Jurnal Sistem Informasi, Teknik Komputer dan Teknologi Pendidikan (JUSTIKPEN)
Volume 3, Nomor 2, Maret, 2024, Page 71-81
ISSN 2828- 7921 (media online)

Implementasi Algoritma Lucifer Dengan Pembangkit Kunci *Multiply With Carry Generator* Untuk Mengamankan Data Citra Digital

Handayani Simanjuntak

Teknik Informatika, Universitas Budi Darma, Medan, Indonesia
Email : handayanisimanjuntak0508@gmail.com

Abstrak− Untuk mengetahui kekuatan dari suatu algoritma dalam pengamanan data tidak hanya tergantung dari algoritma yang digunakan, tetapi diperlukan suatu kunci yang memiliki kerumitan dalam pemecahannya. Algoritma yang digunakan dalam mengamankan data dalam penelitian ini adalah algoritma *lucifer*. Dalam penerapan algoritma *lucifer* jumlah karakter kunci sama dengan jumlah karakter teks ataupun data yang ingin diamankan yaitu sebanyak 16 karakter. Sehingga dalam penggunaan jumlah kunci sebanyak 16 karakter memungkinkan terjadinya perulangan bilangan atau karakter yang sama sehingga mudah bagi penyerang untuk mengetahi pola dalam pembangkitan kunci. Berdasarkan masalah diatas maka untuk mengamankan data sangat diperlukan suatu algoritma yang memiliki pembangkitan kunci yang acak dan rumit. *Multiply Wiyh Carry Generator* adalah algoritma pembangkitan bilangan acak yang semu dengan proses yang cepat dan akan berhenti pada nilai batas akhir pengacakan. Hasil dari penelitian ini, algoritma *multiply wiyh carry generator* sangat optimal dalam pengacakan kunci karena karakter bilangan yang dihasilkan sangat sulit diketahui polanya dan sangat minim perulangan bilangan yang sama, dalam pengimplementasian algoritma *lucifer* dengan pembangkit kunci *multiply wiyh carry generator* dalam mengamankan citra digital menghasilkan nilai *pixel* yang berbeda sehingga sulit bagi penyerang untuk memecahkannya.

Kata Kunci: *Kriptografi, Lucifer*, Multiply With Carry Generator, Citra Digital

Abstract− To know the strength of an algorithm in securing data does not only depend on the algorithm used, but requires a key that is complex in solving. The algorithm used to secure data in this research is the Lucifer algorithm. In implementing the Lucifer algorithm, the number of key characters is the same as the number of text or data characters that you want to secure, namely 16 characters. So, using a key of 16 characters allows repetition of the same number or character, making it easy for attackers to understand the pattern in generating the key. Based on the problem above, to secure data, an algorithm is needed that has random and complicated key generation. Multiply Wiyh Carry Generator is a pseudo random number generation algorithm with a fast process and will stop at the final randomization limit value. The results of this research, the multiply wiyh carry generator algorithm is very optimal in randomizing keys because the pattern of the numbers produced is very difficult to know and there is very minimal repetition of the same number, in implementing the lucifer algorithm with the multiply wiyh carry generator key generator in securing digital images it produces pixel values different ones so that it is difficult for attackers to break them.

Keywords: Cryptography, Lucifer, Multiply With Carry Generator, Digital Image

1. PENDAHULUAN

Kriptografi adalah ilmu yang berperan penting dalam pengamanan data atau teknik yang berhubungan dengan cara mengamankan pesan atau data sehingga dapat dimengerti oleh pihak-pihak yang memiliki hak akses atau kunci rahasia tertentu[1]**.** Kriptografi berasal dari bahasa Yunani, yang terdiri dari dua buah kata yaitu *cryptos* dan *graphia*[2]. Citra digital adalah gabungan dari beberapa titik, garis dan warna yang dapat menghasilkan sebuah objek sehingga dapat dijadikan oleh manusia dalam penyampaian informasi[3] [4]. Citra digital adalah salah satu jenis data yang banyak digunakan dalam berkomunikasi baik secara langsung maupun melalui media internet. Perkembangan sistem informasi yang semakin maju saat ini data citra sering dipertukarkan, diunggah dan dibagikan secara luas oleh pengguna. Sehingga ada beberapa data citra yang bersifat pribadi dan rahasia dan rentang terhadap pemalsuan data citra yang penting untuk diamankan. Permasalahan yang sering terjadi saat ini adalah penyalahgunaan data oleh para peretas yang mengakses data pribadi secara sembarangan tanpa ijin dari pemilik data.

Berdasarkan masalah diatas maka diperlukan suatu teknik untuk mengamankan data yaitu dengan cara mengenkripsikan data tersebut sehingga sulit bagi peretas untuk melihat keaslian data

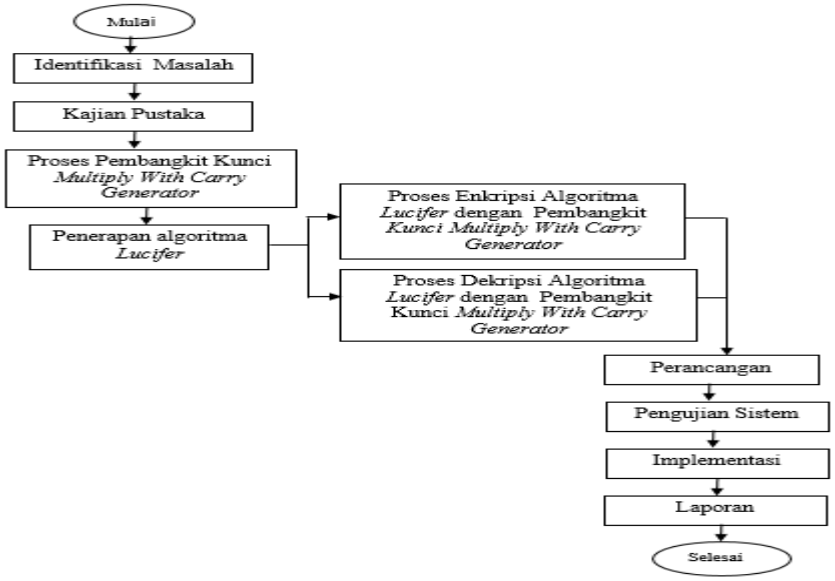
Submitted: **27/02/2024**; Accepted: **20/03/2024**; Published: **31/03/2024**

citra yang sudah dienkripsikan. Namun tingkat keamanan dalam metode kriptografi terletak pada kunci yang digunakan sehingga diperlukan upaya untuk meningkatkan keacakan karakter kunci yang tinggi dan meminimalkan perulangan bilangan atau huruf yang sama, karena semakin sulit pengacakan kunci maka akan meminimalkan kebocoran data[5]. Teknik kriptografi yang digunakan dalam permasalahan ini yaitu pengimplementasian algoritma lucifer dan pembangkitan kunci *Multiply With Carrry Generator* (MWCG). Algoritma *lucifer* merupakan *block cipher*, dimana jumlah kunci sama dengan jumlah *plain* yaitu 128 bit *plain* atau sama dengan 16 karakter, dan menggunakan 128 bit *key*, melakukan proses enkripsi dan dekripsi sebanyak 16 *round*. Sehingga algoritma *lucifer* merupakan algoritma yang memiliki banyak varian[6]. *Multiply With Carry Generator* adalah algoritma yang dapat digunakan dalam membangkitkan sebuah bilangan acak, *multiply with carry generator* dapat disembunyikan dengan proses yang cepat dan akan berhenti bila telah sampai pada nilai batas akhir pengacakan[7].

Penelitian terkait yang dilakukan oleh S N Hutagalung pada tahun 2022 tentang pengimplementasian algoritma *lucifer* bahwa mengamankan data dengan algoritma *lucfer* efektif [8]. D Anggara, A S Sembiring tahun 2019 yaitu meningkatkan keamanan data teks pada citra digital dengan algoritma *lucifer* menggunakan *steganografi gifshuffle* dengan ukuran sampel 10 x 13 *pixel* mengatakan bahwa tidak ada perbedaan yang signifikan terhadap citra asli yang telah dilakukan penyisipan pesan[6].. B K Laila tahun 2022 penerapan pembangkit kunci *naïve shuffle* untuk mengamankan pesan dengan algoritma *lucifer* menyimpulkan bahwa keacakan kunci dengan algoritma tersebut masih kurang acak sehingga diperlukan penambahan proses pengacakan kunci[9]. R A Saputra, I Awalda Tariza dan B Pramono tahun 2022 penerapan algoritma *multiply with carry generator* dalam pengacakan soal *run test* yang dilakukan pada Jumlah keseluruhan soal adalah 1.200 soal. Soal terdiri dari 3 kategori soal yaitu mudah, menengah, dan sulit. Berdasarkan pengujian menunjukkan 10 siswa hasil dari $H0$ adalah diterima. Dibuktikan dengan nilai 10 siswa dapat memenuhi syarat *run test* yaitu $ra \leq r \leq rb$ dalam arti 10 siswa mendapatkan 20 nomor soal acak[7]. Penerapan MWCG pada proses pembangkitan kunci algoritma Beaufort Cipher oleh T Zebua pada tahun 2023 mengatakan bahwa penerapan algoritma MWCG sangat mendukung ketahanan kunci pada saat pengujian *run test* diketahui bahwa penyebaran angka acak yang dihasilkan tidak memperlihatkan pola kunci sehingga walaupun ada bilangan yang muncul 2 kali seperti bilangan 42,27 dan 39 tidap dapat diprediksi kemunculannya[10].

2. METODOLOGI PENELITIAN

Metodologi penelitian ini menjabarkan tentang tahapan-tahapan dalam melakukan penelitian yang saling berkaitan secara sistematis.



Gambar 1: Kerangka kerja penelitian

Submitted: **27/02/2024**; Accepted: **20/03/2024**; Published: **31/03/2024**

Kriptografi berasal dari bahasa Yunani, yang terdiri dari dua buah kata yaitu *cryptos* dan *graphia*. *Cryptos* artinya tersembunyi dan *graphia* berarti tulisan. Kriptografi merupakan tulisan yang di rahasiakan atau goresan pena yang tersembunyi, kriptografi juga dikatakan sebagai seni yang mempelajari informasi dan pelajaran rumus dalam matematika yang fokus pada teknik dalam menyembunyikan data sebelum dikirim ke penerima[2].

2.1 Lucifer

Algoritma *lucifer* adalah *block cipher* pertama. Dan pertama sekali di kembangkan oleh Feistel Chiper yang dijadikan sebagai dasar algoritma *Data Encryption Standard* (DES). Algoritma *lucifer* mengenkripsi 128 bit *plaintext* dan menggunakan kunci sebanyak 128 bit juga[9].

Berikut ini adalah langkah-langkah dalam menyelesaikan proses enkripsi algoritma *lucifer*:

1. *Plaintext* 128 bit akan dibagi menjadi dua *block* yaitu 64 bit menjadi *left bit* (L0) dan 64 bit menjadi *right bit* (R0).
2. Melakukan formulasi enkripsi, untuk mencari nilai L1 yaitu:
$$L_{i+1} = R_i \quad \ldots (2.1)$$
3. Untuk mencari nilai R1
$$R_{i+1} = L_i \oplus f\left(R_i , K_i\right) \quad \ldots (2.2)$$
4. Setelah setiap putaran dilakukan kecuali yang terakhir, bagian kiri dan kanan *block* di tukar.

Berikut ini adalah langkah-langkah dalam menyelesaikan proses dekripsi algoritma *lucifer*:

1. *Ciphertext* dibagi menjadi dua bagian yaitu *Right bit* (Rn) dan *Left bit* (Ln).
2. Untuk mencari nilai L yaitu
$$L_{i-1} = R_i \oplus F\left(L_i , K_i\right) \quad \ldots (2.3)$$
3. Untuk mencari nilai R yaitu
$$R_{i-1} = L_i \quad \ldots (2.4)$$
4. Setelah setiap putaran dilakukan kecuali yang terakhir, bagian kiri dan kanan *block* di tukar.

2.2 Multiply *With Carry Generator*

George Marsaglia merupakan orang yang mengembangkan algoritma MWCG. Bilangan acak dapat melakukan proses pembangkitan berdasarkan algoritma pengacakan bilangan acak. MWCG. merupakan algoritma pembangkit bilangan acak yang semu dan dapat disembunyikan dan dilakukan dengan cepat[7].

Berikut merupakan formula dalam menghasilkan bilangan acak :

$$P_n = \left(q\, P_{n-1} + R_{n-1}\right) Mod\ M$$
$$R_n = \left[\frac{qP_{n-1} + R_{n-1}}{M}\right]$$

Keterangan:
P_n = Bilangan acak ke-n (P_1)
P_{n-1} = Bilangan acak ke n-1 ($P_{1-1} = P_0$)
q = konstanta penggali
R = Konstanta kenaikan ke-n (R_1)
R_{n-1} = Konstanta kenaikan ke n-1 ($R_{1-1} = R_0$)
M = konstanta pembagi

Syarat yang harus digunakan dalam menghasilakan siklus yang panjang dalam melakukan pengacakan algoritma MWCG yaitu:

1. $R_0 < M$
2. qM – 1 adalah bilangan prima.

Penerapan pembangkit kunci *multiply with carry generator* dengan plaintext dan kunci sebanyak 16 *words*. Dalam penerapan MWCG akan menghasilkan *key* sebanyak 16 *words* juga, bila hasil proses perhitungan desimal maka akan dilakukan pembulatan.

3. HASIL DAN PEMBAHASAN

Submitted: **27/02/2024**; Accepted: **20/03/2024**; Published: **31/03/2024**

Jurnal Sistem Informasi, Teknik Komputer dan Teknologi Pendidikan (JUSTIKPEN)
Volume 3, Nomor 2, Maret, 2024, Page 71-81
ISSN 2828- 7921 (media online)

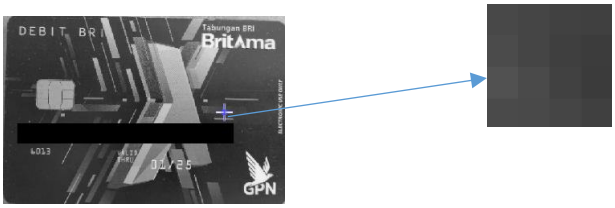
Mengamankan data citra digital dengan algoritma *lucifer* membutuhkan suatu kunci yang mampu menghasilkan bilangan yang acak dan mengurangi perulangan bilangan. Algoritma yang digunakan dalam penelitian ini adalah *multiply with carry generator* dimana algoritma ini mampu melakukan perulangan bilangan yang minim.

3.1 Penerapan Algoritma Lucifer

Berikut adalah proses untuk mengamankan citra digital berdasarkan algoritma lucifer:

a. Pembangkitan kunci algoritma lucifer

Dalam proses pengamanan data maka yang dilakukan terlebih dahulu yaitu penginputan data nilai *pixel* dengann ukuran 4 x 4 sebagai sampel *plainimage* sebanyak 16 karakter atau sama dengan 128 bit. Selanjutnya melakukan pembangkitan kunci, lalu nilai *pixel* diubah ke biner kemudian di enkripsi dengan *key* yang telah di tentukan dan menghasilkan *cipherimage*. Berikut adalah sampel data yang ingin diamankan.



Gambar 2. Sampel Gambar 4 x 4

Adapun hasil proses ekstraksi setiap *pixel* pada *software* matlab adalah sebagai berikut:

72	69	65	64
75	69	63	60
81	74	65	59
74	73	69	64

Gambar 3. Hasil Nilai Setiap *Pixel* Resolusi 4x4

Berdasarkan gambar 4.6, nilai *pixel* yang dihasilkan berdasarkan aplikasi matlab. Berikut adalah nilai *pixel grayscale* resolusi 4x4 yang dihasilkan: 72, 69, 65, 64, 75, 69, 63, 60, 81, 74, 65, 59, 74, 73, 69, 64.

b. Pembangkitan kunci berdasarkan algoritma multiply with carry generator

Penerapan algoritma *lucifer* untuk mengamankan data citra digital. Berdasarkan batasan masalah yang dienkripsi ialah *plainimage* sebanyak 128 bit dan kunci 128 bit.

Nilai *pixel* karakter : 72 69 65 64 75 69 63 60 81 74 65 59 74 73 69 64

Berdasarkan syarat yang telah ditentukan maka peneliti memberikan nilai untuk $R_0 = 23$, $M = 400$, $q = 38$, $P_0 = 42$. Proses pembangkitan bilangan acak yang akan digunakan sebagai kunci dalam penelitian ini akan dilakukan sesuai dengan jumlah *plainimage* maka bilangan yang akan dibangkitkan sebanyak 16 karakter. sehingga nilai untuk setiap variabel di atas yaitu:

Maka proses perhitungannya adalah sebagai berikut:

Kunci 1

$$P_1 = \left(\left(q * p_0\right) + R_0\right) Mod\ M$$

$$P_1 = \left((38 * 42) + 23\right) Mod\ 400$$

$$P_1 = 1.619\ Mod\ 400$$

$P_1 = 19$, sehingga untuk langkah selanjutnya maka nilai P yang di masukkan adalah nilai P yang telah dihasilkan sebelumnya, misalnya untuk proses perhitungan P_2 nilai yang dimasukkan untuk proses perhitungannya adalah nilai P_1.

$$R_1 = \left[\frac{q * P_0 + R_0}{M}\right]$$

$$R_1 = \left[\frac{38 * 42 + 23}{400}\right]$$

$R_1 = 4$, maka nilai kunci untuk karakter pertama untuk *plainimage* dan *Cipherimage* adalah 4

Kunci ke 2

Submitted: **27/02/2024**; Accepted: **20/03/2024**; Published: **31/03/2024**

Jurnal Sistem Informasi, Teknik Komputer dan Teknologi Pendidikan (JUSTIKPEN)
Volume 3, Nomor 2, Maret, 2024, Page 71-81
ISSN 2828- 7921 (media online)

$P_2 = \left(\left(q * p_1\right) + R_1\right) Mod\, M$

$P_2 = \left((38 * 19) + 4\right) Mod\, 400$

$P_2 = 326\ Mod\, 400$

$P_2 = 326$, maka untuk mencari nilai di P_3 nilai yang akan di masukkan adalah 326 berdasarkan nilai P_2.

$R_2 = \left[\frac{q * P_1 + R_1}{M}\right]$

$R_2 = \left[\frac{38 * 19 + 4}{400}\right]$

$R_2 = 2$, sehingga nilai kunci untuk karakter ke dua *plainimage* dan *Cipherimage* adalah 2.

Proses pembangkitan kunci ini di lakukan sampai R_{16} dengan cara yang sama. Sehingga menghasilkan 16 karakter bilangan acak yang dijadikan sebagai kunci untuk mengamankan data. Berikut adalah bilangan yang dihasilkan 4 2 31 37 5 36 24 12 1 6 18 22 18 28 23 26.

c. Proses enkripsi algoritma lucifer

Langkah pertama yang harus dilakukan dalam proses enkripsi adalah melakukan penjadwalan kunci.

Tabel 1. Konversi Kunci ke Biner

Char	Biner	Key Byte Schedule
4	00000100	0
2	00000010	1
31	00011111	2
37	00100101	3
5	00000101	4
36	00100100	5
24	00011000	6
12	00001100	7
1	00000001	8
6	00000110	9
18	00010010	10
22	00010110	11
18	00010010	12
28	00011100	13
23	00010111	14
26	00011010	15

Setelah kunci diubah ke bilangan biner, maka selanjutnya adalah mengatur penjadwalan kunci dengan panjang *message byte* maksimal 8, yang dilakukan untuk setiap round dalam proses enkripsi dan dekripsi.

MESSAGE BYTE

C-I-D ROUND	0	1	2	3	4	5	6	7
1	0	1	2	3	4	5	6	7
2	7	8	9	10	11	12	13	14
3	14	15	0	1	2	3	4	5
4	5	6	7	8	9	10	11	12
5	12	13	14	15	0	1	2	3
6	3	4	5	6	7	8	9	10
7	10	11	12	13	14	15	0	1
8	1	2	3	4	5	6	7	8
9	8	9	10	11	12	13	14	15
10	15	0	1	2	3	4	5	6
11	6	7	8	9	10	11	12	13
12	13	14	15	0	1	2	3	4
13	4	5	6	7	8	9	10	11
14	11	12	13	14	15	0	1	2
15	2	3	4	5	6	7	8	9
16	9	10	11	12	13	14	15	0

Gambar 3. Tabel message Byte

Submitted: **27/02/2024**; Accepted: **20/03/2024**; Published: **31/03/2024**

Jurnal Sistem Informasi, Teknik Komputer dan Teknologi Pendidikan (JUSTIKPEN)
Volume 3, Nomor 2, Maret, 2024, Page 71-81
ISSN 2828- 7921 (media online)

Berdasarkan tabel message byte maka hasil penjadwalan kunci untuk algoritma lucifer ditunjukkan pada tabel 2.

Tabel 2. Tabel Message Byte

	0	1	2	3	4	5	6	7
1	00000100	00000010	00011111	00100101	00000101	00100100	00011000	00001100
2	00001100	00000001	00000110	00010010	00010110	00010010	00011100	00010111
3	00010111	00011010	00000100	00000010	00011111	00100101	00000101	00100100
4	00100100	00011000	00001100	00000001	00000110	00010010	00010110	00010010
5	00010010	00011100	00010111	00011010	00000100	00000010	00011111	00100101
6	00100101	00000101	00100100	00011000	00001100	00000001	00000110	00010010
7	00010010	00010110	00010010	00011100	00010111	00011010	00000100	00000010
8	00000010	00011111	00100101	00000101	00100100	00011000	00001100	00000001
9	00000001	00000110	00010010	00010110	00010010	00011100	00010111	00011010
10	00011010	00000100	00000010	00011111	00100101	00000101	00100100	00011000
11	00011000	00001100	00000001	00000110	00010010	00010110	00010010	00011100
12	00011100	00010111	00011010	00000100	00000010	00011111	00100101	00000101
13	00000101	00100100	00011000	00001100	00000001	00000110	00010010	00010110
14	00010110	00010010	00011100	00010111	00011010	00000100	00000010	00011111
15	00011111	00100101	00000101	00100100	00011000	00001100	00000001	00000110
16	00000110	00010010	00010110	00010010	00011100	00010111	00011010	00000100

Berdasarkan tabel di atas, maka diketahui *subkey* yang akan digunakan untuk melakukan proses enkripsi dan dekripsi pada setiap *round* adalah sebagai berikut :

$K_1 =$ 000001000000001000011111001001010000010100100100000110000000 1100

$K_2 =$ 000011000000000100000110000100100001011000010010000111000001 0111

$K_3 =$ 000101110001101000000100000000100001111100100101000001010010 0100

$K_4 =$ 001001000011000000011000000000100000110000100100001011000010 010

$K_5 =$ 000100100001110000010111000110100000100000000100001111100100 101

$K_6 =$ 001001010000010100100100000110000000110000000001000001100001 0010

$K_7 =$ 000100100001011000010010000111000001011100011010000001000000 0010

$K_8 =$ 000000100001111100100101000001010010010000011000000011000000 0001

$K_9 =$ 000000010000011000010010000101100001001000011100000101110001 1010

$K_{10} =$ 000110100000010000000010000111110010010100000101001001000001 1000

$K_{11} =$ 000110000000110000000001000001100001001000010110000100100001 1100

$K_{12} =$ 000111000001011100011010000001000000001000011111100100101000 0101

$K_{13} =$ 000001010010010000011000000011000000000100000110000100100001 0110

$K_{14} =$ 000101100001001000011100000101110001101000000100000000100001 1111

$K_{15} =$ 000111111001001010000010100100100000011000000001100000000001 00000110

$K_{16} =$ 000001100001001000010110000100100001110000010111000110100000 0100

Berikut ini adalah proses pengenkripsian algoritma *Lucifer* :

Plainimage :72 69 65 64 75 69 63 60 81 74 65 59 74 73 69 64

Kemudian dikonversi dalam bentuk biner seperti di bawah ini.

Tabel 3 Konversi Nilai *Plainimage* Ke Biner

Konversi nilai pixel ke biner					
72		01001000	81		01010001
69		01000101	74		01001010
65		01000001	65		01000001
64	=	01000000	59	=	00111011
75		01001011	74		01001010
69		01000101	73		01001001
63		00111111	69		01000101

Submitted: **27/02/2024**; Accepted: **20/03/2024**; Published: **31/03/2024**

60	00111100	64	01000000

$L_0 = 01001000010001010100000101000000010010110100010100111111100111100$

$R_0 = 01010001010010100100000100111011010010100100100101000101010000000$

Putaran 1

$L_1 = R_0\ maka$

$L_1 = 01010001010010100100000100111011010010100100100101000101010000000$

$R_1 = L_0 F(R_0, K_1)$

01010001010010100100000100111011010010100100100101000101010000000

00000100000000100001111100100101000001010010010000011000000001100⊕

01010101010010000101111000011110010011110111010101110101001100

01001000010001010100000101000000010010110100010100111111001111100⊕

R_1 =00011101000011010001111101011110000001000010100001100010011100000

Putaran 2

$L_2 = R_1\ maka$

$L_2 = 00011101000011010001111101011110000001000010100001100010011100000$

$R_2 = L_1 F(R_1, K_2)$

00011101000011010001111101011110000001000010100001100010011100000

00001100000000010000011000010010000101100001001000011100000010111⊕

00010001000011000001100101001100000100100011101001111110011001111

01010001010010100100000100111011010010100100100101000101010000000⊕

R_2 =01000000010001100101100001110111010110000011100110011101100100111

Setelah dilakukan pemutaran sampai 16 kali, untuk melihat *cipherimage* yang harus dilakukan yaitu menggabungkan hasil dari R_{16} dan L_{16}, kemudian dikelompokkan menjadi delapan bit perkelompok seperti yang dibawah ini, *cipherimage* nilai *pixel* dalam bentuk biner:

00000010000110100010000001101110001101100001111101101100010001010111001101111111010001010000101101101111011110000011010100101111

Nilai *cipherimage pixel* setelah diubah *kedecimal* berdasarkan tabel ASCII: : 2, 26, 32, 110, 54, 31, 108, 69, 115, 127, 69, 11, 111, 120, 106, 95.

Berdasarkan hasil enkripsi yang telah didapatkan, maka warna *pixel* yang dihasilkan adalah:

Gambar 4. Hasil Enkripsi Nilai *Pixel* Citra

d. Proses dekripsi

Proses Dekripsi algoritma *lucifer* merupakan kebalikan dari proses enkripsi. Dimana pada proses pendekripsiannya awal *block* dibagi menjadi dua yaitu *left block* dan *right block*. Berikut adalah proses dekripsi algoritma *lucifer*:

Cipherimage : 2, 26, 32, 110, 54, 31, 108, 69, 115, 127, 69, 11, 111, 120, 106, 95.

Langkah pertama yang akan dilakukan adalah mengubah nilai *cipherimage pixel* karakter menjadi biner.

Tabel 4. Konversi Nilai *Cipherimage* ke Biner

Konversi Nilai *Cipherimage* ke Biner					
2		00000010	115		01110011
26		00011010	127		01111111
32		00100000	69		01000101
110	=	01101110	11	=	00001011
54		00110110	111		01101111
31		00011111	120		01111000
108		01101100	106		01101010
69		01000101	95		01011111

Submitted: **27/02/2024**; Accepted: **20/03/2024**; Published: **31/03/2024**

Jurnal Sistem Informasi, Teknik Komputer dan Teknologi Pendidikan (JUSTIKPEN)
Volume 3, Nomor 2, Maret, 2024, Page 71-81
ISSN 2828- 7921 (media online)

Berdasarkan tabel diatas maka langkah selanjutnya adalah membagi dua *block* awal menjadi *left block* (L) dan *right block* (R).

00000010000110100010000001101110001101100001111101101100010001010111000100000110100010000001101110001101100001111101101100010001010111

01110011011111111010001010000101101101111011110000011010100101111101110011011111111010001010000101101101111011110000011010100101111101110011

R_{16}= 00000010000110100010000001101110001101100001111101101100010001010111000100000110100010000001101110001101100001111101101100010001010111

L_{16}= 01110011011111111010001010000101101101111011110000011010100101111101110011011111111010001010000101101101111011110000011010100101111101110011

Putaran 16

$$R_{15} = L_{16}\ maka$$

R_{15} = 01110011011111111010001010000101101101111011110000011010100101111101110011011111111010001010000101101101111011110000011010100101111101110011

$$L_{15} = R_{16}\,F(L_{16}, K_{16})$$

01110011011111111010001010000101101101111011110000011010100101111101110011011111111010001010000101101101111011110000011010100101111101110011

00000110000100100001011000010010000111000001011100011010000001000⊕

01110101011011010101001100011001011100110110111101110000010110110

00000010000110100010000001101110001101100001111101101100010001010111⊕

L_{15} = 01110111011101110111001101110111010001010111000000011100000111100

Putaran 15

$R_{14} = L_{15}\ maka$

R_{14} = 01110111011101110111001101110111010001010111000000011100000111100

$L_{15} = R_{15}\,F(L_{15}, K_{15})$

01110111011101110111001101110111010001010111000000011100000111100

00011111001001010000010100100100000110000000110000000001000001100⊕

01101000010100100111011001010011010111010111110000011101000110000

01110011011111111010001010000101101101111011110000011010100101111111⊕

L_{14} = 00011011001011010011001101011000001100100000010001110111010000111

Untuk melihat nilai plain *pixel* maka dilakukan penggabungan dari L_0 dan R_0. Kemudian dikelompokkan menjadi delapan bit, berikut hasil penggabungan antara nilai L_0 dan R_0.

01001000010001010100000101000000010010110100010100111111001111000

01010001010010100100000100111011010010100100100101000101010000000

Sehingga nilai *pixel* dalam bentuk desimal menjadi : 72, 69, 65, 64, 75, 69, 63, 60, 81, 74, 65, 59, 74, 75, 69, 64.

3.2 Implementasi

Tampilan sistem merupakan tampilan akhir antar muka dari sistem yang dirancang. Berdasarkan perancangan sistem tersebut, berikut adalah hasil pengimplementasian pengamanan gambar dengan algoritma *lucifer*:

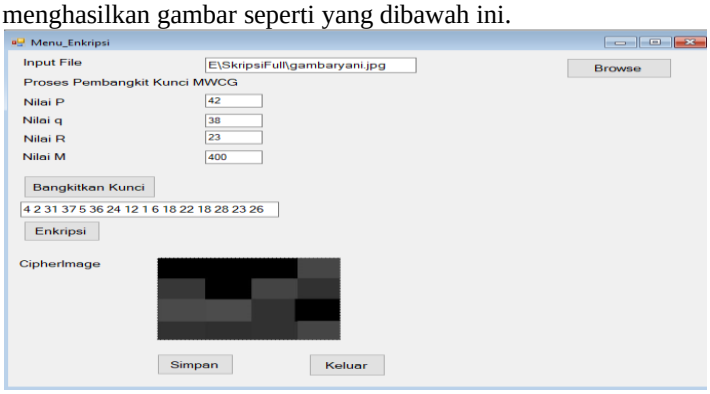
1. *Form* Menu Utama
   Form menu utama merupakan tampilan awal saat menjalankan program. Pada tampilan ini terdapat beberapa menu yang berfungsi untuk mengakses *form-form* pada sistem ini. Berikut adalah tampilan *form* menu utama:

Submitted: **27/02/2024**; Accepted: **20/03/2024**; Published: **31/03/2024**

Jurnal Sistem Informasi, Teknik Komputer dan Teknologi Pendidikan (JUSTIKPEN)
Volume 3, Nomor 2, Maret, 2024, Page 71-81
ISSN 2828- 7921 (media online)

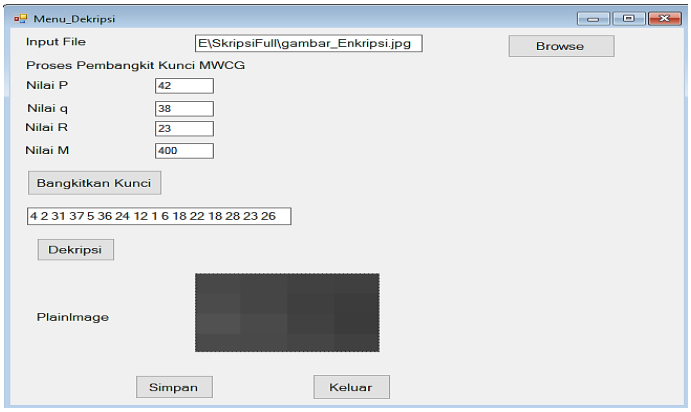
Gambar 5. Form Menu Utama

2. Tampilan Output Enkripsi
   Proses yang akan dilakukan ketika *user* ingin melakukan enkripsi adalah memilih *file* gambar yang akan dienkripsi kemudian melakukan pembangkitan kunci dengan cara mengisi nilai variabel untuk setiap *textbox* yang telah disediakan, dan selanjutnya adalah melakukan proses enkrispi dan menghasilkan gambar seperti yang dibawah ini.



Gambar 6. Tampilan Output Enkripi

3. Tampilan Output Dekripsi
   Proses yang dilakukan dalam pendekripsian gambar adalah memilih *file* gambar yang telah di enkripsi kemudian melakukan pembangkitan kunci, kunci yang digunakan sama dengan kunci proses enkripsi. Selanjutnya adalah memilih tombol dekripsi, maka akan muncil seperti gambar berikut:



Gambar 7. Form Menu Dekripsi

Submitted: **27/02/2024**; Accepted: **20/03/2024**; Published: **31/03/2024**

4. Form About Me

Form about me merupakan *form* yang digunakan untuk menampilkan informasi tentang penulis.

Gambar 8. Form Tentang About Me

4. KESIMPULAN

Kesimpulan dari hasil dan pembahasan dalam penelitian ini antara lain, Kemunculan bilangan dengan nilai yang sama sangat minim sehingga algoritma *multiply with carry generator* dapat digunakan secara optimal dalam melakukan pembangkitan sebuah kunci dalam mengamankan data sehingga sulit bagi penyerang untuk mengetahui pola dalam pembangkitan kunci.

Citra digital dengan format .jpg dapat diterapkan dengan teknik kriptografi algoritma *lucifer*, nilai *pixel* yang dihasilkan berbeda dengan nilai *pixel* awal sehingga menghasilkan sebuah *cipherimage* yang berbeda dengan *plainimage* awal.

Sistem yang dirancang untuk mengamankan data citra digital menggunakan *microsoft visual studio* 2008 berdasarkan proses enkripsi dan dekripsi algoritma *lucifer* dapat mempermudah *user* dalam melakukan proses pengamanan citra.

REFERENCES

[1] E. Ndruru and T. Zebua, "Pembangkitan Kunci Beaufort Cipher Dengan Teknik Blum-blum Shub pada Pengamanan Citra Digital," vol. 3, no. 2, pp. 149–154, 2022.

[2] W. Wahyudi, D. Hartama, I. O. Kirana, S. Sumarno, and I. Gunawan, "Implementasi Algoritma Kriptografi Rivest Shamir Adlemen untuk Mengamankan Data Ijazah pada SMK Swasta Prama Artha Kab. Simalungun," *J. Ilmu Komput. dan Inform.*, vol. 2, no. 1, pp. 57–66, 2022, doi: 10.54082/jiki.19.

[3] K. Mahesa, B. Sugiantoro, and Y. Prayudi, "Pemanfaatan Metode DNA Kriptografi dalam Meningkatkan Keamanan Citra Digital," *J. Ilm. Inform.*, pp. 2615–1049, 2019.

[4] M. Oky and I. Ruing, "Penerapan Kombinasi Algoritma Kriptografi (Caesar, Vigenere, Zig-Zag) dan Metode Steganografi LSB untuk Mengamankan Pesan Kedalam Citra Digital," 2020.

[5] M. Diana and T. Zebua, "Optimalisasi Beaufort Cipher Menggunakan Pembangkit Kunci RC4 Dalam Penyandian SMS," *J-SAKTI (Jurnal Sains Komput. dan Inform.*, vol. 2, no. 1, p. 12, 2018, doi: 10.30645/j-sakti.v2i1.52.

[6] D. Anggara and A. S. Sembiring, "Peningkatan Keamanan Data Teks Terenkripsi Algoritma Lucifer Menggunakan Steganografi Gifshuffle Pada Citra," *KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer)*, vol. 3, no. 1, pp. 439–445, 2019, doi: 10.30865/komik.v3i1.1626.

[7] R. A. Saputra, I. Awalda Tariza, and B. Pramono, "Implementasi Algoritma Multiply With Carry Generator (MWCG) dalam Pengacakan Soal Ujian Semester Berbasis Web pada SMKN 1 Kendari," *Maret*, vol. 7, no. 1, pp. 60–67, 2022, [Online]. Available:

Submitted: **27/02/2024**; Accepted: **20/03/2024**; Published: **31/03/2024**

http://openjournal.unpam.ac.id/index.php/informatika60.

[8] S. N. Hutagalung, "Implementasi Algoritma Lucifer Untuk Mengamankan Data Inventor Pergudangan," vol. 6, no. November, pp. 191–200, 2022, doi: 10.30865/komik.v6i1.5747.

[9] B. K. Laia, "Modifikasi Algoritma Lucifer Dengan Menerapkan Pembangkitan Kunci Berdasarkan Naive Shuffle," vol. 1, no. 3, pp. 110–118, 2022.

[10] T. Zebua, "Penerapan Multiply With Carry Generator pada Proses Pembangkitan Kunci Algoritma Beaufort Cipher," vol. 4, no. 2, pp. 607–613, 2023, doi: 10.47065/josh.v4i2.2928.

Submitted: **27/02/2024**; Accepted: **20/03/2024**; Published: **31/03/2024**