahun 2015 Agustinus Sanggar telah berhasil melakukan penelitian dalam perancangan dan implementasi sistem keamanan file citra terkompresi menggunakan algoritma AES diperoleh kesimpulan yaitu pertama, dalam menjaga kerahasiaan data dilakukan dua proses yaitu proses encode dan decode. Sistem keamanan data terkompresi dapat dikembangkan menjadi *multi user* dan *client serve*. Penambahan *Log* dalam sistem akan mempermudah dalam proses *monitoring* lokasi penyimpanan *ciphertext,* sehingga dapat lokasi penyimpanan terstruktur dan terkontrol [9].

Kemudian, pada tahun 2020 penelitian yang dilakukan oleh Rinmar Siringoringo telah berhasil mengembangkan aplikasi pengamanan file menggunakan algoritma Rijndael dan kriptografi RSA. Pada sistem yang dibangun mampu mengenkripsi dan dekripsi file yang dipilih saja. Hasil penelitian ini diperoleh sistem enkripsi dan dekripsi terhadap plainteks dan kunci simetris (sessionkey) [10].

Dan pada tahun 2020 penelitian yang dilakukan oleh Afif Malvi, dkk. Permasalahan dan pengujian aplikasi yang telah dikembangkan, maka disimpulan bahwa dengan adanya aplikasi ini file gambar dan data lainnya yang dianggap penting bagi perusahaan dapat terjaga kerahasiaannya dari pihak yang tidak bertanggung jawab. Algoritma kriptografi RSA dan steganografi EOF berhasil diimplementasikan dalam pengembangan aplikasi pengamanan file yang terdiri dari dua proses utama. Berdasarkan hasil pengujian, proses pengamanan data juga termasuk cepat, yaitu rata-rata waktu sekitar 5 detik. File gambar digital yang sudah dienkripsi dan disisipkan ke dalam file video dapat dikembalikan menjadi file asli tanpa ada perubahan [11].

2. METODOLOGI PENELITIAN

2.1 Kompresi.

Kompresi (Compression) adalah sebuah cara ilmu komputer untuk memampatkan ukuran data menjadi lebih kecil dari ukuran file aslinya dengan tujuan untuk menghemat kapasitas penyimapanan. Hal ini dilakukan agar mempermudah pengguna (User) dalam melakukan transfer data dalam jaringan. Kompresi data sangat banyak memiliki berbagai macam algoritma yang berbeda-beda yang cocok untuk berbagai macam data. Pada teknik yang digunakan untuk mengkompresi data terdapat berbagai factor yang digunakan sebagai penentu kualitas data yang dikompres, antara lain:

a. Ration of Compresion (RC) Rasio kompresi adalah menghitung kinerja dari representase data yang sudah dikompresi dan sebelum dikompresi. Secara sistematis rasio kompresi dapat di tuliskan sebagai berikut:
   $$RC = \frac{jumlah\ bit\ sebelum\ dikompresi}{jumlah\ bit\ sesudah\ dikompresi} \quad (1)$$
b. Compression Ration (CR) Compression Ration (CR) adalah presentase perbandingan antara data yang sudah dikompresi dengan data yang belum dikompres. Secara sistematis dapat dituliskan sebagai berikut:
   $$CR = \frac{jumlah\ bit\ sesudah\ dikompresi}{jumlah\ bit\ sebelum\ dikompresi}\ x100\% \quad (2)$$
c. Rendudancy (RD) Rendudancy adalah hasil penelitian rasio dengan hasil nilai rasio kompresi secara sistematis dapat ditulis dengan berikut:
   $$RD = \frac{File\ Sebelum\ Dikompres - file\ setelah\ dikompres}{Ukuran\ Data\ Sebelum\ Dikompres}\ \text{x}\ 100\% \quad (3)$$
d. Space Savings Space Savings adalah persentase dari hasil selisih antara ukuran data sebelum di kompresp dengan ukuran data sesudah dikompresi, dapat dirumuskan sebagai berikut:
   $$SS = 100\% - Cr \quad (4)$$

Pada kompresi data terdapat dua teknik dalam melakukan kompresi yaitu:

a. Kompresi Lossless Pada teknik kompresi ini terdapat informasi yang didalamnya terdapat menghasilkan citra yang sama dengan informasi pada citra awal. Citra hasil kompresi sebelumnya dapat dikembangkan secara sempurna menjadi citra asli, tidak terjadi kehilangan informasi.
b. Kompresi Lossy Kompresi data yang bersifat lossy menjajikan terjadinya kehilangan sebagian data tertentu dari pesan tersebut, sehingga dapa menghasilkan rasio kompresi yang tinggi. Apabila citra terkompresi direcontruksikan kembali maka hasilnya tidak sama dengan citra aslinya, tetapi informasi yang terkandung tidak sampai berubah atau hilang [12][13][14].

2.2 File Citra (.tiff).

File citra adalah suatu jenis berkas yang berisi data yang mewakili gambar atau grafik dalam konteks digital. Pengolahan yang dapat dilakukan terhadap citra antara lain yaitu menampilkan bentuk gambar, , melakukan perubahan terhadap gambar dan pencetakan citra digital. Format ini sangat membutuhkan ruang penyimpanan yang cukup besar oleh karena itu sangat dibutuhkan teknik kompresi pada format file TIF tanpa mengurangi resolusi pada gambar saat teknik kompresi. Format file TIFF ini sering kali dimanfaatkan untuk penyimpanan gambar yang akan digunakan untuk keperluan multimedia, medis, forografi profesional dan lainnya.

TIF atau TIFF (Tagged Image Format File) adalah TIFF adalah singkatan dari Tagged Image File Format yang

Journal Global Tecnology Computer
Vol 3, No. 3, Agustus 2024, Hal 95-104
ISSN: 2809-6118 (Online - Elektronik)
DOI: 10.47065/jogtc.v3i3.6257
Website https://ejurnal.seminar-id.com/index.php/jogtc

merupakan format yang fleksibel dan 8 normalnya menyimpan 8 bit atau 16 bit perwarna (red, green, blue) untuk total 24 bit (menggunakan ekstensi file TIFF) dan 48 bit (menggunakan ekstensi file TIF). Beberapa kamera digital menyimpan gambar dengan format TIFF dan menggunakan algoritma pemampatan LZW yang lossless. TIFF dapat menangani perangkat khusus pewarnaan, seperti CMYK yang digunakan oleh tinta percetakan secara khusus [15].

2.3 Algoritma Golomb Code.

Algoritma Golomb Code adalah metode kompresi data yang digunakan untuk mengurangi ukuran file dengan memanfaatkan kemunculan pola yang berulang dalam data. Algoritma ini pertama kali diperkenalkan oleh matematikawan Spanyol Solomon W. Golomb pada tahun 1966. Prinsip dasar dari algoritma Golomb Code adalah pengkodean data dengan menggunakan dua bagian utama: quotient (bagian hasil pembagian) dan remainder (sisa pembagian). Algoritma ini khususnya efektif untuk data yang memiliki distribusi bernilai rendah atau pola yang sering berulang [16].

Langkah-langkah untuk membangun codeword dari Golomb-Rice Code dari nilai integer positif n adalah sebagai berikut:

a. Variabel m sebagai nilai integer.
b. Veriabel q bernilai 1,2…., dan dinyatakan dalam unary code, r bernilai 0, 1, 2,…, m-1.
c. Parameter m > 0 nilai tersebut ditentukan diawal.
d. Menghitung tiga besaran q (hasil bagi), r (sisa), dan c, dengan menggunakan rumus persamaan q =/, r = n - qm, c = [{ } { }].
e. Proses encoding untuk membentuk suatu kode dilakukan dengan langkah:
   1. Buat/pisahkan bit penanda. Ini opsional dan bit menjadi bit paling signifikan.
   2. Pisahkan k (bagian akhir). Kode sisa j = i/ k bit sebagai j nol diikuti oleh 1 atau j 1 diikuti oleh 0 (mirip dengan kode unary).
   3. c = [{ } { }], r = s mod m, t = 2c – m. Output (/) dengan menggunakan kode unary.
   4. Jika r < t: Output bilangan bulat yang dikodekan dalam c 1bit paling signifikan dari r menggunakan kode biner. Maka hasil dari r + t akan menjadi output yang dikodekan dengan code biner.
f. Proses decoding dilakukan dengan langkah memecahkan kode dimulai dari ujung kiri, dikodekan dalam bentuk biner sebagai N nol diikuti oleh 1 atau N 1 diikuti oleh 0, tambahkan digit biner yang tersisa n di belakang kode unary yang telah dihasilkan.
g. Length, jumlah digit biner yang didapat dari codeword.

Dalam naskah, nomor kutipan secara berurutan dalam tanda kurung siku [3], juga tabel angka dan angka secara berurutan seperti yang ditunjukkan pada Tabel 1.

Tabel 1. String Yang Belum Dikompresi

Golomb Code	m=1 k=0	m=2 k=1	m=3	m=4 k=2	m=5	m=6	m=7	m=8 k=3
s=0	0	00	00	000	000	000	000	0000
1	10	01	010	001	001	001	0010	0001
2	110	100	011	010	010	0100	0011	0010
3	1110	101	100	011	0110	0101	0100	0011
4	11110	1100	1010	100	0111	0110	0101	0100
5	150	1101	1011	1001	1000	1000	0110	0101
6	160	11100	1100	1010	1001	1001	0111	0110
7	170	11101	11010	1011	1010	1010	1000	0111
8	180	0	11011	11000	10110	10110	10010	10000
…	…	…	…	…	…	…	…	…

2.4 Algoritma Kriptografi RSA (Rivest-Shamir-Adleman).

Metode RSA (Rivest-Shamir-Adleman) adalah sebuah algoritma kriptografi yang digunakan untuk enkripsi dan dekripsi data. Dikembangkan oleh Ron Rivest, Adi Shamir, dan Leonard Adleman pada tahun 1977, RSA menjadi salah satu metode enkripsi yang paling populer dan banyak digunakan di dunia [11]. RSA menggunakan prinsip dasar matematika teori bilangan. Berikut adalah langkah-langkah dasar dalam metode RSA:

a. Pembangkitan Kunci Pilih dua bilangan prima besar yang acak, yaitu p dan q, hitung n = p * q, yang merupakan modulus. Hitung totien ( ) = (p - 1) * (q - 1), yaitu jumlah bilangan bulat positif yang lebih kecil dari n dan relatif prima dengan n. Pilih bilangan bulat e, di mana 1 < e < totien dan e relatif prima dengan totien. e akan menjadi kunci publik. Hitung bilangan bulat d, di mana d * e ≡ 1 (mod totien). d akan menjadi kunci pribadi.
b. Enkripsi Konversi pesan atau data yang akan dienkripsi menjadi representasi numerik, misalnya dengan menggunakan skema ASCII. Untuk setiap blok data, hitung c = m^e (mod n), di mana m adalah representasi numerik dari blok data dan c adalah ciphertext yang dihasilkan.
c. Dekripsi Untuk setiap blok ciphertext c, hitung m = c^d (mod n), di mana m adalah representasi numerik dari pesan asli atau data yang telah dienkripsi:
   1. Pembangkitan kunci RSA:

Journal Global Tecnology Computer
Vol 3, No. 3, Agustus 2024, Hal 95-104
ISSN: 2809-6118 (Online - Elektronik)
DOI: 10.47065/jogtc.v3i3.6257
Website https://ejurnal.seminar-id.com/index.php/jogtc

a) Generate dua bilangan prima besar, p = 17 dan q = 11.
b) Hitung n = p * q = 187.
c) Hitung nilai phi(n) = (p - 1) * (q - 1) = 160. 11
d) Pilih sebuah bilangan e yang relatif prima dengan phi(n) dan lebih kecil dari phi(n). Misalnya, e = 7.
e) Hitung nilai d sebagai invers modulo dari e terhadap phi(n). Dalam hal ini, d = 23.
f) Kunci publik: (e, n) = (7, 187)
g) Kunci privat: (d, n) = (23, 187)

2. Enkripsi file citra
   a) Baca file citra menjadi data biner.
   b) Bagi data biner menjadi blok-blok yang sesuai. Misalnya, setiap blok memiliki panjang 8 bit.
   c) Ambil blok pertama dari data. Misalkan blok pertama adalah [11001100].
   d) Ubah blok data menjadi bilangan bulat M. Dalam hal ini, M = 204.
   e) Hitung C = M^e mod n. Dalam hal ini, C = 204^7 mod 187 = 48.
   f) Simpan C dalam file terenkripsi.
3. Dekripsi file citra
   a) Baca file terenkripsi yang berisi ciphertext C. Misalkan C = 48.
   b) Hitung M = C^d mod n. Dalam hal ini, M = 48^23 mod 187 = 204.
   c) Ubah bilangan bulat M menjadi bentuk blok data semula. Dalam hal ini, blok semula adalah [11001100].
   d) Simpan blok data semula ke dalam file citra.

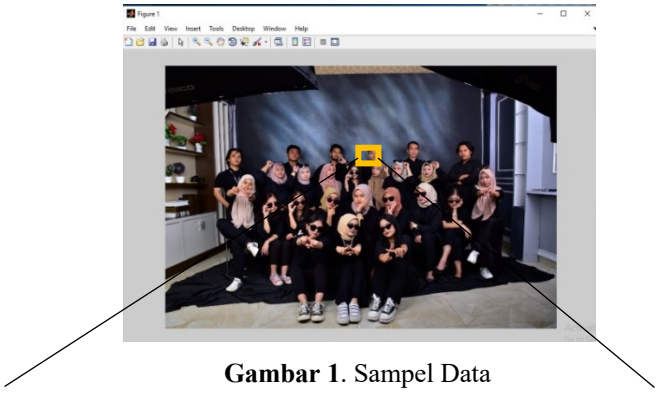
3. HASIL DAN PEMBAHASAN

Analisa pada penelitian ini dituju untuk perhitungan dan perancangan sistem kompresi dan pengamanan file citra dengan tahapan cara kerja algoritma golomb code dan algoritma kriptografi RSA ukuran kapasitas data berdasarkan karakter yang terdapat pada objek yang akan dikompresi serta pengamanan file citra terkompresi yang akan menghasilkan pesan asli yang disebut plaintext dan pesan yang telah dienkripsi disebut ciphertext. Penelitian ini akan memproses 2 tahap proses kompresi file citra yaitu proses kompresi dan proses dekompresi. Proses akan dilakukan dengan menerapkan algoritma Golomb code yang merupakan salah satu teknik kompresi lossless.

3.1 Proses Kompresi File Citra

Dalam melakukan kompresi file citra sebelumnya harus dilakukan analisa terhadap file citra yang akan dikompresi. Dalam menganalisa file citra harus dilakukan mengambilan sample file citra untuk mendapatkan nilai dari pixels-nya. Berikut adalah langkah untuk mengkompresi file citra:

a. Konversikan Citra Digital Menjadi Nilai Pixel
   Pada citra bertipe RGB, setiap pixel memiliki 3 komponen warna yaitu, merah (R), hijau (G), biru (B). Setiap warna memiliki jangkauan nilai antara 0 sampai 225 (8bit). Dari 3 warna masing-masing memiliki ukuran pixel citra 3 3 jadil total dari keseluruhan nilai Red, Green, Blue adalah 27 nilai matriks pixelnya.



Gambar 1. Sampel Data

Red			Green			Blue		
97	136	186	136	186	216	186	216	220
200	210	219	210	219	218	219	218	212
224	215	208	215	208	210	208	210	217

b. Hitung Nilai Rata-Rata Dan Deviasi Standar
   Untuk menentukan parameter dari algoritma Golomb Code, akan dilakukan pengelompokkan nilai *pixel* berdasarkan nilai *frequensi* terbanyak. Urutan karakter nilai *pixel* dapat dilihat pada tabel berikut ini:

Journal Global Tecnology Computer
Vol 3, No. 3, Agustus 2024, Hal 95-104
ISSN: 2809-6118 (Online - Elektronik)
DOI: 10.47065/jogtc.v3i3.6257
Website https://ejurnal.seminar-id.com/index.php/jogtc

Tabel 2. Pembacaan Nilai *pixel* dan Frekuensi

No	Nilai *Pixel*	Frekuensi
1	210	4
2	186	3
3	219	3
4	208	3
5	136	2
6	215	2
7	216	2
8	218	2
9	97	1
10	200	1
11	224	1
12	220	1
13	212	1
14	217	1
		Total = 27

Tahap berikutnya adalah mengurutkan nilai *pixel* berdasarkan frekuensinya, nilai frekuensi paling banyak akan diurutkan yang pertama. Pada pembahasan ini nilai biner diawali dengan berjumlah 8 bit, dapat dilihat pada tabel 3.

Tabel 3. Nilai Pixel Yang Belum Dikompresi

No	Nilai *Pixel*	*Binary*	Bit	Freq	Bit x Freq
1	210	11010011	8	4	32
2	186	10111010	8	3	24
3	219	11011011	8	3	24
4	208	11010000	8	3	24
5	136	10001000	8	2	16
6	215	11010111	8	2	16
7	216	11011000	8	2	16
8	218	11011010	8	2	16
9	97	01100001	8	1	8
10	200	11000000	8	1	8
11	224	11100000	8	1	8
12	220	11011100	8	1	8
13	212	11010100	8	1	8
14	217	11011001	8	1	8
					Total = 216

Adapun proses kompresi *file* citra dapat dilihat sebagai berikut:
Langkah-langkah untuk mencari *codeword* dengn ketentuan $m = 5$

1. $q = [\ \frac{n}{m}\ ] \Rightarrow q$ merupakan sisa hasil bagi.
2. $r = n - qm \Rightarrow r$ merupakan sisa ketika n dibagi dengan m.
3. $m \Rightarrow$ merupakan parameter.
4. $n \Rightarrow$ merupakan nomor digital yang akan dikompres.
5. Menggabungkan q dan r. Reprensentasi biner untuk q dan r kemudian digabungkan bersama untuk memberikan kode akhir dari nomor aslinya (*codeword).*

Berikut ini hasil kompresi file citra yang sudah dikompresi dengan menerapkan metode kompresi *golomb code* yang dilakukan secara manual:

Tabel 4. Hasil Kompresi menggunakan Algoritma *Golomb Code*

No	Nilai *Pixel*	Freq	Q	*q* *Binary*	r	r *Binary*	Golomb Code	Bit	Freq x Bit
1	210	4	0	0	1	01	0001	4	16
2	186	3	0	0	2	00	00100	5	15
3	219	3	0	0	3	01	00101	5	15
4	208	3	0	0	4	110	00110	5	15
5	136	2	0	0	5	111	00111	5	10
6	215	2	1	01	1	00	01000	5	10
7	216	2	1	01	2	01	01001	5	10
8	218	2	1	01	3	10	01010	5	10
9	97	1	1	01	4	11	01011	5	5
10	200	1	1	01	5	00	01100	5	5
11	224	1	2	011	1	01	01101	5	5
12	220	1	2	011	2	110	01110	5	5
13	212	1	2	011	3	111	01111	5	5

Journal Global Tecnology Computer
Vol 3, No. 3, Agustus 2024, Hal 95-104
ISSN: 2809-6118 (Online - Elektronik)
DOI: 10.47065/jogtc.v3i3.6257
Website https://ejurnal.seminar-id.com/index.php/jogtc

No	Nilai *Pixel*	Freq	Q	*q* *Binary*	r	r *Binary*	Golomb Code	Bit	Freq x Bit
14	217	1	2	011	4	0	10000	5	5
									Total = 131

Berdasarkan perhitungan yang telah dilakukan, maka untuk hasilnya dapat diterapkan menggunakan tabel *codeword* dan *binary* dari nilai *pixel* seperti berikut:

Tabel 5. Pengurutan Kode *Golomb Code*

210	186	219	208	136	215	216
0001	00100	00101	00110	00111	01000	01001
218	97	200	224	220	212	217
01010	01011	01100	01101	01110	01111	10000

Setelah kode berhasil diurutkan, maka tahap selanjutnya yaitu penyusunan kembali *string bit* yang telah dihasilkan dari proses kompresi sesuai dengan karakter. Maka penggabungan dari *string bit* yang dihasilkan sebagai berikut: "0001 00100 00101 00110 00111 01000 01001 01010 01011 01100 01101 01110 01111 10000" total keseluruhan = 69 bit.

Tabel 6. *Padding* & *Flagging*

Padding	***Flagging***
69 mod 8 = 5=n	9 - n
7-n+"1"	9 – 5 = 4
7-5+"1" =001	= *00000100*

Jadi penambahan bit dari *padding* 001 dan penambahan bit dari *flagging* 00000100, sehingga ditambahkan ke akhir *string* bit menjadi: "0001 00100 00101 00110 00111 01000 01001 01010 01011 01100 01101 01110 01111 10000 **001 00000100**". Setelah penambahan *padding* dan *flagging* dilakukan, maka total bit menjadi 80. String bit dibagi menjadi 8 biner dan ubah menjadi karakter pada tabel dibawah ini:

Tabel 7. Hasil Karakter Terkompresi

No	*Binary*	*Decimal*	*Hexadicimal*	Karakter
1	00010010	18	12	□
2	00010100	20	14	□
3	11000111	199	c7	ç
4	01000010	66	42	B
5	01010100	84	54	T
6	10110110	182	B6	¶
7	00110101	53	35	5
8	11001111	207	CF	Ï
9	10000001	129	81	□
10	00000100	4	4	

Setelah dilakukan pengkompresian dengan menerapkan algoritma *golomb code* ukuran file berkurang menjadi 131 *bit,* jika dihitung rasio kompresinya adalah:

a. *Ratio of Compression (RC)*

$$RC = \frac{Ukuran\ Data\ Sebelum\ Dikompres}{Ukuran\ Data\ Setelah\ Dikompres} \times 100\%$$

$$R_C = \frac{216}{131} = 1.64\%$$

b. *Compression Ratio (CR)*

$$C_R = \frac{Ukuran\ Data\ Setelah\ Dikompresi}{Ukuran\ Data\ Sebelum\ Dikompresi} \times 100\%$$

$$C_R = \frac{131}{216} \times 100\%$$

$$C_R = 60.64\%$$

c. *Redudancy (R_d)*

$$R_d = \frac{Ukuran\ Data\ Sebelum\ Dikompresi - Ukuran\ Data\ Setelah\ Dikompresi}{Ukuran\ Data\ Sebelum\ Dikompresi} \times 100\%$$

$$R_d = \frac{216-131}{216} \times 100\%$$

$$R_d = 39.5\%$$

d. *Space Saving (SS)*

$S_s = 100\% - C_R$

$S_s = 100\% - 60.64\%$

$S_s = 39.36\%$

3.2 Penerapan Algoritma Kriptografi RSA.

Pada penelitian ini terdapat 3 tahapan untuk proses pengamanan *file* citra yang sudah dikompresi. Proses yang akan

Journal Global Tecnology Computer
Vol 3, No. 3, Agustus 2024, Hal 95-104
ISSN: 2809-6118 (Online - Elektronik)
DOI: 10.47065/jogtc.v3i3.6257
Website https://ejurnal.seminar-id.com/index.php/jogtc

dilakukan adalah menggunakan pembentukan kunci, enskripsi dan deskripsi pada *file* citra dengan menerapkan kunci privat dan kunci public pada *file* citra terkompresi.

3.2.1 Pembentukan Kunci.

Dari sekian banyak algoritma kriptografi kunci-publik yang pernah dibuat, algoritma yangpaling popular adalah algoritma RSA. Algoritma inimelakukan pemfaktoran bilangan yang sangat besar. Oleh karena alasan tersebut RSA dianggap aman. Besaran yang digunakan pada algoritma RSA adalah :

a. *p* dan *q* bilangan prima (rahasia)
b. *n* = *p.q* (tidak rahasia)
c. *(n) = (p-1)(n-1)* (rahasia)
d. *e*(enskripsi) (tidak rahasia)
e. *d*(deskripsi) (rahasia)
f. *m(plaintext)* (rahasia)
g. *c(chipertext)* (tidak rahasia)

Pada proses pembentukan kunci, terdapat beberapa tahap pengerjaan untuk mencari pasangan kunci public dan kunci privat yaitu:

Generate Kunci

Dimana *p* dan *q* diambil dari bilangan prima sembarang *p*=11 dan *q*=13. Untuk mencari nilai *n* maka *p.q* = 11×13 hasilnya adalah *n*=143. Dimana ϕ(*n*)= (*p*-1).(*q*-1) => 10×12 =120. Nilai *e* didapatkan dari jumlah total *string bit* hasil kompresi yang berjumlah 131 dengan ketentuan GCD (131,120) = 11

Tabel 8. Ilustrasi Enkripsi

e = 131, GCD(131,121) = 11
131 mod 120 = 11
120 mod 131 = 120
11 mod 120 = 11
GCD (131,120) = 11
e=11

Kemudian, untuk mencari nilai dekripsi yaitu *d* = *e*^ mod n= $\frac{27+120}{11}$= 1

Berikut ini merupakan tabel hasil kompresi *file* citra menggunakan algoritma *golomb code*:

Tabel 9. Hasil Kompresi

No	*Binary*	*Decimal*	*Hexadicimal*	Karakter
1	00010010	18	12	□
2	00010100	20	14	□
3	11000111	199	c7	ç
4	01000010	66	42	B
5	01010100	84	54	T
6	10110110	182	B6	¶
7	00110101	53	35	5
8	11001111	207	CF	Ï
9	10000001	129	81	□
10	00000100	4	4	

3.2.2 Proses Enkripsi.

Pada proses pembangkitan kunci sebesar 13 *bit* (*n* adalah bit hasil kompresi) dimana *e* dan *d* sudah ditentukan nilai primanya yaitu *e=1* dengan menggunakan pasangan kunci *public* (1,131). Langkah selanjutnya adalah menekan tombol *generate* untuk memulai proses pembangkitan. Berikut ini adalah hasil dari proses enkripsi yang dilkukan secara manual:

Tabel 10. Enkripsi

Karakter	*Decimal* (a)	Proses Enkripsi c=x^e mod n
□	18	18^{11} mod 143= 109
□	20	20^{11} mood 143= 160
ç	199	199^{11} mod 143= 73
B	66	66^{11} mod 143= 83
T	84	84^{11} mod 143= 38
¶	182	182^{11} mod 143= 80
5	53	53^{11} mod 143= 19
Ï	207	207^{11} mod 143= 12
□	129	129^{11} mod 143= 30
	4	4^{11} mod 143= 40

Journal Global Tecnology Computer
Vol 3, No. 3, Agustus 2024, Hal 95-104
ISSN: 2809-6118 (Online - Elektronik)
DOI: 10.47065/jogtc.v3i3.6257
Website https://ejurnal.seminar-id.com/index.php/jogtc

3.2.3 Proses Dekripsi.

Pada proses pembangkitan kunci sebesar 131 *bit* (*n* adalah bit hasil kompresi) dimana *e* dan *d* sudah ditentukan nilai primanya yaitu *d*=3 dengan menggunakan pasangan kunci privat (3,131). Langkah selanjutnya adalah menekan tombol *generate* untuk memulai proses pembangkitan. Berikut ini adalah hasil dari proses dekripsi yang dilkukan secara manual.

Tabel 11. Dekripsi

Hasil Enkripsi	Proses Dekripsi y=c^d mod n	*Plainteks*
109	109^{11} mod 143= 109	m
106	106^{11} mood143=17	DC1
73	73^{11} mod 143= 73	s
83	83^{11} mod 143= 83	S
38	38^{11} mod 143= 38	&
80	80^{11} mod 143= 80	P
19	19^{11} mod 143= 19	EM
12	12^{11} mod 143= 12	FF
30	30^{11} mod 143= 30	1E
40	40^{11} mod 143= 40	©

3.3 Proses Dekompresi File Citra.

Proses dekompresi *file* citra dapat dilakukan dengan mengembalikan karakter menjadi *decimal* kemudian diubah menjadi *string bit* semula. Kemudian menganalisa karakter tersebut kedalam bentuk *biner* dan *decimal* sehingga menghasilkan *bit* hasil kompresi sebelumnya. Adapun *bit* hasil kompresi sebelumnya dapat dilihat pada tabel dibawah ini:

Tabel 12. Nilai *Decimal* dan Biner Hasil Kompresi

No	*Binary*	*Decimal*	*Hexadicimal*	Karakter
1	00010010	18	12	□
2	00010100	20	14	□
3	11000111	199	c7	ç
4	01000010	66	42	B
5	01010100	84	54	T
6	10110110	182	B6	¶
7	00110101	53	35	5
8	11001111	207	CF	Ï
9	10000001	129	81	□
10	00000100	4	4	

Berdasarkan tabel diatas, diambil nilai keseluruhan *biner* kemudian digabungkan menjadi “0001 00100 00101 00110 00111 01000 01001 01010 01011 01100 01101 01110 01111 10000”

Menghilangkan *Padding* dan *Flagging*

Mengurangi *string bit* dengan menghilangkan *padding* dan *flagging* dengan cara membaca 8 *string bit* akhir lalu ubah ke *decimal* dan dinyatakan dengan nilai n. “0001 00100 00101 00110 00111 01000 01001 01010 01011 01100 01101 01110 01111 10000 **001 00000100**”.

n = 00000100= 4

7+n = 7+4 = 11

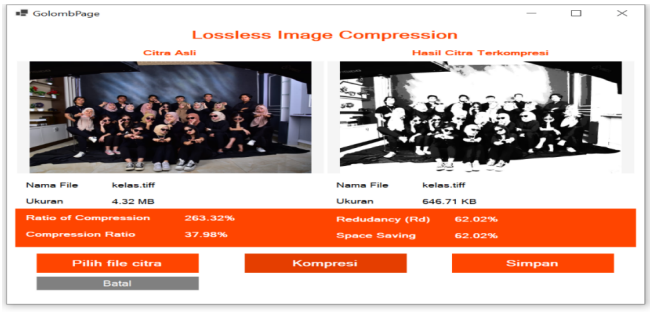
Maka langkah selanjutnya hilangkan sebanyak 11 *bit* dari *string bit* dengan pembacaan dimulai dari sebelah kanan. Sehingga menghasilkan: “0001 00100 00101 00110 00111 01000 01001 01010 01011 01100 01101 01110 01111 10000”. Dari perhitungan diatas dapat disimpulkan bahwa hasil dekompresi dari algoritma *Golomb Code* dengan membaca ulang keseluruhan *sampel* bilangan *biner* hasil dari kompresi menggunakan algoritma *Golomb Code*. Pembacaan bilangan *biner* dimuali dari awal sampai dengan bilangan *biner* membentuk sebuah nilai *biner* yang bernilai benar dan menghasilkan nilai *decimal* berdasarkan tabel kode kebenaran algoritma *Golomb Code*. Berikut adalah hasil dekompresi yang sesuai dengan *string bit* semula.

3.4 Implementasi.

Implementasi adalah tahapan pengujian dari sistem yang telah dirancang. Pada bagian ini dibahas tentang spesifikasi perangkat keras dan perangkat lunak, serta tampilan dari sistem yang sedang dijalankan.

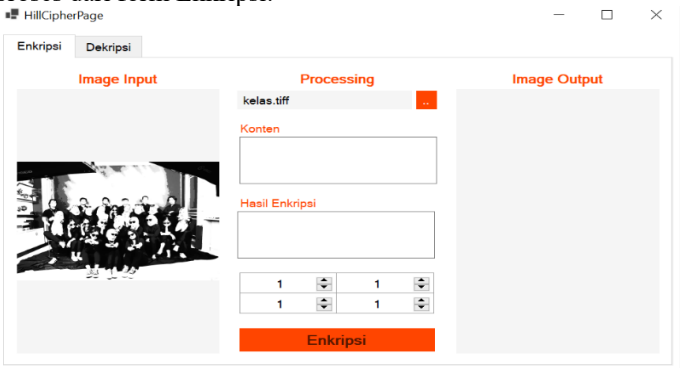
a. *Form* Kompresi
   Form ini akan menampilkan proses kompresi pada *file document* berekstensi. tiff menggunakan algoritma *Golomb-code* dan algoritma *Rivest Shamir-Adleman* (RSA). Dimana tahapan awalnya ialah inpute *file* citra berekstensi .tiff yang akan dikompresi. Setelah melakukan penginputan *file* citra yang akan dikompresi, maka aplikasi akan menampilkan gambar seperti dibawah ini:

Journal Global Tecnology Computer
Vol 3, No. 3, Agustus 2024, Hal 95-104
ISSN: 2809-6118 (Online - Elektronik)
DOI: 10.47065/jogtc.v3i3.6257
Website https://ejurnal.seminar-id.com/index.php/jogtc



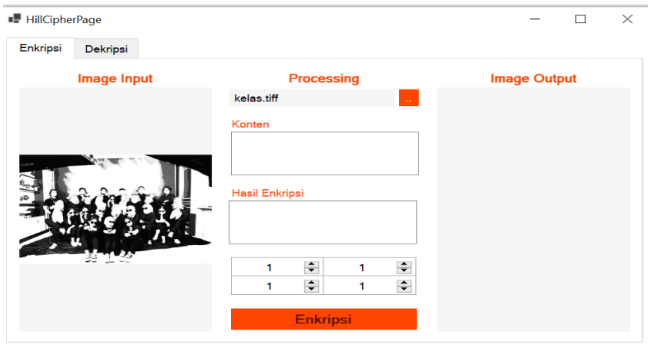
Gambar 2. Tampilan *File* Terpilih Yang Akan Dikompres

b. Form Enkripsi
Berikut ini adalah tampilan proses dari form Enkripsi:



Gambar 3. Hasil Enkripsi

c. *Form* Deskripsi
Berikut ini Tampilan output dari form deskripsi data dilihat sebagai brikut:



Gambar 4. Hasil Deskripsi

d. Hasil Pengujian

Adapun perbedaan *file* sebelum dikompresi dengan menggunakan kedua algoritma dan setelah dikompresi dengan menggunakan kedua algoritma, sebagai berikut :

Tabel 13. Tabel Hasil Pengujian

No	Sebelum Dikompresi		Sesudah Dikompresi			
	Nama *File Document*	Ukuran	Nama *File Document* Terkompresi	Ukuran	Kinerja	
1	Kelas.tiff	4.32MB	Hasil Kompresi .tiff	103.35kb	RC CR RD Ss	1.64% 60.64% 39.35% 39.36%
2	BVC.tiff	1.3 MB	Hasil Kompresi.tiff	395.36kb	RC CR Rd Ss	311.01% 32.15% 67.85% 67.85%
3	Profesionalgrafik.tiff	6.6MB	Hasil Kompresi.tiff	126.31kb	RC CR Rd Ss	856.5% 11.67% 88.33% 88.33%

Journal Global Tecnology Computer
Vol 3, No. 3, Agustus 2024, Hal 95-104
ISSN: 2809-6118 (Online - Elektronik)
DOI: 10.47065/jogtc.v3i3.6257
Website https://ejurnal.seminar-id.com/index.php/jogtc

Dari hasil pengujian diatas, setelah mendapatkan total nilai dari algoritma *golomb code* yang didapat dari parameter yang telah ditentukan maka dapat disimpulkan bahwa *file* terkompresi memiliki ukuran yang lebih kecil dibandingkan dengan *file* sebelum dikompres dengan menggunakan algoritma *golomb code* dengan total 126.31kb.

4. KESIMPULAN

Berdasarkan dari penelitian yang telah dilakukan, didapatkan hasil akhir dari penelitian ini yang dapat disimpulkan menjadi kesimpulan. Setelah mengikuti prosedur kompresi dan dekompresi dengan menggunakan algoritma *golomb code* bahwa *file* citra berformat .tiff yang memiliki ukuran citra yang cukup besar, dapat dimampatkan menjadi citra dengan ukuran yang lebih kecil. Dengan menerapkan algoritma *golomb code* maka hasil terbaik yaitu algoritma *g code* dengan hasil parameter *Ratio of Compression* (Rc) = 1.6, *Compression Ration* (Cr) = 62.5%, *Redudancy* (Rd) = 3.75%, dan *Space Saving* (Ss) = 37.5%. Algoritma kriptografi RSA adalah algoritma pengamanan paling popular dan terbukti mampu melindungi citra digital dari akses yang tidak sah termasuk *file* berformat .tiff dengan menggunakan pasangan kunci publik dan privat dengan mengenkripsi konten citra sehingga hanya pihak yang memiliki kunci privat yang tepat dapat mengaksesnya.

REFERENCES

[1] P. Fitria, "Penerapan Algoritma Rice Codes Pada Aplikasi Kompresi File Gambar," vol. 1, no. 3, pp. 158–165, 2020.
[2] S. Simanjuntak, "Implementasi Metode Taboo Code Untuk Kompresi File Video," *J. Sains dan Teknol. Inf.*, vol. 1, no. 3, pp. 79–84, 2022.
[3] Mawar, "Perancangan Aplikasi Kompresi File Pdf Dengan Menerapkan Algoritma Punctured Elias Codes Mawar," *Inf. dan Teknol. Ilm.*, vol. 7, no. 3, pp. 217–223, 2020, doi: 10.30865/komik.v6i1.5788.
[4] R. Handayani, "Perancangan Aplikasi Kompresi File Audio Menerapkan Algoritma Universal Codes," *KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer)*, vol. 5, no. 1, 2021.
[5] A. P. Wahyadyatmika, R. R. Isnanto, and M. Somantri, "Implementasi Algoritma Kriptografi RSA pada Surat Elektronik (E-Mail)," *Transient*, vol. 3, no. 4, pp. 1–9, 2014.
[6] Y. Sihura, T. Zebua, and H. Hutabarat, "Kinerja Algoritma Yamamoto's Recursive Code dan Algoritma Fixed Length Binary Encoding pada Kompresi File PDF," *J. Comput. Syst. Informatics*, vol. 3, no. 4, pp. 303–312, 2022.
[7] S. N. Ritonga, "Implementasi Algoritma Golmb Code Dalam Kompresi File Video," vol. 5, no. November, pp. 365–373, 2022, doi: 10.30865/komik.v6i1.5748.
[8] M. Hamdani and D. P. Kartikasari, "Pengamanan Pengiriman Citra Terkompresi menggunakan Metode Modulasi Direct Sequence Spread Spectrum (DS-SS)," vol. XIX, no. 2, pp. 48–59, 2017.
[9] A. Ilmiah, "Perancangan Sistem Keamanan Data Terkompresi Menggunakan Advanced Encryption Standard Algorithm pada Studio Foto GreenFrog Perancangan Sistem Keamanan Data Terkompresi Menggunakan Advanced Encryption Standard Algorithm pada Studio Foto GreenFrog," 2015.
[10] R. Siringoringo, "Analisis dan Implementasi Algoritma Rijndael (AES) dan Kriptografi RSA pada Pengamanan File," *KAKIFIKOM (Kumpulan Artik. Karya Ilm. Fak. Ilmu Komputer)*, vol. 02, no. 01, pp. 31–42, 2020, doi: 10.54367/kakifikom.v2i1.666.
[11] A. Malvi and P. Painem, "Pengamanan File Gambar pada Media Video dengan Kriptografi Algoritma RSA dan Steganografi Algoritma End of File (EOF)," *Inform. J. Ilmu Komput.*, vol. 16, no. 2, p. 67, 2020, doi: 10.52958/iftk.v16i2.1860.
[12] P. Algoritma, B. C. Untuk, and K. F. Citra, "Penerapan algoritma boldi-vigna code untuk kompresi file citra," 2022.
[13] N. Aisyah and S. Aripin, "Penerapan Algoritma Elias Omega Code Pada Kompresi File Audio Aplikasi Murottal Muzzamil Hasbalah," *Pelita Inform. Inf. dan Inform.*, vol. 9, no. 2, pp. 113–119, 2020.
[14] E. Siahaan, "Implementasi Algoritma Elias Gamma Code Untuk Kompresi File Teks Pada Aplikasi Watpad," *Resolusi Rekayasa Tek. Inform. dan Inf.*, vol. 2, no. 2, pp. 76–84, 2021.
[15] A. Nur, H. Yuana, and F. Febrinita, "APLIKASI KOMPRESI CITRA DENGAN MENGGUNAKAN ALGORITMA LEMPEL ZIV WELCH (LZW)," *JATI (Jurnal Mhs. Tek. Inform.*, vol. 6, no. 2, 2022, doi: 10.36040/jati.v6i2.5612.
[16] E. K. Gulo, "Perancangan Aplikasi Kompresi Audio dengan Menerapkan Algoritma Golomb," *Pelita Inform. Inf. dan Inform.*, vol. 6, no. 2, pp. 241–244, 2017.