

Penerapan Algoritma Double Data Encryption Standard Untuk Pengamanan File Citra Digital

Feronika Laia

Fakultas Ilmu Komputer dan Teknologi Informasi, Prodi Teknik Informatika, Universitas Budi Darma Medan, Indonesia

Jl. Sisingamangaraja No. 338, Medan, Sumatera Utara, Indonesia

Email: feronikalaia12@gmail.com

Abstrak—Keamanan merupakan suatu hal yang sangat penting, karena berkaitan dengan kerahasiaan atau privasi. Jika suatu data hilang atau diubah oleh pihak yang tidak bertanggungjawab maka dapat merugikan pemilik data tersebut. Saat ini dimana masih banyak orang menggunakan *file* penting yang sebenarnya tidak boleh diketahui oleh orang lain, tetapi masih saja ada orang yang melakukan pencurian data atau *file* tersebut untuk kepentingan pribadinya, sehingga menimbulkan masalah pada *file* atau data tersebut. Seiring dengan berkembangnya teknologi yang semakin canggih maka faktor untuk menjaga kerahasiaan *file* atau data menjadi hal yang penting bagi orang yang tidak berkepentingan agar tidak dapat mengakses *file* atau data tersebut. Salah satu metode yang dapat digunakan untuk mengamankan sebuah data atau *file* adalah dengan menggunakan *algoritma double data encryption standard*, penggunaan algoritma ini data atau *file* akan dikunci dengan melakukan proses enkripsi dan dekripsi. Algoritma ini dapat mengenkripsi *file*, data, audio, video dan lain-lain. Dari hasil pengujian yang telah dilakukan mengenai pengamanan *file* citra menggunakan algoritma *double data encryption standard* ini, maka *file* ataupun data yang dimiliki akan sulit untuk diketahui dan untuk proses enkripsi dan dekripsi *file* yang besar akan memerlukan waktu yang lama, karena pada *file* citra dan data tersebut telah dilakukan penyandian sehingga tidak akan mudah terjadi penyadapan atau penyerangan pada *file* atau data tersebut sehingga terhindar dari hal-hal yang tidak diinginkan.

Kata Kunci: Kriptografi, *file*, citra, algoritma 2DES.

Abstract—Security is a very important thing, because it is related to confidentiality or privacy. If data is lost or changed by irresponsible parties, it can harm the owner of the data. Currently, many people still use important files that other people should not know about, but there are still people who steal data or files for their personal interests, thus causing problems with the files or data. As technology becomes increasingly sophisticated, the factor of maintaining the confidentiality of files or data becomes important so that unauthorized people cannot access the files or data. One method that can be used to secure data or files is to use a standard double data encryption algorithm. Using this algorithm, data or files will be double locked by carrying out the encryption and description process. This algorithm can encrypt files, data, audio, video and others. From the results of tests that have been carried out regarding the security of image files using this standard double data encryption algorithm, the files or data that are owned will be difficult to identify and the process of encrypting and describing large files will take a long time, because of the image and data files. Encryption has been carried out so that it will not be easy for the files or data to be intercepted or attacked, thus avoiding unwanted things.

Keywords: Cryptography, files, images, 2DES algorithm

1. PENDAHULUAN

Citra digital merupakan salah satu jenis data yang saat ini banyak digunakan dalam berkomunikasi baik secara langsung, maupun melalui media internet. Perkembangan media sosial saat ini sudah sangat mempermudah seseorang untuk saling berkomunikasi, bertukar informasi atau bertukar pesan baik dalam bentuk teks, *file*, citra, audio maupun video. Namun disisi lain perkembangan tersebut menyebabkan semakin mudahnya pihak-pihak tertentu untuk melakukan penyerangan dan penyalahgunaan terhadap data atau informasi yang didistribusikan seperti tindakan penyadapan data atau informasi, memanipulasi data, *file* ataupun informasi untuk kepentingan tertentu.

Keamanan citra pada saat ini sangatlah tidak aman karena pada saat ini masih banyak orang yang suka memanipulasi data, *file* ataupun informasi tertentu untuk kepentingan pribadinya. Oleh karena itu, sangat penting untuk mencegahnya jatuh ketangan pihak-pihak lain yang tidak berkepentingan. Pengamanan *file* dalam bentuk citra menjadi salah satu solusi untuk keamanan sebuah *file* yang bersifat rahasia jika *file* tersebut ingin dikirimkan.

Berdasarkan penelitian terdahulu mengatakan bahwa keamanan merupakan salah satu aspek penting dalam pengiriman data maupun komunikasi melalui jaringan. Salah satu cara untuk menjaga keamanan dan kerahasiaan suatu data maupun informasi adalah dengan teknik enkripsi dan dekripsi atau disebut dengan teknik kriptografi [1].

Kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan ketika pesan dikirim dari suatu tempat ke tempat yang lain. Kriptografi bertujuan untuk mengamankan isi data atau menjaga kerahasiaan informasi dari orang yang tidak berhak untuk mengetahui isi data tersebut. Dengan teknik atau algoritma tertentu yang disebut proses enkripsi (*encrypt*), data diubah menjadi data sandi yang bentuknya berbeda dengan data aslinya. Orang yang berhak menerima data akan dan memiliki kunci untuk mengembalikan data sandi menjadi bentuk data aslinya, proses ini disebut dekripsi (*decrypt*).

Berdasarkan penelitian sebelumnya mengatakan bahwa kriptografi dapat mengatasi masalah keamanan data dengan menggunakan kunci, yang dalam hal ini algoritma tidak dirahasiakan lagi, tetapi kunci harus tetap dijaga kerahasiaannya [2].

Algoritma *double data encryption standard* merupakan salah satu algoritma simetris pada kriptografi yang digunakan untuk mengamankan data dengan cara menyandikan data. Proses yang dilakukan dalam penyandian datanya,

yaitu proses enkripsi dan proses dekripsi. Proses enkripsi dan dekripsi suatu data dengan algoritma DDES dengan menggunakan *plaintext* dan dengan pemilihan kuncinya secara bebas. Sehingga dapat membangkitkan kunci internal pada proses enkripsi dan enkripsi data sehingga tidak memudahkan seseorang untuk melakukan pencurian atau penyalahgunaan suatu *file* atau data penting.

Berdasarkan penelitian yang mengimplementasikan metode DDES menyimpulkan bahwa algoritma *Double Data Encryption Standard* dapat diterapkan untuk mengenkripsi dan mendekripsi file yang berukuran besar secara *real-time*. Waktu yang digunakan untuk melakukan proses enkripsi dan dekripsi juga dapat di persingkat menggunakan operasi *Electronic Code Book (ECB)* yang dilakukan secara paralel dengan memanfaatkan *library* [3].

Penelitian ini menguraikan bagaimana mengimplementasikan algoritma *Double Data Encryption Standard* (DDES) untuk meningkatkan pengamanan file citra yang bersifat rahasia dengan tujuan agar pola asli dari citra digital tidak lagi dapat diketahui oleh orang lain. Proses penerapan metode ini dilakukan dengan menyandikan nilai-nilai warna citra digital yang akan diamankan sehingga ketika dipetakan menjadi citra yang baru (citra hasil penyandian) akan dihasilkan citra yang tidak sama seperti citra aslinya.

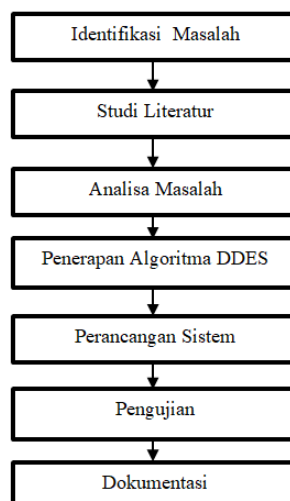
Berdasarkan latar belakang di atas, maka topik penelitian ini adalah judul **“Penerapan Algoritma Double Data Encryption Standard Untuk Pengamanan File Citra Digital”**.

2. METODE PENELITIAN

2.1 Kerangka Kerja Penelitian

Dalam melakukan penelitian ini, penulis melakukan beberapa tahapan, metode mengumpulkan data yang dipakai agar mendapat data yang sangat diperlukan penulis yaitu sebagai berikut:

- Mengidentifikasi Masalah merupakan tahap mengidentifikasi masalah untuk mengetahui masalah apa yang dihadapi dalam pengamanan *file* citra digital, sehingga setelah diidentifikasi, maka akan dicari solusi untuk memecahkan permasalahan tersebut.
- Studi Literatur dilakukan untuk mencari data dan informasi yang berkaitan dengan permasalahan yang sedang diteliti. Studi literatur diambil dari berbagai sumber seperti artikel ilmiah, buku dan jurnal yang berkaitan dengan permasalahan yang ingin diselesaikan pada penelitian ini.
- Analisa Masalah adalah tahap berikutnya yang dilakukan setelah peneliti berhasil mendapatkan beberapa uraian permasalahan. Pada penelitian ini analisa masalah yang dihasilkan adalah bagaimana cara mengamankan file pada citra digital.
- Penerapan algoritma *double data encryption standard* dilakukan dengan memanfaatkan nilai *pixel* untuk proses pengamanan *file* pada citra. Nilai citra desimal dari sampel data akan digunakan untuk melakukan proses pembangkitan kunci, enkripsi dan dekripsi berdasarkan algoritma DDES dalam pengamanan pada *file* citra dengan tujuan agar tidak mudah disalahgunakan oleh pihak tertentu.
- Perancangan Sistem dimulai dengan pemodelan kemudian merancang atau mendesain suatu aplikasi pengamanan menggunakan bahasa pemrograman *Microsoft Visual Studio 2008*.
- Pengujian Merupakan tahap pelaksanaan pengujian aplikasi yang sudah selesai. Pengujian yang dilakukan sebanyak dua kali, yaitu pengujian proses enkripsi dan pengujian proses dekripsi. Pengujian dilakukan untuk melihat kesesuaian hasil yang diperoleh dari aplikasi yang telah dirancang untuk pengamanan *file* tersebut.
- Dokumentasi dilakukan untuk mendokumentasikan seluruh kegiatan penelitian dalam bentuk skripsi yang nantinya juga dibuat dalam bentuk artikel ilmiah yang akan dipublikasikan.



Gambar 1. Kerangka Kerja Penelitian

2.2 Kriptografi

Kriptografi merupakan ilmu yang mempelajari teknik matematika yang berhubungan dengan aspek keamanan informasi, seperti kerahasiaan data, keabsahan data, dan integritas data serta autentifikasi data. Kriptografi adalah ilmu untuk mempelajari penulisan secara rahasia dengan tujuan bahwa komunikasi dan data dapat dikodekan (*encode/encrypt*) dan dikodekan (*decode/decrypt*) kembali untuk mencegah pihak-pihak lain yang ingin mengetahui isinya.

3. HASIL DAN PEMBAHASAN

3.1 Penerapan Algoritma DDES untuk Mengamankan File Citra

Penerapan metode DDES dalam mengamankan *file* citra digital dilakukan dengan menyandikan atau mengubah nilai-nilai *pixel* citra asli menjadi nilai-nilai baru. Proses penyandian ini dilakukan per blok *hexadecimal* dari citra. Jumlah bit per blok proses DDES adalah 64 bit.

Ada tiga proses utama yang harus dilakukan dalam penerapan algoritma DDES ini adalah:

- Proses Pembangkitan Kunci pada penerapan algoritma DDES adalah dua buah kunci eksternal. Kunci eksternal yang dihasilkan bersumber dari 128 bit kunci awal (*initial key*) yang dikelompokkan menjadi dua kelompok masing-masing 64 bit (64 bit menjadi K1 dan 64 bit menjadi K2). Kunci awal (*initial key*) yang digunakan dalam penelitian ini adalah berupa *string* : K1 = fero0987 dan K2 = 7890nika

Tabel 2. Kunci Eksternal

K1			K2		
Char	Decimal	Biner	Char	Decimal	Biner
f	102	01100110	7	55	00110111
e	101	01100101	8	56	00111000
r	114	01110010	9	57	00111001
o	111	01101111	0	48	00110000
0	48	00110000	n	110	01101110
9	57	00111001	i	105	01101001
8	56	00111000	k	107	01101011
7	55	00110111	a	97	01100001

- Proses Pembentukan Kunci 1 (K1)

Biner kunci K1:

0110011001100101011100100110111100110000001110010011100000110111

Lakukan permutasi *compression* 1 (PC 1) terhadap 64 bit K1. Proses PC 1 dilakukan untuk mengkompresi 64 bit K1 menjadi 56 bit.

Tabel 3. Tabel PC-1

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	45	38	30	22
14	6	61	53	46	37	29
21	13	5	28	20	12	4

Nilai-nilai yang ada di dalam tabel menunjukkan posisi bit yang diambil dan diletakkan pada lokasi nilai kolom di dalam tabel PC-1 di atas. berdasarkan tabel di atas terlihat kolom pertama nilainya 57, artinya bahwa dari *string* bit kunci (K1), ambil bit pada posisi ke-57 dan letakkan menjadi bit awal (menjadi bit posisi pertama).

0110011001100101011100100110111100110000001110010011100000110111

Berdasarkan proses tersebut, maka 8 bit dari 64 bit K1 awal akan dibuang dan dimampatkan menjadi 56 bit. Hasil akhir dari proses ini adalah:

PC-1 = 0000000000001111111111111111000110110101011011010000100

Berjumlah 56 bit.

Langkah selanjutnya adalah mengelompokkan biner PC-1 menjadi 2 kelompok (C0 dan D0) masing-masing 28 bit kemudian melakukan proses *Shif Left* Bit

C0 = 0000000000001111111111111111 = 28 bit awal

D0 = 1000110110101011011010000100 = 28 bit berikutnya

Proses selanjutnya adalah melakukan operasi pergeseran bit (*shift left bit operation*) Proses ini merupakan proses pergeseran sejumlah bit dari kiri ke kanan baik biner C0 maupun D0. Jumlah bit yang digeser sesuai dengan tabel *shift left* bit yang sudah ditentukan oleh algoritma DDES.

Tabel 3. Hasil Proses *Shift Left* Bit

Ketentuan		C0	D0
Putaran	Jumlah pindah		
		000000000000111111111111111111	1000110110101011011010000100
1	1	000000000001111111111111111110	0001101101010110110100001001
2	1	000000000011111111111111111100	0011011010101101101000010010
3	2	00000000111111111111111110000	1101101010110110100001001000
4	2	0000001111111111111111000000	0110101011011010000100100011
5	2	0000111111111111111100000000	1010101101101000010010001101
6	2	00111111111111111000000000	1010110110100001001000110110
7	2	11111111111111000000000000	1011011010000100100011011010
8	2	11111111111100000000000011	1101101000010010001101101010
9	1	111111111111000000000000111	1011010000100100011011010101
.....			
16	1	0000000000011111111111111111	1000110110101011011010000100

Gabungkan kembali masing-masing nilai C0 dan D0 yang ada pada tabel 4.3 di atas, sehingga dihasilkan:

C[1],D[1] =	00000000000111111111111110001101101010110110100001001
C[2],D[2] =	0000000000111111111111111000011011010101101101000010010
C[3],D[3] =	0000000011111111111111100001101101010110110100001001000
C[4],D[4] =	0000001111111111111110000000110101011011010000100100011
C[5],D[5] =	00001111111111111111000000001010101101101000010010001101
C[6],D[6] =	0011111111111111110000000001010110110100001001000110110
C[7],D[7] =	11111111111111000000000001011011010000100100011011010
C[8],D[8] =	1111111111110000000000011101101000010010001101101010
C[9],D[9] =	11111111111100000000000111011010000100100011011010101
.....	
C[16],D[16] =	0000000000011111111111111000110110101011011010000100

Berdasarkan hasil di atas, terlihat bahwa jumlah bit untuk masing-masing kelompok gabungan masih 56 bit. Langkah selanjutnya adalah melakukan proses Permutasi *Compression 2* (PC-2) terhadap setiap nilai gabungan C0, D0 di atas.

Tabel 4. Tabel PC-2

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

Hasil dari proses PC-2 ini akan menjadi kunci eksternal pertama (K1) yang digunakan pada proses enkripsi pertama. Hasil seluruh proses PC-2 diperlihatkan pada tabel 5.

Tabel 5. Kunci Eksternal K1

Index	Biner Kunci K1
K[0] =	11010000101011101010111000000010111000101110101
K[1] =	11110000101011101010011011111000100101010010001
K[2] =	11110000101111000100110000100110110001001011011
K[3] =	11100000101101100111011010101111011000100000000
K[4] =	111001001101011001110110101000000010011101100110
K[5] =	1110011011010011011100100111110010101010000110
K[6] =	101011101101001101110011011101000100010011011011
K[7] =	10101111010100110101101100001111011000001001011
K[8] =	001011110101001111011011011010000000011110111011
K[9] =	001111110101000111011001010111110101100000001011

K[15] = 110100011010110010101110100110000010111010100110

Sampai pada tahap ini, telah dihasilkan 16 kunci untuk proses enkripsi peratama. Proses pembangkitan kunci kedua (K2) dengan kunci K2 awal = **7890nika**, dilakukan dengan cara yang sama seperti di sebelumnya, sehingga dihasilkan 16 kunci eksternal kedua (K2). Adapun string biner Kunci K2:

00110111001110000011100100110000011011100110100101100101101100001

Tabel 6. Kunci Eksternal K2

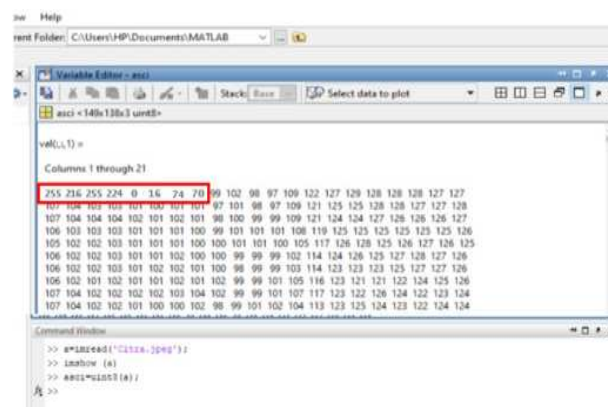
Index	Biner Kunci K2
K[0] =	011000000011110001100100011011000100011001101010
K[1] =	01000000101101000111010001010010110111011110000
K[2] =	110001001100010001110110010010011000110100111101
K[3] =	111001101100001100100010110010110111110010011000
K[4] =	101010101001001100100011011010010101001100111101
K[5] =	101010010001001001011011100100110101100010101110
K[6] =	001001010101001011011000110001000001101110110101
K[7] =	00010110010110011101000010010011001010101111101
K[8] =	000101100100100101010001111000010110011001000111
K[15] =	011100000011111000000100000011111001101101101101

- b. Proses Enkripsi berdasarkan algoritma DDES dilakukan sebanyak dua kali dengan melibatkan K1 dan K2. Masing-masing proses enkripsi terdiri dari 16 *round*. Sehingga, untuk mendapatkan *chipper* citra digital berdasarkan algoritma ini harus melalui 32 *round*. Agar memudahkan proses pengerjaan, maka diambil sebanyak satu blok bit (64 bit sesuai dengan syarat algoritma DDES untuk jumlah bit setiap kelompok).



Gambar 2. Citra Sampel Berjenis *Grayscale*

Software yang digunakan untuk mendapatkan nilai-nilai pixel citra sampel adalah menggunakan *software* Matlab.



Gambar 3. Nilai *Decimal* Citra Sampel

Jumlah nilai yang digunakan dalam contoh penerapan secara manual dalam penelitian ini adalah 8 nilai *pixel* citra (8 *pixel* x 8 bit = 64 bit). Jumlah 64 bit ini memenuhi syarat 1 blok dalam algoritma DDES.

Tabel 7. Nilai *Pixel* Citra yang Digunakan Sebagai Contoh

Index	Decimal	Biner
1	255	11111111
2	216	11011000
3	255	11111111
4	224	11100000
5	0	00000000
6	16	00010000
7	74	01001010
8	70	01000110

Langkah-langkah yang dilakukan dalam proses enkripsi baik enkripsi tahap pertama maupun enkripsi tahap kedua adalah:

1. Melakukan proses *Initial Permutation* (IP) terhadap gabungan biner citra yang menjadi sampel dalam perhitungan ini.

Tabel 8. Tabel IP DES

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Proses ini sama dengan proses PC-1 pada pembangkitan kunci, yaitu memindahkan posisi bit tertentu berdasarkan nilai-nilai yang ada pada tabel 8 di atas. Misalnya bit ke-58 dipindahkan ke posisi pertama, demikian seterusnya. *String* biner citra asli (*plain image* sejumlah 64 bit/1 blok DDES):

111111111011000111111111100000000000000100000100101001000110

Berdasarkan proses pemindahan di atas, maka didapatkan hasil proses *Initial Permutation* (IP) adalah:

IP = 110011110010011110000101000001010000111100001101010001111000101

String biner IP ini kemudian dibagi menjadi dua kelompok, yaitu 32 bit kiri (menjadi L0) dan 32 bit sisanya (menjadi R0). Nilai kelompok L0 dan R0 ini menjadi inisialisasi awal yang digunakan dalam proses enkripsi tahap pertama sebanyak 16 putaran, sehingga dihasilkan:

L[0] = 11001111001001111000010100000101 = 32 bit

R[0] = 00001111000011010100011111000101 = 32 bit

2. Proses Iterasi dilakukan sebanyak 16 *round* hingga dihasilkan *cipherimage* pada tahap enkripsi pertama ini. Proses iterasi ini juga dilakukan sama pada proses iterasi enkripsi tahap kedua.

Iterasi ke-0:

- a) Langkah awal adalah melakukan ekspansi 32 bit R0 menjadi 48 bit

Proses ekspansi ini dilakukan berdasarkan aturan tabel ekspansi (tabel E) yang telah disediakan oleh algoritma DES. Tujuan proses ekspansi ini adalah agar jumlah bit R[0] sama dengan jumlah bit kunci eksternal yang digunakan.

Tabel 9. Tabel Ekspansi

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Proses yang dilakukan pada tahap ekspansi ini adalah memindahkan bit-bit bit pada posisi nilai tabel ekspansi. Misalnya nilai kolom pertama pada tabel ekspansi adalah 32, berarti bit R[0] pada posisi ke 32 dipindahkan menjadi bit pertama, demikian seterusnya. Hasil ekspansi disimpan ke dalam variable E.

R[0] = 00001111000011010100011111000101 = 32 bit

E[0] = 10000101111010000101101010100000111111000001010 = 48 bit

Hasil ekspansi (E[0]) di atas di-XOR-kan dengan biner kunci K1[0] :

E[0] = 10000101111010000101101010100000111111000001010

K1[0] = 110100001010111010101110000000010111000101110101 XOR

A[0] = 01010101010001101111010010100001100011110111111

- b) Langkah berikutnya adalah Substitusi biner hasil Expansi (A[0]) menggunakan tabel S-Box yang telah disediakan oleh algoritma DES.

Tabel 10. Tabel Hasil S-BOX

Kelompok A[0]	Biner Hasil Kelompok	Decimal dari Biner		Hasil S-Box [Decimal]	Biner Hasil S-Box (hasil 4 bit)
		b1,b6 [baris]	b2,b3,b4,b5 [kolom]		
1	010101	1	10	12	1100
2	010100	0	10	2	0010
3	011011	1	13	11	1011
4	110100	2	10	3	0011
5	101000	2	4	10	1010
6	011000	0	12	14	1110
7	111101	3	14	3	0011
8	111111	3	15	11	1011

Gabungkan kembali biner-biner pada kolom hasil S-Box mulai dari kelompok awal sampai akhir, sehingga jumlahnya menjadi 32 bit dan disimpan ke dalam variabel B[0].

B[0] = 11000010101100111010111000111011

- c) Langkah selanjutnya adalah proses permutasi nilai B[0]. Langkah ini dilakukan berdasarkan aturan table P-Box

Tabel 11. Tabel P-Box

17	7	20	21	29	12	18	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Nilai bit B[0] dari hasil proses S-Box akan dipermutasi kembali berdasarkan posisi-posisi pada table P-Box. Misalnya nilai kolom pertama pada table P-Box adalah 17, maka ambil bit B[0] pada posisi ke-17 dan pindahkan menjadi bit pertama. Hasil dari proses Permutasi nilai B[0] adalah :

B[0] = 11000010101100111010111000111011

P-Box[0] = 1101111111000101000110110001100

- d) Cari nilai R[selanjutnya] dan L[selanjutnya] dengan melakukan proses XOR antara nilai P-Box[0] dengan nilai L[0] (lihat biner nilai L[0] awal. Hasil XOR antara P-Box[0] dengan L[0] menjadi nilai R [selanjutnya]

P-Box[0] = 1101111111000101000110110001100

L[0] = 11001111001001111000010100000101 XOR

R[1] = 00010000110001010000100010001001

L[selanjutnya] atau L[1] adalah nilai R[sebelumnya], artinya:

L[1] = R[0]

L[1] = 00001111000011010100011111000101

Sampai pada proses iterasi ke-0 ini, maka nilai L[1] dan R[1] telah didapatkan:

L[1] = 00001111000011010100011111000101

R[1] = 00010000110001010000100010001001

Nilai L[1] dan R[1] digunakan pada proses iterasi berikutnya untuk mendapatkan string bit L[2] dan R[2].

Proses yang dilakukan sama seperti yang dilakukan untuk mendapatkan nilai L[1] dan R[1] di atas.

Iterasi ke-1:

Proses yang dilakukan pada iterasi 1 ini pada dasarnya sama seperti proses yang dilakukan pada iterasi ke-0. String bit input yang digunakan pada iterasi ke-1 ini adalah hasil nilai L[1] dan R[1] yang telah dihasilkan dari iterasi sebelumnya (iterasi ke-0)

L[1] = 00001111000011010100011111000101

R[1] = 00010000110001010000100010001001

Biner R[1] akan diekspansi menggunakan tabel ekspansi, kemudian hasil ekspansinya di XOR dengan kunci K[1], sehingga hasilnya:

E(R[1]) = 100010100001011000001010100001010001010001010010

$K1[1] = \underline{111100001010111010100110111111000100101010010001}$ XOR

$A[1] = 01111010101110001010110001111001010111011000011$

Biner $A[1]$ disubstitusikan dengan tabel S-Box, sehingga dihasilkan :

$B[1] = 01111111011001111001110100101111$

Nilai $B[1]$ dipermutasikan menggunakan tabel P-Box, sehingga :

$P(B[1]) = 1111100101001011111111000111110$

Biner $P(B[1])$ XOR dengan $L[1]$, sehingga dihasilkan nilai $R[2]$:

$P(B[1]) = 1111100101001011111111000111110$

$L[1] = \underline{00001111000011010100011111000101}$ XOR

$R[2] = 11110110010001101011100111111011$

$L[2] = R[1]$

$L[2] = 00010000110001010000100010001001$

Sehingga hasil iterasi ke-1 ini didapatkan nilai biner $L[2]$ dan $R[2]$:

$L[2] = 00010000110001010000100010001001$

$R[2] = 11110110010001101011100111111011$

Nilai $L[2]$ dan $R[2]$ akan digunakan pada iterasi berikutnya untuk mendapatkan *string* biner $L[3]$ dan $R[4]$.

Iterasi ke-2:

$L[2] = 00010000110001010000100010001001$

$R[2] = 11110110010001101011100111111011$

Ekspansi $R[2]$ berdasarkan tabel ekspansi, sehingga dihasilkan :

$E(R[2]) = 11111010110000100000110101011111001111111110111$

$K1[2] = \underline{111100001011111000100110000100110110001001011011}$ XOR

$A[2] = 000010100111110000101011010011000101110110101100$

Biner $A[2]$ disubstitusikan dengan tabel S-Box, sehingga dihasilkan :

$B[2] = 01000001101100010000010010001110$

Nilai $B[2]$ dipermutasikan menggunakan tabel P-Box, sehingga :

$P(B[2]) = 10001100000000101100000100101101$

Biner $P(B[2])$ XOR dengan $L[2]$, sehingga dihasilkan nilai $R[3]$:

$P(B[2]) = 10001100000000101100000100101101$

$L[2] = \underline{00010000110001010000100010001001}$ XOR

$R[3] = 10011100110001111100100110100100$

$L[3] = R[2]$

$L[3] = 11110110010001101011100111111011$

Sehingga hasil iterasi ke-2 ini didapatkan nilai biner $L[3]$ dan $R[3]$:

$L[3] = 11110110010001101011100111111011$

$R[3] = 10011100110001111100100110100100$

Nilai $L[3]$ dan $R[3]$ akan digunakan pada iterasi berikutnya untuk mendapatkan $L[4]$ dan $R[4]$.

Iterasi ke-3:

$L[3] = 11110110010001101011100111111011$

$R[3] = 10011100110001111100100110100100$

Ekspansi $R[3]$ berdasarkan tabel ekspansi, sehingga dihasilkan :

$E(R[3]) = 01001111100101100000111111001010011110100001001$

$K1[3] = \underline{11100000101101100111011010110111011000100000000}$ XOR

$A[3] = 101011110010000001111001010100101000110000001001$

Biner $A[3]$ disubstitusikan dengan tabel S-Box, sehingga dihasilkan :

$B[3] = 10011000110111000011001010101010$

Nilai $B[3]$ dipermutasikan menggunakan tabel P-Box, sehingga :

$P(B[3]) = 00101100101010110001010111000011$

Biner $P(B[3])$ XOR dengan $L[3]$, sehingga dihasilkan nilai $R[4]$:

$P(B[3]) = 00101100101010110001010111000011$

$L[3] = \underline{11110110010001101011100111111011}$ XOR

$R[4] = 11011010111011011010110000111000$

$L[4] = R[3]$

$L[4] = 10011100110001111100100110100100$

Sehingga hasil iterasi ke-3 ini didapatkan nilai biner $L[4]$ dan $R[4]$:

$L[4] = 10011100110001111100100110100100$

$R[4] = 11011010111011011010110000111000$

Nilai $L[4]$ dan $R[5]$ akan digunakan pada iterasi berikutnya untuk mendapatkan $L[5]$ dan $R[5]$.

Iterasi ke-4:

$L[4] = 10011100110001111100100110100100$

$R[4] = 11011010111011011010110000111000$

Eksansi $R[4]$ berdasarkan tabel ekspansi, sehingga dihasilkan :

$E(R[4]) = 011011110101011101011011110101011000000111110001$

$K1[4] = 111001001101011001110110101000000010011101100110$ XOR

$A[4] = 100010111000000100101101011101011010011010010111$

Biner $A[4]$ disubstitusikan dengan tabel S-Box, sehingga dihasilkan :

$B[4] = 00011001100111011000011110101011$

Nilai $B[4]$ dipermutasikan menggunakan tabel P-Box, sehingga :

$P(B[4]) = 10001101001010100111110101001011$

Biner $P(B[4])$ XOR dengan $L[4]$, sehingga dihasilkan nilai $R[5]$:

$P(B[4]) = 10001101001010100111110101001011$

$L[4] = 10011100110001111100100110100100$ XOR

$R[5] = 00010001111011011011010011101111$

$L[5] = R[4]$

$L[5] = 11011010111011011010110000111000$

Sehingga hasil iterasi ke-1 ini didapatkan nilai biner $L[5]$ dan $R[5]$:

$L[5] = 11011010111011011010110000111000$

$R[5] = 00010001111011011011010011101111$

Nilai $L[5]$ dan $R[5]$ akan digunakan pada iterasi berikutnya.

Iterasi ke-5 sampai iterasi ke-15 dilakukan dengan cara yang sama seperti di atas, sehingga dihasilkan $L[15]$ dan $R[15]$. Hasil keseluruhan proses enkripsi pertama ini adalah:

Tabel 12. Hasil Keseluruhan *Round* Proses Enkripsi Tahap Pertama

Round	Biner $R[i]$	Biner $L[i]$
0	00001111000011010100011111000101	11001111001001111000010100000101
1	00010000110001010000100010001001	00001111000011010100011111000101
2	11110110010001101011100111111011	00010000110001010000100010001001
3	10011100110001111100100110100100	11110110010001101011100111111011
4	11011010111011011010110000111000	10011100110001111100100110100100
5	00010001111011011011010011101111	11011010111011011010110000111000
6	11111100100111010000000101001111	00010001111011011011010011101111
7	01101011100110010000011111100111	11111100100111010000000101001111
8	10110011110101000000110100110001	01101011100110010000011111100111
9	00001010101110001111110011011011	10110011110101000000110100110001
.....		
16	01010001110001111111010101011000	11010001001010111110111010111001

Langkah berikutnya adalah menggabungkan *string* biner bagian $R[16]$ dengan $L[16]$, kemudian dilakukan proses *invers initial permutation* (IP^{-1}). Proses IP^{-1} sama dengan proses IP yang dilakukan pada bagian awal.

Tabel 13. Tabel IP^{-1}

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Gabungan *string* bit $R[16]$ dan $L[16]$:

01010001110001111111010101011000 11010001001010111110111010111001

Hasil IP^{-1} dari *string* biner $R[16]$ dan $L[16]$ adalah :

111101100011100000011100001010111000111001011101101110110011110

String bit hasil IP^{-1} dari proses enkripsi tahap pertama ini terlebih dahulu melalui proses *initial permutation* (IP) berdasarkan tabel IP , sehingga dihasilkan *string* bit $R[0]$ dan $L[0]$ yang menjadi input pada proses enkripsi tahap kedua.

String bit hasil IP^{-1} dari proses enkripsi tahap pertama :

111101100011100000011100001010111000111001011101101110110011110

Hasil IP : 010100011100011111110101010110001101000100101011110111010111001

String bit hasil IP ini dibagi menjadi dua kelompok. Masing-masing 32 bit :

$L[0] = 01010001110001111111010101011000$

$R[0] = 1101000100101011110111010111001$

Nilai $L[0]$ dan $R[0]$ inilah yang digunakan pada proses iterasi enkripsi tahap kedua.

Tabel 14. Hasil Iterasi Enkripsi Tahap Kedua

Round	Biner $R[i]$	Biner $L[i]$
0	01110010110111101010010011001011	1101000100101011110111010111001
1	11000101010110111001001000110100	01110010110111101010010011001011
2	10100000111000000001100110011110	11000101010110111001001000110100
3	11011011001011011000001010000111	10100000111000000001100110011110
4	00000001011101110100111110101010	11011011001011011000001010000111
5	01101010001110110001100010000110	00000001011101110100111110101010
6	11100101110100001011001011100100	01101010001110110001100010000110
7	11010101100010000101110111001001	11100101110100001011001011100100
8	0001011001111110000101100111110	11010101100010000101110111001001
9	01000001101011100111100001101000	00010110011111110000101100111110
.....		
15	01001011011110101111101011011111	11010111110101110100000110000010

String biner $R[15]$ dan $L[15]$ digabungkan, kemudian dilakukan proses IP^{-1} , sehingga dihasilkan *string* biner :

$R[15], L[15] = 01001011011110101111101011011111101011110101110100000110000010$

String biner $R[15]$ dan $L[15]$ dipermutasikan berdasarkan tabel IP^{-1} seperti pada proses enkripsi tahap pertama, sehingga dihasilkan biner *cipher* akhir (*cipher* hasil enkripsi tahap kedua) dari citra asli adalah : 1110100111110111101000010101010110110101000101001111110110100111

Biner *cipher* video ini dikelompokkan menjadi 8 bit per kelompok, kemudian dikonversi menjadi nilai *hexadecimal*.

Tabel 15. Hasil Pengelompokkan Biner *Cipher Image*

Index	Biner	Decimal
1	11101001	233
2	11110111	247
3	10100001	161
4	01010101	85
5	10110101	181
6	00010100	20
7	11111101	253
8	10100111	167

c. Proses Dekripsi

Tahap pertama merupakan dekripsi menggunakan kunci kedua (K2) dengan urutan terbaik (dimulai dari kunci ke-15-k-0), sedangkan tahap dekripsi yang kedua menggunakan kunci pertama (K1).

String biner *cipher image*:

1110100111110111101000010101010110110101000101001111110110100111

Dekripsi Tahap Pertama menggunakan kunci kedua (K2) :

2. Lakukan proses IP terhadap *string* biner *cipher image*.

3. Proses iterasi untuk menentukan nilai $L[i]$ dan $R[i]$

Iterasi ke-0 ($i = 0$):

Nilai $L[i+1]$ pada iterasi awal ini didapatkan dari nilai $R[0]$, sehingga dihasilkan :

$L[i+1] = R[0]$ atau $L[0+1] = R[0]$, sehingga :

$L[1] = 11010111110101110100000110000010$

$R[0] = 11010111110101110100000110000010$

String bit $R[0]$ akan diproses untuk mendapatkan nilai $R[i+1]$.

Proses pencarian *string* bit $R[i+1]$ dilakukan dengan meng-expand nilai $R[i]$, menggunakan tabel ekspansi (tabel 4.9) seperti pada proses enkripsi, kemudian simpan di dalam variabel E. Ekspansi *string* biner $R[i]$ kemudian XOR dengan $K2[i]$, hasilnya disimpan ke dalam variabel $E[i]$. Urutan kunci K2 yang digunakan dimulai dari index ke-15 hingga index ke-0). Sehingga pada proses ini dihasilkan :

$E(R[0]) = 011010101111111010101110101000000011110000000101$

$K2[0] = 011100000011111000000100000011111001101101101101$ XOR

$A[0] = 0001101011000000101010101011111010011101101000$

Substitusikan *string* bit $A[0]$ ke dalam tabel S-Box DES dan simpan hasilnya ke dalam variabel B, sehingga dihasilkan:

$B[0] = 00011101000010111110110110001001$

Permutasikan *string* bit $B[i]$, berdasarkan tabel P-Box, sehingga:

$P(B[0]) = 10011001010011000110100011011011$

Menentukan nilai $R[i+1]$:

$$R[i+1] = P(B[i]) \text{ XOR } L[i-1]$$

$$P(B[0]) = 10011001010011000110100011011011$$

$$L[0] = \underline{0100101101111010111110101101111} \text{ XOR}$$

$$R[1] = 11010010001101101001001000000100 \rightarrow \text{dijadikan } L[2]$$

Iterasi ke-1 (i = 1):

$$L[2] = 11010010001101101001001000000100$$

$$R[1] = 11010010001101101001001000000100$$

Ekspansi $R[1]$ kemudian XOR dengan $K[1]$:

$$E(R[1]) = 01101010010000011010110101001010010000000001001$$

$$K2[1] = \underline{00110000001111100000011011111011111010100000010} \text{ XOR}$$

$$A[1] = 01011010011111110101011101101111011010100001011$$

String bit $A[1]$ di S-Box, sehingga dihasilkan :

$$B[1] = 11000001011100010010000010010011$$

Permutasikan *string* bit $B[1]$, berdasarkan tabel P-Box, sehingga :

$$P(B[1]) = 10000110100000111100100010000101$$

Menentukan nilai $R[i+1]$:

$$R[i+1] = P(B[i]) \text{ XOR } L[i-1]$$

$$P(B[1]) = 10000110100000111100100010000101$$

$$L[1] = \underline{11010111110101110100000110000010} \text{ XOR}$$

$$R[2] = 01010001010101001000100100000111 \rightarrow \text{dijadikan } L[3]$$

Iterasi ke-2 (i = 2):

$$L[3] = 01010001010101001000100100000111$$

$$R[2] = 01010001010101001000100100000111$$

Ekspansi $R[2]$ kemudian XOR dengan $K[2]$:

$$E(R[2]) = 1010101000101010101010001010001010010100000001110$$

$$K2[2] = \underline{1001000010101010100011100010101010011101011010} \text{ XOR}$$

$$A[2] = 001110101000000000100111011011111000111101010100$$

String bit $A[2]$ di S-Box, sehingga dihasilkan :

$$B[2] = 10001010101001101001000100110011$$

Permutasikan *string* bit $B[1]$, berdasarkan tabel P-Box, sehingga :

$$P(B[2]) = 01100011110010100011110100000100$$

Menentukan nilai $R[i+1]$:

$$R[i+1] = P(B[i]) \text{ XOR } L[i-1]$$

$$P(B[2]) = 01100011110010100011110100000100$$

$$L[1] = \underline{11010010001101101001001000000100} \text{ XOR}$$

$$R[3] = 10110001111111001010111100000000 \rightarrow \text{dijadikan } L[4]$$

Iterasi ke-3 (i = 3):

$$L[4] = 10110001111111001010111100000000$$

$$R[3] = 10110001111111001010111100000000$$

Ekspansi $R[3]$ kemudian XOR dengan $K[3]$:

$$E(R[3]) = 010110100011111111111100101010101110100000000001$$

$$K2[3] = \underline{11011001100010001010100011110110111010101000001} \text{ XOR}$$

$$A[3] = 100000111011011101010001101000110000010101000000$$

String bit $A[3]$ di S-Box, sehingga dihasilkan :

$$B[3] = 01000101111101001010011101011101$$

Permutasikan *string* bit $B[1]$, berdasarkan tabel P-Box, sehingga :

$$P(B[3]) = 00001111001100011111100110111100$$

Menentukan nilai $R[i+1]$:

$$R[i+1] = P(B[i]) \text{ XOR } L[i]$$

$$P(B[3]) = 00001111001100011111100110111100$$

$$L[3] = \underline{01010001010101001000100100000111} \text{ XOR}$$

$$R[4] = 01011110011001010111000010111011 \rightarrow \text{dijadikan } L[5]$$

Iterasi ke-4 (i = 4):

$$L[5] = 01011110011001010111000010111011$$

$$R[4] = 01011110011001010111000010111011$$

Ekspansi $R[4]$ kemudian XOR dengan $K[4]$:

$$E(R[4]) = 10101111110000110000101010111010000101011110110$$

$$K2[4] = \underline{010110110000010010101001000111101010001011100011} \text{ XOR}$$

$$A[4] = 111101001100011110100011101001001011011100010101$$

String bit A[4] di S-Box, sehingga dihasilkan :

B[4] = 01100011100011110001110001100110

Permutasikan string bit B[1], berdasarkan tabel P-Box, sehingga :

P(B[4]) = 11110000010100101101011101101000

Menentukan nilai R[i+1] :

$R[i+1] = P(B[i]) \text{ XOR } L[i]$

P(B[4]) = 11110000010100101101011101101000

L[4] = 1011000111111001010111100000000 XOR

R[5] = 01000001101011100111100001101000 → dijadikan L[6]

Proses iterasi ke-5 sampai iterasi ke-15, dilakukan dengan cara yang sama seperti di atas hingga didapatkan nilai L[15] dan R[15]. Hasil L[i] dan R[i] untuk keseluruhan round pada tahap enkripsi pertama ini adalah :

Tabel 16. Hasil Round Dekripsi Tahap Pertama

Round	L[i]	R[i]
0	11010111110101110100000110000010	11010111110101110100000110000010
1	11010010001101101001001000000100	11010010001101101001001000000100
2	01010001010101001000100100000111	01010001010101001000100100000111
3	101100011111100101011100000000	101100011111100101011100000000
4	01011110011001010111000010111011	01011110011001010111000010111011
5	01000001101011100111100001101000	01000001101011100111100001101000
6	0001011001111110000101100111110	0001011001111110000101100111110
7	11010101100010000101110111001001	11010101100010000101110111001001
8	11100101110100001011001011100100	11100101110100001011001011100100
9	01101010001110110001100010000110	01101010001110110001100010000110
10	00000001011101110100111110101010	00000001011101110100111110101010
.....		
15	0101000111000111111010101011000	11010001001010111110111010111001

String biner R[15] dan L[15] pada tabel di atas digabungkan, kemudian dilakukan proses *invers initial permutation* (IP⁻¹) seperti pada proses enkripsi, sehingga dihasilkan :

R[15] = 0101000111000111111010101011000

L[15] = 1101000100101011110111010111001

String bit hasil penggabungan:

010100011100011111101010110001101000100101011110111010111001

String bit ini akan diproses melalui IP⁻¹, sehingga dihasilkan:

111101100011100000011100001010111000111001011101101110110011110

String bit hasil IP⁻¹ di atas merupakan hasil akhir dari proses dekripsi tahap pertama. String bit ini akan digunakan pada proses dekripsi tahap kedua.

Dekripsi Tahap Kedua menggunakan Kunci K1

Proses dekripsi tahap kedua ini dilakukan seperti proses dekripsi tahap pertama, hanya saja string bit kunci yang digunakan adalah kunci K1. Hasil proses dekripsi tahap kedua, disajikan pada tabel di bawah ini:

Tabel 17. Hasil Round Dekripsi Tahap Kedua

Round	R[i]	L[i]
0	11010001001010111110111010111001	11010001001010111110111010111001
1	01100000010011001010000000111101	01100000010011001010000000111101
2	0101111101110001101011010111010	0101111101110001101011010111010
3	01010011010101101011111010111111	01010011010101101011111010111111
4	0011111011111001101101011110101	0011111011111001101101011110101
5	10010111011100100001101111011011	10010111011100100001101111011011
6	00001010101110001111110011011011	00001010101110001111110011011011
7	10110011110101000000110100110001	10110011110101000000110100110001
8	01101011100110010000011111100111	01101011100110010000011111100111
9	11111100100111010000000101001111	11111100100111010000000101001111
.....		
15	11001111001001111000010100000101	00001111000011010100011111000101

String biner hasil proses IP⁻¹ ini merupakan string biner *plainimage* hasil proses dekripsi. String biner ini akan dikelompokkan menjadi 8 bit setiap kelompok untuk menghasilkan nilai-nilai *plain image* awal.

Tabel 18. Hasil Dekripsi Tahap Kedua (*PlainImage*)

Biner	Decimal
11111111	255
11011000	216
11111111	255
11100000	224
00000000	0
00010000	16
01001010	74
01000110	70

Nilai-nilai *hexa* pada tabel 17. di atas sama dengan nilai-nilai *hexa* citra asli (*plain image*) sebelum dilakukan proses enkripsi. Nilai-nilai tersebut akan dipetakan kembali menjadi nilai warna *pixel* citra digital, sehingga didapatkan citra digital asli.

4. KESIMPULAN

Berdasarkan penelitian yang sudah dilakukan pada pengamanan *file* citra digital maka dapat disimpulkan bahwa Percobaan yang dilakukan penulis berhasil membuktikan bahwa pengamanan *file* citra dapat berjalan dengan baik, sehingga *file* yang bersifat rahasia tidak dapat diketahui oleh siapa pun. Dengan melakukan perubahan dari citra asli menjadi *cipherimage* disebut enkripsi dan proses deskripsi dilakukan dengan cara mengembalikan *cipherimage* menjadi citra asli. Penerapan *algoritma double data encryption standard* dilakukan dengan mengubah nilai *pixel* citra asli menjadi nilai yang baru yang di dalamnya memiliki proses pembangkitan kunci, proses enkripsi dan proses deskripsi. Perancangan aplikasi untuk pengamanan *file* citra menggunakan bahasa pemrograman *visual basic* 2008 agar dapat memudahkan dalam proses enkripsi dan deskripsi *file* citra. Adapun saran dari penulis yang berkaitan dengan teknik pengamanan *file* citra adalah untuk Penggunaan algoritma *double data encryption standard* diharapkan dapat di kembangkan lagi dengan kriptografi lainnya seperti kriptografi simetris, kriptografi klasik, dan lain-lain. Aplikasi pengamanan data ini dapat diimplementasikan di instansi yang memiliki beberapa *file* atau dokumen penting yang harus dijaga. Aplikasi yang dibangun diharapkan mampu dikembangkan dengan bahasa pemrograman lainnya seperti berbasis *website* atau *android*.

REFERENCES

- [1] J. A. Sinuhaji, "Implementasi Pengamanan Citra Digital Menggunakan Algoritma ICE," vol. 1, pp. 284–293, 2020, doi: 10.30865/json.v1i3.2138.
- [2] Sujarweni, "Bab II Landasan Teori," *J. Chem. Inf. Model.*, vol. 53, no. 9, pp. 1689–1699, 2018.
- [3] A. Shorman and M. Qatawneh, "Performance Improvement of Double Data Encryption Standard Algorithm using Parallel Computation," *Int. J. Comput. Appl.*, vol. 179, no. 25, pp. 1–6, 2018, doi: 10.5120/ijca2018916527.
- [4] B. A. B. Ii and T. Pustaka, "BAB II TINJAUAN PUSTAKA 2.1 Tinjauan Pustaka," pp. 3–15, 2016.
- [5] I. W. Sinaga, I. Saputra, and T. Zebua, "Pengamanan Data Nilai Pada Aplikasi E-Raport Berdasarkan Algoritma 2Des," *KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer)*, vol. 3, no. 1, pp. 290–298, 2019, doi: 10.30865/komik.v3i1.1604.
- [6] M. Kurniasih, "Bab Ii Landasan Teori," *J. Chem. Inf. Model.*, vol. 53, no. 9, pp. 8–24, 2018.
- [7] R. D. Anas, "Identifikasi Nomor Plat Kendaraan Menggunakan Metode Wavelet Haar," pp. 1–21, 2015.
- [8] D. A. Wp, "Peningkatan Keamanan Data dengan Metode Cropping Selection Pseudorandom," vol. 4, no. 3, pp. 132–138, 2016.
- [9] "Perancangan aplikasi pengamanan citra digital menggunakan java dengan algoritma rijndael.pdf."
- [10] O. K. Sulaiman, M. Ihwani, and S. F. Rizki, "Model Keamanan Informasi Berbasis Tanda Tangan Digital Dengan Data Encryption Standard (Des) Algorithm," *InfoTekJar (Jurnal Nas. Inform. dan Teknol. Jaringan)*, vol. 1, no. 1, pp. 14–19, 2016, doi: 10.30743/infotekjar.v1i1.82.
- [11] A. H. Kridalaksana, A. Y. Rangan, and A. Ansharie, "Enkripsi Data Audio Menggunakan Metode Kriptografi Rsa," *Sebatik*, vol. 17, no. 1, pp. 6–10, 2017, doi: 10.46984/sebatik.v17i1.79.
- [12] Y. P. Dewi, "Pengembangan Teknik Steganografi Dengan Kriptografi Modifikasi dari Caesar Cipher dan SHA-256 Untuk Merahasiakan Pesan," *J. Comput. Sci. Vis. ...*, vol. 5, pp. 10–21, 2020, [Online]. Available: <http://journal.unusida.ac.id/index.php/jik/article/view/129%0Ahttps://journal.unusida.ac.id/index.php/jik/article/download/129/215>.