
LAPORAN PROJEK MATA KULIAH

KRIPTOGRAFI



**IMPLEMENTASI ALGORITMA COLUMNAR TRANSPOSITION CIPHER DAN DES
SERTA PEMBANGKITAN KUNCI DENGAN ALGORITMA PLAYFAIR CIPHER DAN
DES MEGGUNAKAN BAHASA PEMROGRAMAN C++**

Disusun Oleh : Faishal Fitra Ramadhan NIM. 08011282126060
Muhammad Arya Al Fajri NIM. 08011282126062
Dosen Pengajar : Dr. Anita Desiani, S.Si., M.Kom

**JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS SRIWIJAYA
2023**

I. PENDAHULUAN

Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi, seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data (Amin, 2017). Kriptografi dapat diartikan juga sebagai ilmu atau seni untuk menjaga keamanan pesan. Secara umum, kriptografi terbagi atas dua metode yaitu metode Simetris dan Asimetris (Basri, 2016). Kunci simetris merupakan kunci yang dipakai dalam proses enkripsi dan dekripsi. Sedangkan kunci asimetris memiliki kunci yang berbeda dalam proses enkripsi dan dekripsi, proses enkripsi menggunakan kunci publik dan proses dekripsi menggunakan kunci privat (Saputro et al., 2020). Beberapa contoh dari algoritma simetris adalah yang umum digunakan adalah algoritma DES, Playfair Cipher, dan Columnar Transposition Cipher.

Algoritma DES dikembangkan di IBM dibawah kepemimpinan W.L.Tuchman pada tahun 1972. Algoritma ini didasarkan pada algoritma Lucifer yang dibuat oleh Horst Feistel. Algoritma ini telah disetujui oleh National Bureau of Standard (NBS) setelah penilaian kekuatannya oleh National Security Agency (NSA) Amerika Serikat. DES termasuk ke dalam sistem kriptografi simetri dan tergolong jenis cipher blok. DES beroperasi pada ukuran blok 64 bit. DES mengenkripsikan 64 bit plainteks menjadi 64 bit cipherteks dengan menggunakan 56 bit kunci internal (internal key) atau upa-kunci (subkey). Kunci internal dibangkitkan dari kunci eksternal (external key) yang panjangnya 64 bit (Nugroho & Pramusinto, 2018). Namun, Manajemen kunci di DES kurang fleksibel dan rentan terhadap beberapa jenis serangan, seperti serangan pertukaran kunci dan serangan terhadap mekanisme pembangkitan kunci. Hal ini mengurangi keamanan secara keseluruhan.

Metode Playfair Cipher merupakan salah satu metode yang digolongkan dalam kriptografi klasik yang proses enkripsinya menggunakan pemrosesan dalam bentuk blok-blok yang sangat besar (Susanti, 2020). Metode Playfair Cipher menggunakan pembentukan tabel berdasarkan kunci yang diketahui. Komponen yang penting pada metode playfair adalah tabel cipher yang digunakan untuk melakukan enkripsi dan dekripsi tabel bawaan yang diperkenalkan oleh playfair adalah tabel yang berbentuk matrik berukuran (5x5) yang berisi huruf kapital dari A-Z dengan menghilangkan J. Tabel bawaan yang ada pada Playfair Cipher tidak dapat mengenkripsi Plaintext yang berisi angka (0-9) dan simbol-simbol. Kelemahan yang lain pada playfair adalah terjadinya ambigu pada hasil dekripsi karena pada persiapan enkripsi, Playfair Cipher memiliki mekanisme mengganti J dengan I dan jumlah karakter pada tabel 5x5 itu terbatas.

Columnar Transposition Cipher adalah sebuah algoritma kriptografi klasik yang sederhana dan sangat mudah untuk diimplementasikan (Siregar et al., 2020). Meskipun algoritma ini relatif mudah dipecahkan, namun jika dikombinasikan dengan algoritma lain maka akan sulit untuk memecahkannya (Dar, 2014). Pengamanan pesan melalui teknik ini dilakukan dengan cara memasukkan setiap huruf pada kolom sebuah tabel secara horizontal, kemudian kolom tabel tersebut diacak berdasarkan kata kunci yang telah ditentukan. Setelah itu tuliskan kembali setiap huruf dari tabel tersebut secara vertikal untuk menghasilkan ciphertext (Tendean, n.d.).

Pada penelitian ini akan dilakukan kombinasi antara ketiga algoritma simetris yaitu algoritma Playfair Cipher dan DES untuk membangkitkan kunci pesan serta algoritma Columnar Transposition Cipher untuk enkripsi pesan. Penerapan kombinasi ketiga algoritma tersebut diharapkan bisa meningkatkan keamanan pesan sehingga sulit ditebak oleh pihak manapun.

II. METODE

Langkah- Langkah yang digunakan dalam membangkitkan kunci menggunakan algoritma Playfair Cipher dan DES yaitu sebagai berikut :

a. Algoritma Playfair Cipher

Langkah-langkah enkripsi dalam algoritma Playfair Cipher melibatkan beberapa aturan untuk menggantikan pasangan huruf dalam teks sederhana dengan pasangan huruf yang dihasilkan dari matriks kunci Playfair. Berikut adalah langkah-langkah umum untuk enkripsi menggunakan Playfair Cipher :

- **Membentuk Matriks Kunci**

Matriks kunci Playfair berukuran 5x5 dan berisi huruf-huruf unik dari kunci yang diberikan. Jika ada huruf yang berulang, hanya ambil huruf pertama dan abaikan yang lain. Jika kunci mengandung huruf "J", seringkali dianggap sebagai huruf yang sama dengan "I".

- **Penggantian Huruf**

Pasangan huruf dalam teks diganti dengan aturan pertama yaitu jika keduanya berada dalam satu baris atau satu kolom, ganti dengan huruf yang berada di sebelah kanan (jika dalam baris) atau di bawah (jika dalam kolom). Aturan selanjutnya yaitu Jika keduanya membentuk suatu persegi panjang, ganti dengan huruf yang berada di sudut yang berlawanan dari persegi panjang tersebut.

- **Penanganan Huruf Tunggal**

Jika teks sederhana mengandung huruf tunggal atau ada huruf yang harus diabaikan (seperti "J" yang dianggap sama dengan "I"), tambahkan huruf palsu (biasanya "X") untuk membentuk pasangan huruf.

Hasil enkripsi dari Playfair Cipher adalah teks yang dihasilkan setelah menggantikan setiap pasangan huruf dalam teks sederhana dengan pasangan huruf yang dihasilkan dari matriks kunci.

b. Algoritma Columnar Transposition Cipher

Berikut adalah langkah-langkah untuk melakukan enkripsi menggunakan algoritma Columnar Transposition Cipher :

- **Pemilihan Kunci untuk Pembentukan Matriks**

Pilih kunci yang akan digunakan untuk mengatur ulang urutan karakter dalam teks sederhana. Urutan kolom dalam matriks akan diatur berdasarkan urutan huruf dalam kunci. Setelah itu, Susun huruf-huruf kunci dalam suatu matriks. Jumlah kolom matriks ini akan sama dengan panjang kunci, dan urutan huruf dalam kunci menentukan urutan kolom.

- **Proses Enkripsi**

Plainteks dimasukkan ke dalam matriks secara berurutan, membaca dari kiri ke kanan dan dari atas ke bawah. Jika matriks tidak terisi penuh, tambahkan karakter fiktif atau kosong untuk melengkapi matriks. Hasil Enkripsi Columnar Transposition Cipher merupakan hasil penggabungan kolom-kolom matriks yang dibaca sesuai urutan huruf dalam kunci.

c. Algoritma DES

Berikut adalah langkah-langkah untuk melakukan enkripsi menggunakan algoritma DES :

- **Persiapan Kunci**

Kunci yang asli sepanjang 64 bit diubah menjadi kunci setengah (56 bit) dengan menghilangkan setiap bit kedua setelah setiap 7 bit. Dari kunci setengah ini, dihasilkan 16 subkunci, masing-masing sepanjang 48 bit, yang akan digunakan pada setiap putaran DES.

- **Inisialisasi Permutasi Awal**

Sebelum putaran pertama, terhadap blok plainteks dilakukan permutasi awal (initial permutation atau IP). Tujuan permutasi awal adalah mengacak plainteks sehingga urutan

bit-bit di dalamnya berubah. Pengacakan dilakukan dengan menggunakan matriks permutasi awal berikut.

Tabel IP

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

- Pembagian Menjadi Blok Kiri dan Blok Kanan

Blok data hasil permutasi awal dibagi menjadi dua bagian, blok kiri (32 bit) dan blok kanan (32 bit).

- Putaran Enkripsi

Enkripsi DES terdiri dari 16 putaran identik, di mana setiap putaran menggunakan subkunci yang dihasilkan pada langkah pertama. Setiap putaran melibatkan beberapa operasi yaitu :

- Permutasi kompresi PC-1, yaitu ukuran 64-bit dikompresi menjadi 56-bit dengan membuang *parity bit*. Hasil kompresi lalu dipecah menjadi 2 yaitu C_0 dan D_0 .

Adapun tabel PC-1 adalah sebagai berikut :

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

- Left Shift Operation, yaitu pergeseran pada C_0 dan D_0 menggunakan tabel pergeseran bit 16 putaran. Adapun bentuk tabelnya adalah sebagai berikut :

Iterasi	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bit	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

- Penggabungan kembali C_0 dan D_0 , lalu diinput kedalam tabel PC-2 dan terjadi kompresi data 56-bit menjadi 48-bit. Adapun tabel PC-2 adalah sebagai berikut :

Tabel PC-2

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

- Fungsi F

Fungsi f diaplikasikan pada blok kanan pada setiap putaran. Fungsi ini melibatkan beberapa operasi yaitu :

- Blok kanan yang awalnya 32 bit diperluas menjadi 48 bit. Adapun tabel permutasi ekspansi adalah sebagai berikut :

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
32	1	2	3	4	5	4	5	6	7	8	9	8	9	10	11
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
12	13	12	13	14	15	16	17	18	17	18	19	20	21	20	21
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
22	23	24	25	24	25	26	27	28	29	28	29	30	31	32	1

- Hasil perluasan di-XOR dengan subkunci putaran.
- Substitusi, yaitu blok 48-bit hasil XOR di-substitusi menggunakan tabel substitusi S-Box. Adapun Tabel S-Box adalah sebagai berikut :

S_1 :

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2 :

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15

13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
----	---	----	---	---	----	---	---	----	---	---	----	---	---	----	---

S_3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4 :

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5 :

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	16
3	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6 :

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	13	1	7	6	0	8	13

S_7 :

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8 :

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

- Permutasi, dimana hasil substitusi di-substitusi kembali menggunakan tabel permutasi. Adapun Tabel Fungsi Permutasi adalah sebagai berikut :

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
16	7	20	21	29	12	28	17	1	15	23	26	5	18	31	10

17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

- Hasil permutasi di-XOR dengan blok kiri.
- Pertukaran Blok Kiri dan Blok Kanan
Setelah setiap putaran, blok kiri dan blok kanan ditukar. Blok kanan menjadi blok kiri untuk putaran berikutnya, dan sebaliknya.
- Permutasi Akhir
Setelah 16 putaran, blok kiri dan blok kanan digabungkan dan melewati permutasi akhir (FP) yang merupakan invers dari permutasi awal. Hasil akhir dari permutasi akhir adalah blok data yang telah dienkripsi, sepanjang 64 bit.

III. PERHITUNGAN MANUAL

Perhitungan Manual ditujukan untuk lebih memahami proses pembangkitan kunci pada algoritma Playfair Cipher dan DES serta proses enkripsi dan dekripsi pada algoritma Columnar Transposition Cipher yang dikombinasikan dengan DES. Adapun parameter yang digunakan adalah sebagai berikut :

A. Membangun Kunci dengan Playfair Cipher

Hal pertama yang harus dilakukan yaitu menentukan plainteks yang akan dijadikan kunci dan kunci untuk Playfair Cipher. Misalnya, diambil contoh :

Plainteks Kunci : himastik

Kunci Pembangkit : faishal

Selanjutnya, plainteks kunci dan kunci pesan akan ditulis dalam bentuk matriks dengan panjang tetap. Baris 1 diisi dengan kunci pesan dilanjutkan baris 2 dan seterusnya akan diisi dengan plainteks kunci. Jika plainteks kunci lebih panjang dari kunci pesan, maka plainteks kunci tersebut ditulis ke baris selanjutnya hingga plainteks kunci tersebut selesai ditulis. Untuk kolom dan baris matriks yang belum terisi akan diisi dengan null atau dibiarkan kosong atau ditempatkan oleh suatu karakter. Untuk plainteks kunci akan dibaca dari pesan paling atas kiri sampai paling bawah kanan yang berurutan berdasarkan kunci pesan yang dimiliki sehingga didapat hasil pada tabel 1 sebagai berikut.

f	a	i	s	h
---	---	---	---	---

l	b	c	d	e
g	k	m	n	o
p	q	r	t	u
v	w	x	y	z

Berdasarkan tabel 1, didapatkan Hasil Enkripsi Plainteks Playfair Cipher yaitu “fskidyam”.

B. Kombinasi Kunci dengan Algoritma DES

1. Menentukan Plainteks Pesan dan Kunci Pesan

Plainteks : fskidyam

Kunci Pesan : sempurna

2. Mengubah Plainteks Kunci dan Kunci Pesan menjadi bilangan biner.

Kunci

Teks	Hexa	Biner							
f	66	1	2	3	4	5	6	7	8
		0	1	1	0	0	1	1	0
Teks	Hexa	Biner							
s	73	1	2	3	4	5	6	7	8
		0	1	1	1	0	0	1	1
Teks	Hexa	Biner							
k	6B	1	2	3	4	5	6	7	8
		0	1	1	0	1	0	1	1
Teks	Hexa	Biner							
i	69	1	2	3	4	5	6	7	8
		0	1	1	0	1	0	0	1
Teks	Hexa	Biner							
d	64	1	2	3	4	5	6	7	8
		0	1	1	0	0	1	0	0
Teks	Hexa	Biner							
y	79	1	2	3	4	5	6	7	8
		0	1	1	1	1	0	0	1
Teks	Hexa	Biner							
a	61	1	2	3	4	5	6	7	8
		0	1	1	0	0	0	0	1
Teks	Hexa	Biner							
m	6D	1	2	3	4	5	6	7	8
		0	1	1	0	1	1	0	1

Pesan

Teks	Hexa	Biner							
s	73	1	2	3	4	5	6	7	8
Teks	Hexa	Biner							
e	65	1	2	3	4	5	6	7	8
		0	1	1	0	0	1	0	1
Teks	Hexa	Biner							
m	6D	1	2	3	4	5	6	7	8
		0	1	1	0	1	1	0	1
Teks	Hexa	Biner							
p	70	1	2	3	4	5	6	7	8
		0	1	1	1	0	0	0	0
Teks	Hexa	Biner							
u	75	1	2	3	4	5	6	7	8
		0	1	1	1	0	1	0	1
Teks	Hexa	Biner							
r	72	1	2	3	4	5	6	7	8
		0	1	1	1	0	0	1	0
Teks	Hexa	Biner							
n	6E	1	2	3	4	5	6	7	8
		0	1	1	0	1	1	1	0
Teks	Hexa	Biner							
a	61	1	2	3	4	5	6	7	8
		0	1	1	0	0	0	0	1

3. Initial Permutation (IP) pada Plainteks Pesan

Plainteks (X)									IP(X)							
1	2	3	4	5	6	7	8	→	1	2	3	4	5	6	7	8
0	1	1	0	0	1	1	0		1	1	1	1	1	1	1	1
9	10	11	12	13	14	15	16	→	9	10	11	12	13	14	15	16
0	1	1	1	0	0	1	1		0	0	1	0	0	0	1	0
17	18	19	20	21	22	23	24	→	17	18	19	20	21	22	23	24
0	1	1	0	1	0	1	1		1	0	0	1	0	0	0	1
25	26	27	28	29	30	31	32	→	25	26	27	28	29	30	31	32
0	1	1	0	1	0	0	1		1	1	1	0	1	1	1	0
33	34	35	36	37	38	39	40	→	33	34	35	36	37	38	39	40
0	1	1	0	0	1	0	0		0	0	0	0	0	0	0	0
41	42	43	44	45	46	47	48	→	41	42	43	44	45	46	47	48
0	1	1	1	1	0	0	1		1	1	1	1	1	1	1	1
49	50	51	52	53	54	55	56	→	49	50	51	52	53	54	55	56
0	1	1	0	0	0	0	1		1	0	1	0	1	1	0	0
57	58	59	60	61	62	63	64	→	57	58	59	60	61	62	63	64
0	1	1	0	1	1	0	1		0	0	0	0	0	1	1	1

Hasil IP(X) :

11111111 00100010 10010001 11101110 00000000 11111111 10101100 00000111

Selanjutnya bit pada IP(X) dipecah menjadi 2 :

L_0 : 11111111 00100010 10010001 11101110

R_0 : 00000000 11111111 10101100 00000111

4. Generate Kunci Menggunakan Tabel Permutasi Kompresi PC-1

Kompresi 64-bit menjadi 56-bit dengan membuang 1 bit (*parity bit*) tiap blok kunci.

Kunci								Output							
1	2	3	4	5	6	7	8	→	1	2	3	4	5	6	7
0	1	1	1	0	0	1	1	→	0	0	0	0	0	0	0
9	10	11	12	13	14	15	16	→	8	9	10	11	12	13	14
0	1	1	0	0	1	0	1	→	0	1	1	1	1	1	1
17	18	19	20	21	22	23	24	→	15	16	17	18	19	20	21
0	1	1	0	1	1	0	1	→	1	1	1	1	1	1	1
25	26	27	28	29	30	31	32	→	22	23	24	25	26	27	28
0	1	1	1	0	0	0	0	→	1	1	1	0	0	1	1
33	34	35	36	37	38	39	40	→	29	30	31	32	33	34	35
0	1	1	1	0	1	0	1	→	0	1	1	0	0	0	0

41	42	43	44	45	46	47	48	→	36	37	38	39	40	41	42
0	1	1	1	0	0	1	0		1	0	1	0	1	0	1
49	50	51	52	53	54	55	56	→	43	44	45	46	47	48	49
0	1	1	0	1	1	1	0		1	0	0	1	0	0	0
57	58	59	60	61	62	63	64	→	50	51	52	53	54	55	56
0	1	1	0	0	0	1			1	0	0	1	0	0	1

Hasil $C_0 D_0 =$

0000000 0111111 1111111 1110011 0110000 1010101 1001000 1001001

Selanjutnya bit pada $C_0 D_0$ dipecah menjadi 2.

C_0 : 0000000 0111111 1111111 1110011

D_0 : 0110000 1010101 1001000 1001001

5. *Left Shift Operation*

Melakukan pergeseran pada C_0 dan D_0 menggunakan tabel pergeseran bit 16 putaran.

$C_0 D_0 =$ 0000000 0111111 1111111 1110011 0110000 1010101 1001000 1001001

C_0	=	0000000 0111111 1111111	D_0	=	0110000 1010101 1001000
		1110011			1001001
C_1	=	0000000 1111111 1111111	D_1	=	1100001 0101011 0010001
		1100110			0010010
C_2	=	0000001 1111111 1111111	D_2	=	1000010 1010110 0100010
		1001100			0100101
C_3	=	0000111 1111111 1111110	D_3	=	0001010 1011001 0001001
		0110000			0010110
C_4	=	0011111 1111111 1111001	D_4	=	0101010 1100100 0100100
		1000000			1011000
C_5	=	1111111 1111111 1100110	D_5	=	0101011 0010001 0010010
		0000000			1100001
C_6	=	1111111 1111111 0011000	D_6	=	0101100 1000100 1001011
		0000011			0000101
C_7	=	1111111 1111100 1100000	D_7	=	0110010 0010010 0101100
		0001111			0010101
C_8	=	1111111 1110011 0000000	D_8	=	1001000 1001001 0110000
		0111111			1010101
C_9	=	1111111 1100110 0000000	D_9	=	0010001 0010010 1100001
		1111111			0101011
C_{10}	=	1111111 0011000 0000011	D_{10}	=	1000100 1001011 0000101
		1111111			0101100
C_{11}	=	1111100 1100000 0001111	D_{11}	=	0010010 0101100 0010101
		1111111			0110010
C_{12}	=	1110011 0000000 0111111	D_{12}	=	1001001 0110000 1010101
		1111111			1001000

$$\begin{array}{ll}
C_{13} = 1001100\ 0000001\ 1111111 & D_{13} = 0100101\ 1000010\ 1010110 \\
& 1111111 & 0100010 \\
C_{14} = 0110000\ 0000111\ 1111111 & D_{14} = 0010110\ 0001010\ 1011001 \\
& 1111110 & 0001001 \\
C_{15} = 1000000\ 0011111\ 1111111 & D_{15} = 1011000\ 0101010\ 1100100 \\
& 1111001 & 0100100 \\
C_{16} = 0000000\ 0111111\ 1111111 & D_{16} = 0110000\ 1010101\ 1001000 \\
& 1110011 & 1001001
\end{array}$$

Setiap hasil putaran digabungkan kembali menjadi C_iD_i dan diinput kedalam tabel Permutation Compression 2 (PC-2) dan terjadi kompresi data C_iD_i 56-bit menjadi C_iD_i 48-bit.

Berikut hasil outputnya :

$$\begin{array}{ll}
C_1D_1 & = 0000000\ 1111111\ 1111111\ 1100110\ 1100001\ 0101011\ 0010001\ 0010010 \\
K_1 & = 111000\ 001011\ 111011\ 101110\ 110101\ 100100\ 011000\ 010010
\end{array}$$

$$\begin{array}{ll}
C_2D_2 & = 0000001\ 1111111\ 1111111\ 1001100\ 1000010\ 1010110\ 0100010\ 0100101 \\
K_2 & = 111000\ 001011\ 011011\ 110110\ 100000\ 011001\ 100110\ 000110
\end{array}$$

$$\begin{array}{ll}
C_3D_3 & = 0000111\ 1111111\ 1111110\ 0110000\ 0001010\ 1011001\ 0001001\ 0010110 \\
K_3 & = 111101\ 001101\ 111001\ 110110\ 010001\ 000000\ 011010\ 110101
\end{array}$$

$$\begin{array}{ll}
C_4D_4 & = 0011111\ 1111111\ 1111001\ 1000000\ 0101010\ 1100100\ 0100100\ 1011000 \\
K_4 & = 111001\ 101111\ 001101\ 110010\ 010110\ 110000\ 100011\ 001101
\end{array}$$

$$\begin{array}{ll}
C_5D_5 & = 1111111\ 1111111\ 1100110\ 0000000\ 0101011\ 0010001\ 0010010\ 1100001 \\
K_5 & = 101011\ 101101\ 011101\ 110111\ 000000\ 101101\ 000110\ 011001
\end{array}$$

$$\begin{array}{ll}
C_6D_6 & = 1111111\ 1111111\ 0011000\ 0000011\ 0101100\ 1000100\ 1001011\ 0000101 \\
K_6 & = 111011\ 110101\ 001101\ 011011\ 000000\ 110011\ 010100\ 100101
\end{array}$$

$$\begin{array}{ll}
C_7D_7 & = 1111111\ 1111100\ 1100000\ 0001111\ 0110010\ 0010010\ 0101100\ 0010101 \\
K_7 & = 001011\ 111101\ 001111\ 111001\ 111010\ 100000\ 100110\ 100000
\end{array}$$

$$\begin{aligned} C_8 D_8 &= 1111111 1110011 0000000 0111111 1001000 1001001 0110000 1010101 \\ K_8 &= 100111 110101 100111 011011 010000 000100 101100 011111 \end{aligned}$$

$$\begin{aligned} C_9 D_9 &= 1111111 1100110 0000000 1111111 0010001 0010010 1100001 0101011 \\ K_9 &= 000111 110100 101111 011011 101001 001000 110101 000000 \end{aligned}$$

$$\begin{aligned} C_{10} D_{10} &= 1111111 0011000 0000011 1111111 1000100 1001011 0000101 0101100 \\ K_{10} &= 001111 110111 100110 011101 100010 001010 011001 010110 \end{aligned}$$

$$\begin{aligned} C_{11} D_{11} &= 1111100 1100000 0001111 1111111 0010010 0101100 0010101 0110010 \\ K_{11} &= 000111 110010 110111 001101 011111 011100 011010 000000 \end{aligned}$$

$$\begin{aligned} C_{12} D_{12} &= 1110011 0000000 0111111 1111111 1001001 0110000 1010101 1001000 \\ K_{12} &= 010110 110110 110010 111101 000110 000100 010001 001011 \end{aligned}$$

$$\begin{aligned} C_{13} D_{13} &= 1001100 0000001 1111111 1111111 0100101 1000010 1010110 0100010 \\ K_{13} &= 110111 011010 110110 101100 100011 101111 000000 000100 \end{aligned}$$

$$\begin{aligned} C_{14} D_{14} &= 0110000 0000111 1111111 1111110 0010110 0001010 1011001 0001001 \\ K_{14} &= 110100 101010 111010 101111 101000 000110 011111 100000 \end{aligned}$$

$$\begin{aligned} C_{15} D_{15} &= 1000000 0011111 1111111 1111001 1011000 0101010 1100100 0100100 \\ K_{15} &= 111110 011011 111000 100110 101110 001000 101000 000011 \end{aligned}$$

$$\begin{aligned} C_{16} D_{16} &= 0000000 0111111 1111111 1110011 0110000 1010101 1001000 1001001 \\ K_{16} &= 111100 011011 111000 101110 001000 110000 000101 111100 \end{aligned}$$

6. Mengekspansi data

Data diekspansi R_{i-1} 32bit menjadi 48bit sebanyak 16 kali putaran dengan nilai perputaran $1 \leq i \leq 16$ menggunakan Tabel Ekspansi (E). Hasil $E(R_{i-1})$ kemudian di XOR dengan K_i dan menghasilkan vektor matriks A_i . Kemudian, vektor matriks

A_i disubstitusikan ke 8 buah S-box (substitution box), Dimana blok ke-1 disubstitusikan ke S_1 , blok ke-2 disubstitusikan ke S_2 , hingga blok ke-8 disubstitusikan ke S_8 . Setiap blok dari A_i dipisahkan menjadi 2 blok, yaitu:

- Blok pertama dan terakhir (Sebagai pembanding baris)
- Blok kedua hingga kelima (Sebagai pembanding kolom)

Selanjutnya bandingkan dengan memeriksa diantara keduanya dan disesuaikan dengan tabel S-box yang akan menghasilkan output B_i 32bit. Setelah didapat vektor B_i , lalu permutasikan bit vektor B_i dengan tabel P-Box, lalu kelompokkan menjadi 4 blok Dimana tiap-tiap blok memiliki 32bit data. Hasil $P(B_i)$ di XOR kan dengan untuk mendapatkan nilai R_i . Sedangkan nilai L_i sendiri diperoleh dari nilai R_{i-1} dengan $1 \leq i \leq 16$. Lakukan langkah tersebut hingga mendapatkan R_{16} dan L_{16} .

Iterasi 1

```

E(R0) = 100000 000001 011111 111111 110101 011000 000000 001110
K1    = 111000 001011 111011 101110 110101 100100 011000 010010
.....XOR
A1    = 011000 001010 100100 010001 000000 111100 011000 011100

B1     = 0101 1011 0100 0100 0010 1011 0101 1100
P(B1)  = 01011010 00111001 11110000 10100010

P(B1)  = 01011010 00111001 11110000 10100010
L0     = 11111111 00100010 10010001 11101110
.....XOR
R1     = 10100101 00011011 01100001 01001100

```

Iterasi 2

```

E(R1) = 010100 001010 100011 110110 101100 000010 101001 011001
K2    = 111000 001011 011011 110110 100000 011001 100110 000110
.....XOR
A2    = 101100 000001 111000 000000 001100 011011 001111 011111

B2     = 0010 0011 0101 0111 1011 1011 1010 0010
P(B2)  = 11110101 01100011 01110110 10000001

P(B2)  = 11110101 01100011 01110110 10000001
L1     = 00000000 11111111 10101100 00000111
.....XOR
R2     = 11110101 10011100 11011010 10000110

```

Iterasi 3

$E(R_2)$ = 011110 101011 110011 111001 011011 110101 010000 001101
 K_3 = 111101 001101 111001 110110 010001 000000 011010 110101
.....XOR
 A_3 = 100011 100110 001010 001111 001010 110101 001010 111000

 B_3 = 1100 1011 0011 0011 1010 0001 0000 1111

 $P(B_3)$ = 11001101 11001010 11101000 10100100

 $P(B_3)$ = 11001101 11001010 11101000 10100100
 L_2 = 10100101 00011011 01100001 01001100
.....XOR
 R_3 = 01101000 11010001 10001001 11101000

Iterasi 4

$E(R_3)$ = 001101 010001 011010 100011 110001 010011 111101 010000
 K_4 = 111001 101111 001101 110010 010110 110000 100011 001101
.....XOR
 A_4 = 110100 111110 010111 010001 100111 100011 011110 011101

 B_4 = 1001 1111 1110 0100 0111 0011 0001 1001
 $P(B_4)$ = 01101010 10101101 01111001 10010110

 $P(B_4)$ = 01101010 10101101 01111001 10010110
 L_3 = 11110101 10011100 11011010 10000110
.....XOR
 R_4 = 10011111 00110001 10100011 00010000

Iterasi 5

$E(R_4)$ = 010011 111110 100110 100011 110100 000110 100010 100001
 K_5 = 101011 101101 011101 110111 000000 101101 000110 011001
.....XOR
 A_5 = 111000 010011 111011 010100 110100 101011 100100 111000

 B_5 = 0011 0000 0101 1000 1100 0101 1011 1111
 $P(B_5)$ = 00001111 00000111 00101110 01101011

 $P(B_5)$ = 00001111 00000111 00101110 01101011
 L_4 = 01101000 11010001 10001001 11101000
.....XOR
 R_5 = 01100111 11010110 10100111 10000011

Iterasi 6

$E(R_5) = 101100\ 001111\ 111010\ 101101\ 010100\ 001111\ 110000\ 000110$
 $K_6 = 111011\ 110101\ 001101\ 011011\ 000000\ 110011\ 010100\ 100101$
.....XOR
 $A_6 = 010111\ 111010\ 110111\ 110110\ 010100\ 111100\ 100100\ 100011$

 $B_6 = 1011\ 0011\ 0011\ 1110\ 0011\ 1011\ 1011\ 0001$
 $P(B_6) = 01110110\ 11100000\ 01111110\ 11000111$

 $P(B_6) = 01110110\ 11100000\ 01111110\ 11000111$
 $L_5 = 10011111\ 00110001\ 10100011\ 00010000$
.....XOR
 $R_6 = 11101001\ 11010001\ 11011101\ 11010111$

Iterasi 7

$E(R_6) = 111101\ 010011\ 111010\ 100011\ 111011\ 111011\ 111010\ 101111$
 $K_7 = 001011\ 111101\ 001111\ 111001\ 111010\ 100000\ 100110\ 100000$
.....XOR
 $A_7 = 110110\ 101110\ 110101\ 011010\ 000001\ 011011\ 011100\ 001111$

 $B_7 = 0111\ 0001\ 1110\ 1100\ 1110\ 1011\ 0110\ 0100$
 $P(B_7) = 00010001\ 00110101\ 11110111\ 11100110$

 $P(B_7) = 00010001\ 00110101\ 11110111\ 11100110$
 $L_6 = 01100111\ 11010110\ 10100111\ 10000011$
.....XOR
 $R_7 = 01110110\ 11100011\ 01010000\ 01100101$

Iterasi 8

$E(R_7) = 101110\ 101101\ 011100\ 000110\ 101010\ 100000\ 001100\ 001010$
 $K_8 = 100111\ 110101\ 100111\ 011011\ 010000\ 000100\ 101100\ 011111$
.....XOR
 $A_8 = 001001\ 011000\ 111011\ 011101\ 111010\ 100100\ 100000\ 010101$

 $B_8 = 1110\ 1100\ 0101\ 1110\ 0011\ 1111\ 0001\ 0110$
 $P(B_8) = 00110110\ 11101011\ 10110010\ 11111000$

 $P(B_8) = 00110110\ 11101011\ 10110010\ 11111000$
 $L_7 = 11101001\ 11010001\ 11011101\ 11010111$
.....XOR
 $R_8 = 11011111\ 00111010\ 01101111\ 00101111$

Iterasi 9

$E(R_8) = 111011\ 111110\ 100111\ 110100\ 001101\ 011110\ 100101\ 011111$
 $K_9 = 000111\ 110100\ 101111\ 011011\ 101001\ 001000\ 110101\ 000000$
.....XOR

$A_9 = 111100\ 001010\ 001000\ 101111\ 100100\ 010110\ 010000\ 011111$

$B_9 = 0101\ 1011\ 0110\ 1000\ 0001\ 0100\ 0011\ 0010$

$P(B_9) = 01100010\ 00001011\ 11000100\ 01001110$

$P(B_9) = 01100010\ 00001011\ 11000100\ 01001110$

$L_8 = 01110110\ 11100011\ 01010000\ 01100101$

.....XOR

$R_9 = 00010100\ 11101000\ 10010100\ 00101011$

Iterasi 10

$E(R_9) = 100010\ 101001\ 011101\ 010001\ 010010\ 101000\ 000101\ 010110$

$K_{10} = 001111\ 110111\ 100110\ 011101\ 100010\ 001010\ 011001\ 010110$

.....XOR

$A_{10} = 101101\ 011110\ 111011\ 001100\ 110000\ 100010\ 011100\ 000000$

$B_{10} = 0001\ 1010\ 0101\ 1001\ 1111\ 1110\ 0110\ 1101$

$P(B_{10}) = 11111101\ 00111101\ 00001100\ 11101010$

$P(B_{10}) = 11111101\ 00111101\ 00001100\ 11101010$

$L_9 = 11011111\ 00111010\ 01101111\ 00101111$

.....XOR

$R_{10} = 00100010\ 00000111\ 01100011\ 11000101$

Iterasi 11

$E(R_{10}) = 100100\ 000100\ 000000\ 001110\ 101100\ 000111\ 111000\ 001010$

$K_{11} = 000111\ 110010\ 110111\ 001101\ 011111\ 011100\ 011010\ 000000$

.....XOR

$A_{11} = 100011\ 110110\ 110111\ 000011\ 110011\ 011011\ 100010\ 001010$

$B_{11} = 1100\ 0110\ 0011\ 1000\ 1111\ 1011\ 0100\ 1111$

$P(B_{11}) = 01111101\ 10110110\ 10101000\ 11110100$

$P(B_{11}) = 01111101\ 10110110\ 10101000\ 11110100$

$L_{10} = 00010100\ 11101000\ 10010100\ 00101011$

.....XOR

$R_{11} = 01101001\ 01011110\ 00111100\ 11011111$

Iterasi 12

$E(R_{11}) = 101101\ 010010\ 101011\ 111100\ 000111\ 111001\ 011011\ 111110$

$K_{12} = 010110\ 110110\ 110010\ 111101\ 000110\ 000100\ 010001\ 001011$

.....XOR

$A_{12} = 111011\ 100100\ 011001\ 000001\ 000001\ 111101\ 001010\ 110101$

$B_{12} = 0000\ 0111\ 1100\ 1101\ 1110\ 1000\ 0000\ 1001$

$P(B_{12}) = 11011001\ 00000101\ 01011001\ 11010000$

$P(B_{12}) = 11011001\ 00000101\ 01011001\ 11010000$
 $L_{11} = 00100010\ 00000111\ 01100011\ 11000101$
.....XOR
 $R_{12} = 11111011\ 00000010\ 00111010\ 00010101$

Iterasi 13

$E(R_{12}) = 111111\ 110110\ 100000\ 000100\ 000111\ 110100\ 000010\ 101011$
 $K_{13} = 110111\ 011010\ 110110\ 101100\ 100011\ 101111\ 000000\ 000100$
.....XOR
 $A_{13} = 001000\ 101100\ 010110\ 101000\ 100100\ 011011\ 000010\ 101111$

$B_{13} = 0010\ 1101\ 0111\ 1100\ 0001\ 1011\ 1011\ 1101$
 $P(B_{13}) = 00111110\ 00101001\ 01111110\ 01110101$

$P(B_{13}) = 00111110\ 00101001\ 01111110\ 01110101$
 $L_{12} = 01101001\ 01011110\ 00111100\ 11011111$
.....XOR
 $R_{13} = 01010111\ 01110111\ 01000010\ 10101010$

Iterasi 14

$E(R_{13}) = 001010\ 101110\ 101110\ 101110\ 101000\ 000101\ 010101\ 010100$
 $K_{14} = 110100\ 101010\ 111010\ 101111\ 101000\ 000110\ 011111\ 100000$
.....XOR
 $A_{14} = 111110\ 000100\ 010100\ 000001\ 000000\ 000011\ 001010\ 110100$

$B_{14} = 0000\ 1000\ 1100\ 1101\ 0010\ 1111\ 0000\ 1010$
 $P(B_{14}) = 10011000\ 00101011\ 00110001\ 11001000$

$P(B_{14}) = 10011000\ 00101011\ 00110001\ 11001000$
 $L_{13} = 11111011\ 00000010\ 00111010\ 00010101$
.....XOR
 $R_{14} = 01100011\ 00101001\ 00001011\ 11011101$

Iterasi 15

$E(R_{14}) = 101100\ 000110\ 100101\ 010010\ 100001\ 010111\ 111011\ 111010$
 $K_{15} = 111110\ 011011\ 111000\ 100110\ 101110\ 001000\ 101000\ 000011$
.....XOR
 $A_{15} = 010010\ 011101\ 011101\ 110100\ 001111\ 011111\ 010011\ 111001$

$B_{15} = 1010\ 1011\ 1111\ 0011\ 0001\ 1000\ 0011\ 0011$
 $P(B_{15}) = 11110110\ 11001011\ 01001111\ 00000100$

$P(B_{15}) = 11110110\ 11001011\ 01001111\ 00000100$
 $L_{14} = 01010111\ 01110111\ 01000010\ 10101010$
.....XOR

$$R_{15} = 10100001 \ 10111100 \ 00001101 \ 10101110$$

Iterasi 16

$$E(R_{15}) = 010100 \ 000011 \ 110111 \ 111000 \ 000001 \ 011011 \ 110101 \ 011101$$

$$K_{16} = 111100 \ 011011 \ 111000 \ 101110 \ 001000 \ 110000 \ 000101 \ 111100$$

$$\dots\dots\dots \text{XOR}$$

$$A_{16} = 101000 \ 011000 \ 001111 \ 010110 \ 001001 \ 101011 \ 110000 \ 100001$$

$$B_{16} = 1101 \ 1100 \ 1010 \ 0101 \ 0100 \ 0101 \ 1010 \ 0010$$

$$P(B_{16}) = 10000000 \ 10001110 \ 10110101 \ 00011111$$

$$P(B_{16}) = 10000000 \ 10001110 \ 10110101 \ 00011111$$

$$L_{15} = 01100011 \ 00101001 \ 00001011 \ 11011101$$

$$\dots\dots\dots \text{XOR}$$

$$R_{16} = 11100011 \ 10100111 \ 10111110 \ 11000010$$

$$L_{16} = R_{15} = 10100001 \ 10111100 \ 00001101 \ 10101110$$

7. Menggabungkan R_{16} dan L_{16} dan dipermutasi dengan tabel IP^{-1}

Hasil yang didapatkan setelah digabungkan dan dipermutasi menggunakan tabel Inverse Initial Permutation (IP^{-1}) :

$$R_{16}L_{16} = 11100011 \ 10100111 \ 10111110 \ 11000010 \ 10100001 \ 10111100 \ 00001101 \ 10101110$$

Menghasilkan output:

$$\text{Cipher (dalam biner)} = 11011000 \ 01010111 \ 00111110 \ 00101110 \ 00100100 \ 11110110 \ 01000001 \ 11110111$$

atau

$$\text{Cipher (dalam hexa)} = D8 \ 57 \ 3E \ 2E \ 24 \ F6 \ 41 \ F7$$

$$\text{Cipher (dalam karakter)} = \text{IW>.\$} \div A,$$

C. Enkripsi Kunci dengan Columnar Transposition Cipher

Hal pertama yang harus dilakukan yaitu menentukan Plainteks untuk Columnar Transposition Cipher. Kunci yang digunakan untuk enkripsi pesan yaitu kunci yang sudah dibangkitkan menggunakan Playfair Cipher dan DES. Misal diambil contoh :

$$\text{Kunci pesan} : \text{IW>.\$} \div A,$$

$$\text{Plainteks Pesan} : \text{faishali}$$

Selanjutnya plainteks dan kunci pesan ditulis dalam bentuk matriks dengan panjang tetap. Baris 1 diisi dengan kunci pesan, baris 2 dan seterusnya diisi dengan plainteks pesan. Jika plainteks pesan lebih panjang dari kunci pesan, maka plainteks pesan tersebut dilanjutkan ditulis ke baris selanjutnya hingga pesan plainteks tersebut selesai ditulis. Untuk kolom dan baris matriks yang belum terisi dibiarkan kosong atau ditempatkan oleh suatu karakter. Untuk plainteks pesan akan dibacakan dari pesan paling atas sampai paling bawah yang berurutan berdasarkan kunci pesan yang dimiliki sehingga didapatlah hasil pada Tabel berikut.

İ	W	>	.	\$	÷	A	,
f	a	i	s	h	a	l	i

Berdasarkan tabel diatas, didapatlah Cipherteks Columnar Transposition Cipher yaitu faishila.

D. Kombinasi Enkripsi Kunci dengan Algoritma DES

1. Menentukan Plainteks Pesan dan Kunci Pesan

Plainteks : faihsila

Kunci Pesan : İW>.\$ ÷A,

2. Mengubah Plainteks Kunci dan Kunci Pesan menjadi bilangan biner.

Kunci										Pesan									
Teks	Hexa	Biner								Teks	Hexa	Biner							
f	66	1	2	3	4	5	6	7	8	İ	D8	1	2	3	4	5	6	7	8
		0	1	1	0	0	1	1	0			1	1	0	1	1	0	0	
a	61	1	2	3	4	5	6	7	8	W	57	1	2	3	4	5	6	7	8
		0	1	1	0	0	0	1				0	1	0	1	0	1	1	1
i	69	1	2	3	4	5	6	7	8	>	3E	1	2	3	4	5	6	7	8
		0	1	1	0	1	0	0	1			0	0	1	1	1	1	1	0
h	68	1	2	3	4	5	6	7	8	.	2E	1	2	3	4	5	6	7	8
		0	1	1	0	1	0	0	0			0	0	1	0	1	1	1	0
s	73	1	2	3	4	5	6	7	8	\$	24	1	2	3	4	5	6	7	8
		0	1	1	1	0	0	1	1			0	0	1	0	0	1	0	0
i	69	1	2	3	4	5	6	7	8	÷	F6	1	2	3	4	5	6	7	8
		0	1	1	0	1	0	0	1			1	1	1	1	0	1	1	0

Teks	Hexa	Biner							
1	6C	1	2	3	4	5	6	7	8
		0	1	1	0	1	1	0	0
Teks	Hexa	Biner							
a	61	1	2	3	4	5	6	7	8
		0	1	1	0	0	0	0	1

Teks	Hexa	Biner							
A	41	1	2	3	4	5	6	7	8
		0	1	0	0	0	0	0	1
Teks	Hexa	Biner							
,	F7	1	2	3	4	5	6	7	8
		1	1	1	1	0	1	1	1

3. Initial Permutation (IP) pada Plainteks Pesan

Plainteks (X)									IP(X)							
1	2	3	4	5	6	7	8	→	1	2	3	4	5	6	7	8
0	1	1	0	0	1	1	0		1	1	1	1	1	1	1	1
9	10	11	12	13	14	15	16	→	9	10	11	12	13	14	15	16
0	1	0	1	0	1	1	1		0	0	0	1	0	0	0	0
17	18	19	20	21	22	23	24	→	17	18	19	20	21	22	23	24
0	1	1	0	1	0	0	1		0	1	0	0	0	0	0	1
25	26	27	28	29	30	31	32	→	25	26	27	28	29	30	31	32
0	1	1	0	1	0	0	0		1	0	1	1	0	1	1	0
33	34	35	36	37	38	39	40	→	33	34	35	36	37	38	39	40
0	1	1	1	0	0	1	1		0	0	0	0	0	0	0	0
41	42	43	44	45	46	47	48	→	41	42	43	44	45	46	47	48
0	1	1	0	1	0	0	1		1	1	1	1	1	1	1	1
49	50	51	52	53	54	55	56	→	49	50	51	52	53	54	55	56
0	1	1	0	1	1	0	0		0	1	1	0	1	1	0	0
57	58	59	60	61	62	63	64	→	57	58	59	60	61	62	63	64
0	1	1	0	0	0	0	1		0	0	0	1	0	0	0	1

Hasil IP(X) :

11111111 00010000 01000001 10110110 00000000 11111111 01101100 00010001

Selanjutnya bit pada IP(X) dipecah menjadi 2 :

L₀ : 11111111 00010000 01000001 10110110

R₀ : 00000000 11111111 01101100 00010001

4. Generate Kunci Menggunakan Tabel Permutasi Kompresi PC-1

Kompresi 64-bit menjadi 56-bit dengan membuang 1 bit (*parity bit*) tiap blok kunci.

Kunci									Output						
1	2	3	4	5	6	7	8	→	1	2	3	4	5	6	7
1	1	0	1	1	0	0	0		1	0	1	0	0	0	0
9	10	11	12	13	14	15	16	→	8	9	10	11	12	13	14
0	1	0	1	0	1	1	1		1	1	1	1	1	0	0

17	18	19	20	21	22	23	24	→	15	16	17	18	19	20	21
0	0	1	1	1	1	1	0		1	1	1	0	1	1	1
25	26	27	28	29	30	31	32	→	22	23	24	25	26	27	28
0	0	1	0	1	1	1	0		1	0	0	1	0	1	0
33	34	35	36	37	38	39	40	→	29	30	31	32	33	34	35
0	0	1	0	0	1	0	0		1	0	1	0	1	1	1
41	42	43	44	45	46	47	48	→	36	37	38	39	40	41	42
1	1	1	1	0	1	1	0		0	1	0	1	1	1	1
49	50	51	52	53	54	55	56	→	43	44	45	46	47	48	49
0	1	0	0	0	0	0	1		1	0	0	0	0	0	1
57	58	59	60	61	62	63	64	→	50	51	52	53	54	55	56
1	1	1	1	0	1	1	1		1	0	1	0	1	1	1

Hasil $C_0 D_0 =$

1010000 1111000 1110111 1001010 1010111 1101111 1100001 1010110

Selanjutnya bit pada $C_0 D_0$ dipecah menjadi 2.

C_0 : 1010000 1111000 1110111 1001010

D_0 : 1010111 0101111 1000001 1010111

5. Left Shift Operation

Melakukan pergeseran pada C_0 dan D_0 menggunakan tabel pergeseran bit 16 putaran.

$C_0 D_0 =$ 1010000 1111000 1110111 1001010 1010111 0101111 1000001 1010111

C_0	=	1010000 1111000 1110111	D_0	=	1010111 0101111 1000001
		1001010			1010111
C_1	=	0100001 1110001 1101111	D_1	=	0101110 1011111 0000011
		0010101			0101111
C_2	=	1000011 1100011 1011110	D_2	=	1011101 0111110 0000110
		0101010			1011110
C_3	=	0001111 0001110 1111001	D_3	=	1110101 1111000 0011010
		0101010			1111010
C_4	=	0111100 0111011 1100101	D_4	=	1010111 1100000 1101011
		0101000			1101011
C_5	=	1110001 1101111 0010101	D_5	=	1011111 0000011 0101111
		0100001			0101110
C_6	=	1000111 0111100 1010101	D_6	=	1111100 0001101 0111101
		0000111			0111010
C_7	=	0011101 1110010 1010100	D_7	=	1110000 0110101 1110101
		0011110			1101011
C_8	=	1110111 1001010 1010000	D_8	=	1000001 1010111 1010111
		1111000			0101111
C_9	=	1101111 0010101 0100001	D_9	=	0000011 0101111 0101110
		1110001			1011111

$C_{10} =$	0111100 1010101 0000111 1000111	$D_{10} =$	0001101 0111101 0111010 1111100
$C_{11} =$	1110010 1010100 0011110 0011101	$D_{11} =$	0110101 1110101 1101011 1110000
$C_{12} =$	1001010 1010000 1111000 1110111	$D_{12} =$	1010111 1010111 0101111 1000001
$C_{13} =$	0101010 1000011 1100011 1011110	$D_{13} =$	1011110 1011101 0111110 0000110
$C_{14} =$	0101010 0001111 0001110 1111001	$D_{14} =$	1111010 1110101 1111000 0011010
$C_{15} =$	0101000 0111100 0111011 1100101	$D_{15} =$	1101011 1010111 1100000 1101011
$C_{16} =$	1010000 1111000 1110111 1001010	$D_{16} =$	1010111 0101111 1000001 1010111

Setiap hasil putaran digabungkan kembali menjadi $C_i D_i$ dan diinput kedalam tabel Permutation Compression 2 (PC-2) dan terjadi kompresi data $C_i D_i$ 56-bit menjadi $C_i D_i$ 48-bit.

Berikut hasil outputnya :

$$C_1 D_1 = 0100001 \ 1110001 \ 1101111 \ 0010101 \ 0101110 \ 1011111 \ 0000011 \ 0101111$$

$$K_1 = 100100 \ 011011 \ 010011 \ 110101 \ 100001 \ 111011 \ 011111 \ 010101$$

$$C_2 D_2 = 0001111 \ 0001110 \ 1111001 \ 0101010 \ 1110101 \ 1111000 \ 0011010 \ 1111010$$

$$K_2 = 011001 \ 001110 \ 101100 \ 111010 \ 011101 \ 101111 \ 001001 \ 101110$$

$$C_3 D_3 = 0000111 \ 1111111 \ 1111110 \ 0110000 \ 0001010 \ 1011001 \ 0001001 \ 0010110$$

$$K_3 = 111101 \ 001101 \ 111001 \ 110110 \ 010001 \ 000000 \ 011010 \ 110101$$

$$C_4 D_4 = 0111100 \ 0111011 \ 1100101 \ 0101000 \ 1010111 \ 1100000 \ 1101011 \ 1101011$$

$$K_4 = 101001 \ 101011 \ 110100 \ 100011 \ 001101 \ 001011 \ 110111 \ 101110$$

$$C_5 D_5 = 1110001 \ 1101111 \ 0010101 \ 0100001 \ 1011111 \ 0000011 \ 0101111 \ 0101110$$

$$K_5 = 111010 \ 110010 \ 111001 \ 010011 \ 101011 \ 001011 \ 110011 \ 110011$$

$$C_6 D_6 = 1000111 \ 0111100 \ 1010101 \ 0000111 \ 1111100 \ 0001101 \ 0111101 \ 0111010$$

$$K_6 = 011011 \ 011111 \ 011010 \ 011000 \ 011011 \ 111110 \ 111001 \ 110011$$

$$\begin{aligned} C_7D_7 &= 0011101 \ 1110010 \ 1010100 \ 0011110 \ 1110000 \ 0110101 \ 1110101 \ 1101011 \\ K_7 &= 010101 \ 101001 \ 010111 \ 011010 \ 001111 \ 111100 \ 110101 \ 011010 \end{aligned}$$

$$\begin{aligned} C_8D_8 &= 1110111 \ 1001010 \ 1010000 \ 1111000 \ 1000001 \ 1010111 \ 1010111 \ 0101111 \\ K_8 &= 011111 \ 101100 \ 100001 \ 010011 \ 100011 \ 011101 \ 010101 \ 010110 \end{aligned}$$

$$\begin{aligned} C_9D_9 &= 1101111 \ 0010101 \ 0100001 \ 1110001 \ 0000011 \ 0101111 \ 0101110 \ 1011111 \\ K_9 &= 100111 \ 010111 \ 101100 \ 110001 \ 110111 \ 010001 \ 101111 \ 111000 \end{aligned}$$

$$\begin{aligned} C_{10}D_{10} &= 0111100 \ 1010101 \ 0000111 \ 1000111 \ 0001101 \ 0111101 \ 0111010 \ 1111100 \\ K_{10} &= 100001 \ 110011 \ 011111 \ 001101 \ 010100 \ 011111 \ 101001 \ 111001 \end{aligned}$$

$$\begin{aligned} C_{11}D_{11} &= 1110010 \ 1010100 \ 0011110 \ 0011101 \ 0110101 \ 1110101 \ 1101011 \ 1110000 \\ K_{11} &= 010110 \ 110101 \ 011011 \ 000101 \ 011100 \ 111011 \ 110000 \ 111100 \end{aligned}$$

$$\begin{aligned} C_{12}D_{12} &= 1001010 \ 1010000 \ 1111000 \ 1110111 \ 1010111 \ 1010111 \ 0101111 \ 1000001 \\ K_{12} &= 010110 \ 011101 \ 100111 \ 101000 \ 101010 \ 010011 \ 110110 \ 111110 \end{aligned}$$

$$\begin{aligned} C_{13}D_{13} &= 0101010 \ 1000011 \ 1100011 \ 1011110 \ 1011110 \ 1011101 \ 0111110 \ 0000110 \\ K_{13} &= 100100 \ 001110 \ 000111 \ 101111 \ 001011 \ 010111 \ 101010 \ 110111 \end{aligned}$$

$$\begin{aligned} C_{14}D_{14} &= 0101010 \ 0001111 \ 0001110 \ 1111001 \ 1111010 \ 1110101 \ 1111000 \ 0011010 \\ K_{14} &= 101100 \ 010100 \ 111100 \ 000111 \ 011101 \ 110100 \ 100011 \ 110111 \end{aligned}$$

$$\begin{aligned} C_{15}D_{15} &= 0101000 \ 0111100 \ 0111011 \ 1100101 \ 1101011 \ 1010111 \ 1100000 \ 1101011 \\ K_{15} &= 011000 \ 010011 \ 101110 \ 100101 \ 100001 \ 111000 \ 100111 \ 011111 \end{aligned}$$

$$\begin{aligned} C_{16}D_{16} &= 1010000 \ 1111000 \ 1110111 \ 1001010 \ 1010111 \ 0101111 \ 1000001 \ 1010111 \\ K_{16} &= 011010 \ 101011 \ 010001 \ 101100 \ 111101 \ 010010 \ 011110 \ 011010 \end{aligned}$$

6. Mengekspansi data

Data diekspansi R_{i-1} 32bit menjadi 48bit sebanyak 16 kali putaran dengan nilai perputaran $1 \leq i \leq 16$ menggunakan Tabel Ekspansi (E). Hasil $E(R_{i-1})$ kemudian di XOR dengan K_i dan menghasilkan vektor matriks A_i . Kemudian, vektor matriks A_i disubstitusikan ke 8 buah S-box (substitution box), Dimana blok ke-1 disubstitusikan ke S_1 , blok ke-2 disubstitusikan ke S_2 , hingga blok ke-8 disubstitusikan ke S_8 . Setiap blok dari A_i dipisahkan menjadi 2 blok, yaitu:

- Blok pertama dan terakhir (Sebagai pembanding baris)
- Blok kedua hingga kelima (Sebagai pembanding kolom)

Selanjutnya bandingkan dengan memeriksa diantara keduanya dan disesuaikan dengan tabel S-box yang akan menghasilkan output B_i 32bit. Setelah didapat vektor B_i , lalu permutasikan bit vektor B_i dengan tabel P-Box, lalu kelompokkan menjadi 4 blok Dimana tiap-tiap blok memiliki 32bit data. Hasil $P(B_i)$ di XOR kan dengan untuk mendapatkan nilai R_i . Sedangkan nilai L_i sendiri diperoleh dari nilai R_{i-1} dengan $1 \leq i \leq 16$. Lakukan langkah tersebut hingga mendapatkan R_{16} dan L_{16} .

Iterasi 1

```
E(R0) = 100000 000001 011111 111110 101101 011000 000010 100010
K1    = 100100 011011 010011 110101 100001 111011 011111 010101
.....XOR
A1    = 000100 011010 001100 001011 001101 100011 011101 110111

B1    = 1101 0000 1111 1111 1101 0011 1000 0000
P(B1) = 10100101 11100101 10110001 01000111

P(B1) = 10100101 11100101 10110001 01000111
L0    = 11111111 00010000 01000001 10110110
.....XOR
R1    = 01011010 11110101 11110000 11110001
```

Iterasi 2

```
E(R1) = 101011 110101 011110 101011 111110 100001 011110 100010
K2    = 110010 001100 110001 011110 111111 010011 001001 001011
.....XOR
A2    = 011001 111001 101111 110101 000001 110010 010101 111001

B2    = 1001 0000 0111 0101 1110 0000 0101 0011
P(B2) = 10000111 10010111 00011000 10000110
```

$P(B_2) = 10000111 \ 10010111 \ 00011000 \ 10000110$
 $L_1 = 10000111 \ 10010111 \ 00011000 \ 10000110$
.....XOR
 $R_2 = 10000111 \ 01101000 \ 01110100 \ 10010111$

Iterasi 3

$E(R_2) = 011110 \ 101011 \ 110011 \ 111001 \ 011011 \ 110101 \ 010000 \ 001101$
 $K_3 = 011001 \ 001110 \ 101100 \ 111010 \ 011101 \ 101111 \ 001001 \ 101110$
.....XOR
 $A_3 = 101001 \ 000000 \ 000001 \ 101010 \ 000011 \ 010110 \ 011011 \ 000000$

 $B_3 = 0100 \ 1111 \ 1101 \ 1011 \ 1011 \ 0100 \ 1111 \ 1101$

 $P(B_3) = 11101111 \ 01011001 \ 11001101 \ 11111001$

 $P(B_3) = 11101111 \ 01011001 \ 11001101 \ 11111001$
 $L_2 = 01011010 \ 11110101 \ 11110000 \ 11110001$
.....XOR
 $R_3 = 10110101 \ 10101100 \ 00111101 \ 00001000$

Iterasi 4

$E(R_3) = 010110 \ 101011 \ 110101 \ 011000 \ 000111 \ 111010 \ 100001 \ 010001$
 $K_4 = 101001 \ 101011 \ 110100 \ 100011 \ 001101 \ 001011 \ 110111 \ 101110$
.....XOR
 $A_4 = 111111 \ 000000 \ 000001 \ 111011 \ 001010 \ 010001 \ 010110 \ 110111$

 $B_4 = 1101 \ 1111 \ 1101 \ 0111 \ 1010 \ 0110 \ 0111 \ 0000$
 $P(B_4) = 11000111 \ 11111001 \ 11010101 \ 10011010$

 $P(B_4) = 11000111 \ 11111001 \ 11010101 \ 10011010$
 $L_3 = 10000111 \ 01101000 \ 01110100 \ 10010111$
.....XOR
 $R_4 = 01000000 \ 10010001 \ 10100001 \ 00001101$

Iterasi 5

$E(R_4) = 101000 \ 000001 \ 010010 \ 100011 \ 110100 \ 000010 \ 100001 \ 011010$
 $K_5 = 111010 \ 110010 \ 111001 \ 010011 \ 101011 \ 001011 \ 110011 \ 110011$
.....XOR
 $A_5 = 010010 \ 110011 \ 101011 \ 110000 \ 011111 \ 001000 \ 010110 \ 101101$

 $B_5 = 1010 \ 0110 \ 1001 \ 1111 \ 0110 \ 1001 \ 0111 \ 1000$
 $P(B_5) = 11011110 \ 11010100 \ 00110111 \ 11010000$

 $P(B_5) = 11011110 \ 11010100 \ 00110111 \ 11010000$
 $L_4 = 10110101 \ 10101100 \ 00111101 \ 00001000$
.....XOR

$$R_5 = 01101011 \ 01111000 \ 00001010 \ 11011000$$

Iterasi 6

$$E(R_5) = 001101 \ 010110 \ 101111 \ 110000 \ 000001 \ 010101 \ 011011 \ 110000$$

$$K_6 = 011011 \ 011111 \ 011010 \ 011000 \ 011011 \ 111110 \ 111001 \ 110011$$

$$\dots\dots\dots \text{XOR}$$

$$A_6 = 010110 \ 001001 \ 110101 \ 101000 \ 011010 \ 101011 \ 100000 \ 100011$$

$$B_6 = 1100 \ 1111 \ 1110 \ 1100 \ 0000 \ 0101 \ 0001 \ 0001$$

$$P(B_6) = 01000010 \ 10001001 \ 11111001 \ 01011100$$

$$P(B_6) = 01000010 \ 10001001 \ 11111001 \ 01011100$$

$$L_5 = 01000000 \ 10010001 \ 10100001 \ 00001101$$

$$\dots\dots\dots \text{XOR}$$

$$R_6 = 00000010 \ 00011000 \ 01011000 \ 01010001$$

Iterasi 7

$$E(R_6) = 100000 \ 000100 \ 000011 \ 110000 \ 001011 \ 110000 \ 001010 \ 100010$$

$$K_7 = 010101 \ 101001 \ 010111 \ 011010 \ 001111 \ 111100 \ 110101 \ 011010$$

$$\dots\dots\dots \text{XOR}$$

$$A_7 = 110101 \ 101101 \ 010100 \ 101010 \ 010100 \ 001100 \ 011111 \ 111001$$

$$B_7 = 0011 \ 0100 \ 1100 \ 1011 \ 0011 \ 0110 \ 0110 \ 0011$$

$$P(B_7) = 10100000 \ 01110011 \ 00001111 \ 11011010$$

$$P(B_7) = 10100000 \ 01110011 \ 00001111 \ 11011010$$

$$L_6 = 01101011 \ 01111000 \ 00001010 \ 11011000$$

$$\dots\dots\dots \text{XOR}$$

$$R_7 = 11001011 \ 00001011 \ 00000101 \ 00000010$$

Iterasi 8

$$E(R_7) = 011001 \ 010110 \ 100001 \ 010110 \ 100000 \ 001010 \ 100000 \ 000101$$

$$K_8 = 011111 \ 101100 \ 100001 \ 010011 \ 100011 \ 011101 \ 010101 \ 010110$$

$$\dots\dots\dots \text{XOR}$$

$$A_8 = 000110 \ 111010 \ 000000 \ 000101 \ 000011 \ 110111 \ 110101 \ 001011$$

$$B_8 = 0001 \ 0011 \ 1010 \ 1011 \ 1011 \ 0111 \ 0000 \ 0011$$

$$P(B_8) = 11100001 \ 01100010 \ 01101001 \ 11001110$$

$$P(B_8) = 11100001 \ 01100010 \ 01101001 \ 11001110$$

$$L_7 = 00000010 \ 00011000 \ 01011000 \ 01010001$$

$$\dots\dots\dots \text{XOR}$$

$$R_8 = 11100011 \ 01111010 \ 00110001 \ 10011111$$

Iterasi 9

$$E(R_8) = 111100 \ 000110 \ 101111 \ 110100 \ 000110 \ 100011 \ 110011 \ 111111$$

$K_9 = 100111\ 010111\ 101100\ 110001\ 110111\ 010001\ 101111\ 111000$
.....XOR
 $A_9 = 011011\ 010001\ 000011\ 000101\ 010001\ 110010\ 001100\ 000101$

 $B_9 = 0101\ 1100\ 0111\ 1011\ 0101\ 0000\ 1000\ 1101$
 $P(B_9) = 10101100\ 01001101\ 10001000\ 01110111$

 $P(B_9) = 10101100\ 01001101\ 10001000\ 01110111$
 $L_8 = 11001011\ 00001011\ 00000101\ 00000010$
.....XOR
 $R_9 = 01100111\ 01000110\ 10001101\ 01110101$

Iterasi 10

$E(R_9) = 101100\ 001110\ 101000\ 001101\ 010001\ 011010\ 101110\ 101010$
 $K_{10} = 100001\ 110011\ 011111\ 001101\ 010100\ 011111\ 101001\ 111001$
.....XOR
 $A_{10} = 001101\ 111101\ 110111\ 000000\ 000110\ 000101\ 001111\ 010011$

 $B_{10} = 1101\ 1110\ 0011\ 0111\ 0001\ 0100\ 1010\ 0101$
 $P(B_{10}) = 11100100\ 11001000\ 10011100\ 00111111$

 $P(B_{10}) = 11100100\ 11001000\ 10011100\ 00111111$
 $L_9 = 11100011\ 01111010\ 00110001\ 10011111$
.....XOR
 $R_{10} = 00000111\ 10110010\ 10101101\ 10100000$

Iterasi 11

$E(R_{10}) = 000000\ 00111\ 110110\ 100101\ 010101\ 011011\ 110100\ 000000$
 $K_{11} = 010110\ 110101\ 011011\ 000101\ 011100\ 111011\ 110000\ 111100$
.....XOR
 $A_{11} = 010110\ 110010\ 101101\ 100000\ 001101\ 100100\ 000101\ 111100$

 $B_{11} = 1100\ 1000\ 1000\ 1010\ 1101\ 1111\ 1011\ 0101$
 $P(B_{11}) = 00110011\ 11101100\ 10101101\ 01101001$

 $P(B_{11}) = 00110011\ 11101100\ 10101101\ 01101001$
 $L_{10} = 01100111\ 01000110\ 10001101\ 01110101$
.....XOR
 $R_{11} = 01010100\ 10101010\ 00100000\ 00011110$

Iterasi 12

$E(R_{11}) = 001010\ 101001\ 010101\ 010100\ 000100\ 000000\ 000011\ 111000$
 $K_{12} = 010110\ 011101\ 100111\ 101000\ 101010\ 010011\ 110110\ 111110$
.....XOR
 $A_{12} = 011100\ 110100\ 110010\ 111100\ 101110\ 011011\ 110101\ 000110$

B₁₂ = 0000 1100 0001 1000 1000 1011 0000 0100
P(B₁₂) = 00010101 00101000 00100000 01110000

P(B₁₂) = 00010101 00101000 00100000 01110000
L₁₁ = 00000111 10110010 10101101 10100000
.....XOR
R₁₂ = 00010010 10011010 10001101 11010000

Iterasi 13

E(R₁₂) = 000010 100101 010011 110101 010001 011011 111010 100000
K₁₃ = 100100 001110 000111 101111 001011 010111 101010 110111
.....XOR
A₁₃ = 100110 101011 010100 011010 111010 001110 000000 010111

B₁₃ = 1000 1111 1100 1100 0011 1000 0100 1011
P(B₁₃) = 01111000 10011011 01011001 11010000

P(B₁₃) = 01111000 10011011 01011001 11010000
L₁₂ = 01010100 10101010 00100000 00011100
.....XOR
R₁₃ = 00101100 00110001 01111001 11001100

Iterasi 14

E(R₁₃) = 000101 011000 000110 100010 101111 110011 111001 011000
K₁₄ = 101100 010100 111100 000111 011101 110100 100011 110111
.....XOR
A₁₄ = 101001 001100 111010 100101 111000 000111 010010 101111

B₁₄ = 0100 0011 1010 0000 0110 0010 1100 1101
P(B₁₄) = 01001000 00110100 11001001 10100101

P(B₁₄) = 01001000 00110100 11001001 10100101
L₁₃ = 00010010 10011010 10001101 11010000
.....XOR
R₁₄ = 01011010 10101110 01000100 01110101

Iterasi 15

E(R₁₄) = 101011 110101 010101 011100 001000 001000 001110 101010
K₁₅ = 011000 010011 101110 100101 100001 111000 100111 011111
.....XOR
A₁₅ = 110011 100110 111011 111001 101101 110100 101001 110111

B₁₅ = 1011 1011 0101 1100 0010 0100 0001 0000
P(B₁₅) = 01000110 10001001 01010010 11001010

P(B₁₅) = 01000110 10001001 01010010 11001010

$L_{14} = 00101100\ 00110001\ 01111001\ 11001100$
 $\dots\dots\dots$ XOR
 $R_{15} = 01101010\ 10111000\ 00101011\ 00000110$

Iterasi 16

$E(R_{15}) = 001101\ 010101\ 010111\ 110000\ 000101\ 010110\ 100000\ 001100$
 $K_{16} = 011010\ 101011\ 010001\ 101100\ 111101\ 010010\ 011110\ 011010$
 $\dots\dots\dots$ XOR
 $A_{16} = 010111\ 111110\ 000110\ 011100\ 111000\ 000100\ 011010\ 010010$

$B_{16} = 1011\ 1111\ 1110\ 0100\ 0110\ 1010\ 1010\ 1001$
 $P(B_{16}) = 01011000\ 10101101\ 01011111\ 10010111$

$P(B_{16}) = 01011000\ 10101101\ 01011111\ 10010111$
 $L_{15} = 01011010\ 10101110\ 01000100\ 01110101$
 $\dots\dots\dots$ XOR
 $R_{16} = 11111011\ 10010101\ 00000010\ 10010111$

$L_{16} = R_{15} = 11110000\ 10100011\ 00110100\ 01010011$

7. Menggabungkan R_{16} dan L_{16} dan dipermutasi dengan tabel IP^{-1}

Hasil yang didapatkan setelah digabungkan dan dipermutasi menggunakan tabel Inverse Initial Permutation (IP^{-1}) :

$R_{16}L_{16} = 11111011\ 10010101\ 00000010\ 10010111\ 11110000\ 10100011\ 00110100$
 01010011

Menghasilkan output:

Cipher (dalam biner) = $01110011\ 01100111\ 00011001\ 01000000\ 11011011\ 11101000$
 $11000010\ 11110001$

atau

Cipher (dalam hexa) = 73 67 19 40 DB E8 C2 F1

Cipher (dalam karakter) = sg↓@█p±

E. Dekripsi Kunci dengan Algoritma DES

1. Menentukan Cipherteks Pesan dan Kunci Pesan

Cipherteks : ĨW>.\$÷A,

Kunci Pesan : sempurna

2. Mengubah Plainteks Kunci dan Kunci Pesan menjadi bilangan biner.

Kunci

Teks	Hexa	Biner							
İ	D8	1	2	3	4	5	6	7	8
		1	1	0	1	1	0	0	0
Teks	Hexa	Biner							
W	57	1	2	3	4	5	6	7	8
		0	1	0	1	0	1	1	1
Teks	Hexa	Biner							
>	3E	1	2	3	4	5	6	7	8
		0	0	1	1	1	1	1	0
Teks	Hexa	Biner							
.	2E	1	2	3	4	5	6	7	8
		0	0	1	0	1	1	1	0
Teks	Hexa	Biner							
\$	24	1	2	3	4	5	6	7	8
		0	0	1	0	0	1	0	0
Teks	Hexa	Biner							
÷	F6	1	2	3	4	5	6	7	8
		1	1	1	1	0	1	1	0
Teks	Hexa	Biner							
A	41	1	2	3	4	5	6	7	8
		0	1	0	0	0	0	0	1
Teks	Hexa	Biner							
,	F7	1	2	3	4	5	6	7	8
		1	1	1	1	0	1	1	1

Pesan

Teks	Hexa	Biner							
s	73	1	2	3	4	5	6	7	8
		0	1	1	1	0	0	1	1
Teks	Hexa	Biner							
e	65	1	2	3	4	5	6	7	8
		0	1	1	0	0	1	0	1
Teks	Hexa	Biner							
m	6D	1	2	3	4	5	6	7	8
		0	1	1	0	1	1	0	1
Teks	Hexa	Biner							
p	70	1	2	3	4	5	6	7	8
		0	1	1	1	0	0	0	0
Teks	Hexa	Biner							
u	75	1	2	3	4	5	6	7	8
		0	1	1	1	0	1	0	1
Teks	Hexa	Biner							
r	72	1	2	3	4	5	6	7	8
		0	1	1	1	0	0	1	0
Teks	Hexa	Biner							
n	6E	1	2	3	4	5	6	7	8
		0	1	1	0	1	1	1	0
Teks	Hexa	Biner							
a	61	1	2	3	4	5	6	7	8
		0	1	1	0	0	0	0	1

3. Initial Permutation (IP) pada Cipherteks Pesan

Cipherteks (X)									IP(X)							
1	2	3	4	5	6	7	8	→	1	2	3	4	5	6	7	8
1	1	0	1	1	0	0	0		1	1	1	0	0	0	1	1
9	10	11	12	13	14	15	16	→	9	10	11	12	13	14	15	16
0	1	0	1	0	1	1	1		1	0	1	0	0	1	1	1
17	18	19	20	21	22	23	24	→	17	18	19	20	21	22	23	24
0	0	1	1	1	1	1	0		1	0	1	1	1	1	1	0
25	26	27	28	29	30	31	32	→	25	26	27	28	29	30	31	32
0	0	1	0	1	1	1	0		1	1	0	0	0	0	1	0
33	34	35	36	37	38	39	40	→	33	34	35	36	37	38	39	40
0	0	1	0	0	1	0	0		1	0	1	0	0	0	0	1
41	42	43	44	45	46	47	48	→	41	42	43	44	45	46	47	48
1	1	1	1	0	1	1	0		1	0	1	1	1	1	0	0

49	50	51	52	53	54	55	56	→	49	50	51	52	53	54	55	56
0	1	0	0	0	0	0	1		0	0	0	0	1	1	0	1
57	58	59	60	61	62	63	64	→	57	58	59	60	61	62	63	64
1	1	1	1	0	1	1	1		1	0	1	0	1	1	1	0

Hasil IP(X) :

01100011 00100110 10111111 01100011 00100001 10111100 10001101 00101111

Selanjutnya bit pada IP(X) dipecah menjadi 2 :

L₀ : 11100011 10100111 10111110 11000010

R₀ : 10100001 10111100 00001101 10101110

4. Generate Kunci Menggunakan Tabel Permutasi Kompresi PC-1

Kompresi 64-bit menjadi 56-bit dengan membuang 1 bit (*parity bit*) tiap blok kunci.

Kunci								→	Output						
1	2	3	4	5	6	7	8		1	2	3	4	5	6	7
0	1	1	1	0	0	1	1	→	0	0	0	0	0	0	0
9	10	11	12	13	14	15	16	→	8	9	10	11	12	13	14
0	1	1	0	0	1	0	1	→	0	1	1	1	1	1	1
17	18	19	20	21	22	23	24	→	15	16	17	18	19	20	21
0	1	1	0	1	1	0	1	→	1	1	1	1	1	1	1
25	26	27	28	29	30	31	32	→	22	23	24	25	26	27	28
0	1	1	1	0	0	0	0	→	1	1	1	0	0	1	1
33	34	35	36	37	38	39	40	→	29	30	31	32	33	34	35
0	1	1	1	0	1	0	1	→	0	1	1	0	0	0	0
41	42	43	44	45	46	47	48	→	36	37	38	39	40	41	42
0	1	1	1	0	0	1	0	→	1	0	1	0	1	0	1
49	50	51	52	53	54	55	56	→	43	44	45	46	47	48	49
0	1	1	0	1	1	1	0	→	1	0	0	1	0	0	0
57	58	59	60	61	62	63	64	→	50	51	52	53	54	55	56
0	1	1	0	0	0	0	1	→	1	0	0	1	0	0	1

Hasil C₀ D₀ =

0000000 0111111 1111111 1110011 0110000 1010101 1001000 1001001

Selanjutnya bit pada C₀ D₀ dipecah menjadi 2.

C₀ : 0000000 0111111 1111111 1110011

D₀ : 0110000 1010101 1001000 1001001

5. Left Shift Operation

Melakukan pergeseran pada C₀ dan D₀ menggunakan tabel pergeseran bit 16 putaran.

$C_0 D_0 = 0000000 0111111 1111111 1110011 0110000 1010101 1001000 1001001$

$C_0 = 0000000 0111111 1111111$	$D_0 = 0110000 1010101 1001000$
1110011	1001001
$C_1 = 0000000 1111111 1111111$	$D_1 = 1100001 0101011 0010001$
1100110	0010010
$C_2 = 0000001 1111111 1111111$	$D_2 = 1000010 1010110 0100010$
1001100	0100101
$C_3 = 0000111 1111111 1111110$	$D_3 = 0001010 1011001 0001001$
0110000	0010110
$C_4 = 0011111 1111111 1111001$	$D_4 = 0101010 1100100 0100100$
1000000	1011000
$C_5 = 1111111 1111111 1100110$	$D_5 = 0101011 0010001 0010010$
0000000	1100001
$C_6 = 1111111 1111111 0011000$	$D_6 = 0101100 1000100 1001011$
0000011	0000101
$C_7 = 1111111 1111100 1100000$	$D_7 = 0110010 0010010 0101100$
0001111	0010101
$C_8 = 1111111 1110011 0000000$	$D_8 = 1001000 1001001 0110000$
0111111	1010101
$C_9 = 1111111 1100110 0000000$	$D_9 = 0010001 0010010 1100001$
1111111	0101011
$C_{10} = 1111111 0011000 0000011$	$D_{10} = 1000100 1001011 0000101$
1111111	0101100
$C_{11} = 1111100 1100000 0001111$	$D_{11} = 0010010 0101100 0010101$
1111111	0110010
$C_{12} = 1110011 0000000 0111111$	$D_{12} = 1001001 0110000 1010101$
1111111	1001000
$C_{13} = 1001100 0000001 1111111$	$D_{13} = 0100101 1000010 1010110$
1111111	0100010
$C_{14} = 0110000 0000111 1111111$	$D_{14} = 0010110 0001010 1011001$
1111110	0001001
$C_{15} = 1000000 0011111 1111111$	$D_{15} = 1011000 0101010 1100100$
1111001	0100100
$C_{16} = 0000000 0111111 1111111$	$D_{16} = 0110000 1010101 1001000$
1110011	1001001

Setiap hasil putaran digabungkan kembali menjadi $C_i D_i$ dan diinput kedalam tabel Permutation Compression 2 (PC-2) dan terjadi kompresi data $C_i D_i$ 56-bit menjadi $C_i D_i$ 48-bit.

Berikut hasil outputnya :

$C_1 D_1 = 0000000 1111111 1111111 1100110 1100001 0101011 0010001 0010010$
 $K_1 = 111000 001011 111011 101110 110101 100100 011000 010010$

$$\begin{aligned} C_2D_2 &= 0000001\ 1111111\ 1111111\ 1001100\ 1000010\ 1010110\ 0100010\ 0100101 \\ K_2 &= 111000\ 001011\ 011011\ 110110\ 100000\ 011001\ 100110\ 000110 \end{aligned}$$

$$\begin{aligned} C_3D_3 &= 0000111\ 1111111\ 1111110\ 0110000\ 0001010\ 1011001\ 0001001\ 0010110 \\ K_3 &= 111101\ 001101\ 111001\ 110110\ 010001\ 000000\ 011010\ 110101 \end{aligned}$$

$$\begin{aligned} C_4D_4 &= 0011111\ 1111111\ 1111001\ 1000000\ 0101010\ 1100100\ 0100100\ 1011000 \\ K_4 &= 111001\ 101111\ 001101\ 110010\ 010110\ 110000\ 100011\ 001101 \end{aligned}$$

$$\begin{aligned} C_5D_5 &= 1111111\ 1111111\ 1100110\ 0000000\ 0101011\ 0010001\ 0010010\ 1100001 \\ K_5 &= 101011\ 101101\ 011101\ 110111\ 000000\ 101101\ 000110\ 011001 \end{aligned}$$

$$\begin{aligned} C_6D_6 &= 1111111\ 1111111\ 0011000\ 0000011\ 0101100\ 1000100\ 1001011\ 0000101 \\ K_6 &= 111011\ 110101\ 001101\ 011011\ 000000\ 110011\ 010100\ 100101 \end{aligned}$$

$$\begin{aligned} C_7D_7 &= 1111111\ 1111100\ 1100000\ 0001111\ 0110010\ 0010010\ 0101100\ 0010101 \\ K_7 &= 001011\ 111101\ 001111\ 111001\ 111010\ 100000\ 100110\ 100000 \end{aligned}$$

$$\begin{aligned} C_8D_8 &= 1111111\ 1110011\ 0000000\ 0111111\ 1001000\ 1001001\ 0110000\ 1010101 \\ K_8 &= 100111\ 110101\ 100111\ 011011\ 010000\ 000100\ 101100\ 011111 \end{aligned}$$

$$\begin{aligned} C_9D_9 &= 1111111\ 1100110\ 0000000\ 1111111\ 0010001\ 0010010\ 1100001\ 0101011 \\ K_9 &= 000111\ 110100\ 101111\ 011011\ 101001\ 001000\ 110101\ 000000 \end{aligned}$$

$$\begin{aligned} C_{10}D_{10} &= 1111111\ 0011000\ 0000011\ 1111111\ 1000100\ 1001011\ 0000101\ 0101100 \\ K_{10} &= 001111\ 110111\ 100110\ 011101\ 100010\ 001010\ 011001\ 010110 \end{aligned}$$

$$\begin{aligned} C_{11}D_{11} &= 1111100\ 1100000\ 0001111\ 1111111\ 0010010\ 0101100\ 0010101\ 0110010 \\ K_{11} &= 000111\ 110010\ 110111\ 001101\ 011111\ 011100\ 011010\ 000000 \end{aligned}$$

$$C_{12}D_{12} = 1110011\ 0000000\ 0111111\ 1111111\ 1001001\ 0110000\ 1010101\ 1001000$$

$$K_{12} = 010110\ 110110\ 110010\ 111101\ 000110\ 000100\ 010001\ 001011$$

$$C_{13}D_{13} = 1001100\ 0000001\ 1111111\ 1111111\ 0100101\ 1000010\ 1010110\ 0100010$$

$$K_{13} = 110111\ 011010\ 110110\ 101100\ 100011\ 101111\ 000000\ 000100$$

$$C_{14}D_{14} = 0110000\ 0000111\ 1111111\ 1111110\ 0010110\ 0001010\ 1011001\ 0001001$$

$$K_{14} = 110100\ 101010\ 111010\ 101111\ 101000\ 000110\ 011111\ 100000$$

$$C_{15}D_{15} = 1000000\ 0011111\ 1111111\ 1111001\ 1011000\ 0101010\ 1100100\ 0100100$$

$$K_{15} = 111110\ 011011\ 111000\ 100110\ 101110\ 001000\ 101000\ 000011$$

$$C_{16}D_{16} = 0000000\ 0111111\ 1111111\ 1110011\ 0110000\ 1010101\ 1001000\ 1001001$$

$$K_{16} = 111100\ 011011\ 111000\ 101110\ 001000\ 110000\ 000101\ 111100$$

6. Mengekspansi data

Data diekspansi R_{i-1} 32bit menjadi 48bit sebanyak 16 kali putaran dengan nilai perputaran $1 \leq i \leq 16$ menggunakan Tabel Ekspansi (E). Hasil $E(R_{i-1})$ kemudian di XOR dengan K_i dan menghasilkan vektor matriks A_i . Kemudian, vektor matriks A_i disubstitusikan ke 8 buah S-box (substitution box), Dimana blok ke-1 disubstitusikan ke S_1 , blok ke-2 disubstitusikan ke S_2 , hingga blok ke-8 disubstitusikan ke S_8 . Setiap blok dari A_i dipisahkan menjadi 2 blok, yaitu:

- Blok pertama dan terakhir (Sebagai pembanding baris)
- Blok kedua hingga kelima (Sebagai pembanding kolom)

Selanjutnya bandingkan dengan memeriksa diantara keduanya dan disesuaikan dengan tabel S-box yang akan menghasilkan output B_i 32bit. Setelah didapat vektor B_i , lalu permutasikan bit vektor B_i dengan tabel P-Box, lalu kelompokkan menjadi 4 blok Dimana tiap-tiap blok memiliki 32bit data. Hasil $P(B_i)$ di XOR kan dengan untuk mendapatkan nilai R_i . Sedangkan nilai L_i sendiri diperoleh dari nilai R_{i-1} dengan $1 \leq i \leq 16$. Lakukan langkah tersebut hingga mendapatkan R_{16} dan L_{16} .

Iterasi 1

$$E(R_0) = 010100\ 000011\ 110111\ 111000\ 000001\ 011011\ 110101\ 011101$$

$K_{16} = 111100\ 011011\ 111000\ 101110\ 001000\ 110000\ 000101\ 111100$
.....XOR
 $A_1 = 101000\ 011000\ 001111\ 010110\ 001001\ 101011\ 110000\ 100001$

 $B_1 = 0101\ 1001\ 1010\ 0101\ 0011\ 1000\ 0001\ 1011$
 $P(B_1) = 10111010\ 00001010\ 11011001\ 10000110$

 $P(B_1) = 11100011\ 10100111\ 10111110\ 11000010$
 $L_0 = 01100011\ 00100110\ 10111111\ 01100011$
.....XOR
 $R_1 = 10000000\ 10000001\ 00000001\ 10100001$

Iterasi 2

$E(R_1) = 110000\ 000001\ 010000\ 000010\ 100000\ 000011\ 110100\ 000011$
 $K_{15} = 111110\ 011011\ 111000\ 100110\ 101110\ 001000\ 101000\ 000011$
.....XOR
 $A_2 = 001110\ 011010\ 101000\ 100100\ 001110\ 001011\ 011100\ 000000$

 $B_2 = 0111\ 0011\ 1111\ 0100\ 0010\ 0100\ 0110\ 0110$
 $P(B_2) = 01000100\ 00010011\ 11010111\ 10101110$

 $P(B_2) = 01000100\ 00010011\ 11010111\ 10101110$
 $L_1 = 10100001\ 10111100\ 00001101\ 10101110$
.....XOR
 $R_2 = 01100101\ 10101111\ 01011010\ 10000001$

Iterasi 3

$E(R_2) = 101100\ 001011\ 110101\ 011110\ 101011\ 110101\ 010000\ 000010$
 $K_{14} = 110100\ 101010\ 111010\ 101111\ 101000\ 000110\ 011111\ 100000$
.....XOR
 $A_3 = 011000\ 100001\ 001111\ 110001\ 000011\ 110011\ 001111\ 100010$

 $B_3 = 0101\ 1101\ 1010\ 1001\ 1011\ 1110\ 1010\ 1011$

 $P(B_3) = 11101111\ 01011001\ 11001101\ 11111001$

 $P(B_3) = 10111001\ 00101010\ 11001101\ 11011111$
 $L_2 = 11011001\ 00101100\ 01100110\ 11100101$
.....XOR
 $R_3 = 01100000\ 00000110\ 10101011\ 00111010$

Iterasi 4

$E(R_3) = 001100\ 000000\ 000000\ 001101\ 010101\ 010110\ 100111\ 110100$
 $K_{13} = 110111\ 011010\ 110110\ 101100\ 100011\ 101111\ 000000\ 000100$
.....XOR
 $A_4 = 111011\ 011010\ 110110\ 100001\ 110110\ 111001\ 100111\ 110000$

B₄ = 0000 0000 1100 0011 0101 0110 1000 0000
P(B₄) = 10100000 01100101 00000001 00001001

P(B₄) = 10100000 01100101 00000001 00001001
L₃ = 01100101 10101111 01011010 10000001
.....XOR
R₄ = 11000101 11001010 01011011 10001000

Iterasi 5

E(R₄) = 011000 001011 111001 010100 001011 110111 110001 010001
K₁₂ = 010110 110110 110010 111101 000110 000100 010001 001011
.....XOR
A₅ = 001110 111101 001011 101001 001101 110011 100000 011010

B₅ = 1000 1110 0100 1010 1101 1110 0001 0000
P(B₅) = 01110011 11101101 00000000 01011000

P(B₅) = 01110011 11101101 00000000 01011000
L₄ = 01100000 00000110 10101011 00111010
.....XOR
R₅ = 00010011 11101011 10101011 01100010

Iterasi 6

E(R₅) = 000010 100111 111101 010111 110101 010110 101100 000100
K₁₁ = 000111 110010 110111 001101 011111 011100 011010 000000
.....XOR
A₆ = 000101 010101 001010 011010 101010 001010 110110 000100

B₆ = 0111 0001 0011 1100 1101 0010 1000 1000
P(B₆) = 00101101 00100100 11010010 01000111

P(B₆) = 00101101 00100100 11010010 01000111
L₅ = 11000101 11001010 01011011 10001000
.....XOR
R₆ = 11101000 11101110 10001001 11001111

Iterasi 7

E(R₆) = 111101 010001 011101 011101 010001 010011 111001 011111
K₁₀ = 001111 110111 100110 011101 100010 001010 011001 010110
.....XOR
A₇ = 110010 100110 111011 000000 110011 011001 100000 001001

B₇ = 1100 1011 0101 0111 1111 0000 0001 1010
P(B₇) = 11101111 11001111 11010000 10000000

$P(B_7) = 11101111 \ 11001111 \ 11010000 \ 10000000$
 $L_6 = 00010011 \ 11101011 \ 10101011 \ 01100010$
.....XOR
 $R_7 = 11111100 \ 00100100 \ 01111011 \ 11100010$

Iterasi 8

$E(R_7) = 011111 \ 111000 \ 000100 \ 001000 \ 001111 \ 110111 \ 111100 \ 000101$
 $K_9 = 000111 \ 110100 \ 101111 \ 011011 \ 101001 \ 001000 \ 110101 \ 000000$
.....XOR
 $A_8 = 011000 \ 001100 \ 101011 \ 010011 \ 100110 \ 111111 \ 001001 \ 000101$

 $B_8 = 0101 \ 0011 \ 1001 \ 0111 \ 1011 \ 1101 \ 0100 \ 1101$
 $P(B_8) = 11111101 \ 01010000 \ 11111001 \ 10101010$

 $P(B_8) = 11111101 \ 01010000 \ 11111001 \ 10101010$
 $L_7 = 11101000 \ 11101110 \ 10001001 \ 11001111$
.....XOR
 $R_8 = 00010101 \ 10111110 \ 01110000 \ 01100101$

Iterasi 9

$E(R_8) = 100010 \ 101011 \ 110111 \ 111100 \ 001110 \ 100000 \ 001100 \ 001010$
 $K_8 = 100111 \ 110101 \ 100111 \ 011011 \ 010000 \ 000100 \ 101100 \ 011111$
.....XOR
 $A_9 = 000101 \ 011110 \ 010000 \ 100111 \ 011110 \ 100100 \ 100000 \ 010101$

 $B_9 = 0111 \ 1010 \ 0001 \ 0110 \ 1001 \ 1111 \ 0001 \ 0110$
 $P(B_9) = 01110111 \ 01101010 \ 10110010 \ 00101010$

 $P(B_9) = 01110111 \ 01101010 \ 10110010 \ 00101010$
 $L_8 = 11111100 \ 00100100 \ 01111011 \ 11100010$
.....XOR
 $R_9 = 10001011 \ 01001110 \ 11001001 \ 11001000$

Iterasi 10

$E(R_9) = 010001 \ 010110 \ 101001 \ 011101 \ 011001 \ 010011 \ 111001 \ 010001$
 $K_7 = 001011 \ 111101 \ 001111 \ 111001 \ 111010 \ 100000 \ 100110 \ 100000$
.....XOR
 $A_{10} = 011010 \ 101011 \ 100110 \ 100100 \ 100011 \ 110011 \ 011111 \ 110001$

 $B_{10} = 1001 \ 1111 \ 1001 \ 1001 \ 1000 \ 1110 \ 0110 \ 1111$
 $P(B_{10}) = 11011101 \ 10111010 \ 01001101 \ 01111010$

 $P(B_{10}) = 11011101 \ 10111010 \ 01001101 \ 01111010$
 $L_9 = 00010101 \ 10111110 \ 01110000 \ 01100101$
.....XOR
 $R_{10} = 11001000 \ 00000100 \ 00111101 \ 00011111$

Iterasi 11

$E(R_{10}) = 111001\ 010000\ 000000\ 001000\ 000111\ 111010\ 100011\ 111111$
 $K_6 = 111011\ 110101\ 001101\ 011011\ 000000\ 110011\ 010100\ 100101$
.....XOR
 $A_{11} = 000010\ 100101\ 001101\ 010011\ 000111\ 001001\ 110111\ 011010$

 $B_{11} = 0100\ 1010\ 0110\ 0111\ 1100\ 0111\ 1111\ 0000$
 $P(B_{11}) = 11000011\ 01111101\ 10110100\ 00001101$

 $P(B_{11}) = 11000011\ 01111101\ 10110100\ 00001101$
 $L_{10} = 10001011\ 01001110\ 11001001\ 11001000$
.....XOR
 $R_{11} = 01001000\ 00110011\ 01111101\ 11000101$

Iterasi 12

$E(R_{11}) = 101001\ 010000\ 000110\ 100110\ 101111\ 111011\ 111000\ 001010$
 $K_5 = 101011\ 101101\ 011101\ 110111\ 000000\ 101101\ 000110\ 011001$
.....XOR
 $A_{12} = 000010\ 111101\ 011011\ 010001\ 101111\ 010110\ 111110\ 010011$

 $B_{12} = 0100\ 1110\ 1011\ 0100\ 1101\ 0100\ 0010\ 0101$
 $P(B_{12}) = 01100101\ 00001100\ 10011101\ 00111100$

 $P(B_{12}) = 01100101\ 00001100\ 10011101\ 00111100$
 $L_{11} = 11001000\ 00000100\ 00111101\ 00011111$
.....XOR
 $R_{12} = 10101101\ 00001000\ 10100000\ 00100011$

Iterasi 13

$E(R_{12}) = 110101\ 011010\ 100001\ 010001\ 010100\ 000000\ 000100\ 000111$
 $K_4 = 111001\ 101111\ 001101\ 110010\ 010110\ 110000\ 100011\ 001101$
.....XOR
 $A_{13} = 001100\ 110101\ 101100\ 100011\ 000010\ 110000\ 100111\ 001010$

 $B_{13} = 1011\ 0111\ 0011\ 1111\ 1100\ 0111\ 1000\ 1111$
 $P(B_{13}) = 11001101\ 11100110\ 01111010\ 01111111$

 $P(B_{13}) = 11001101\ 11100110\ 01111010\ 01111111$
 $L_{12} = 01001000\ 00110011\ 01111101\ 11000101$
.....XOR
 $R_{13} = 10000101\ 11010101\ 00000111\ 10111010$

Iterasi 14

$E(R_{13}) = 010000\ 001011\ 111010\ 101010\ 100000\ 001111\ 110111\ 110101$
 $K_3 = 111101\ 001101\ 111001\ 110110\ 010001\ 000000\ 011010\ 110101$

$$\begin{aligned}
& \dots\dots\dots \text{XOR} \\
A_{14} &= 101101\ 000110\ 000011\ 011100\ 110001\ 001111\ 101101\ 000000 \\
B_{14} &= 0001\ 1110\ 0111\ 0100\ 0110\ 0101\ 1010\ 1101 \\
P(B_{14}) &= 01001100\ 00001101\ 00111100\ 10111111 \\
P(B_{14}) &= 01001100\ 00001101\ 00111100\ 10111111 \\
L_{13} &= 10101101\ 00001000\ 10100000\ 00100011 \\
& \dots\dots\dots \text{XOR} \\
R_{14} &= 11100001\ 00000101\ 10011100\ 10011100
\end{aligned}$$

Iterasi 15

$$\begin{aligned}
E(R_{14}) &= 011100\ 000010\ 100000\ 001011\ 110011\ 111001\ 010011\ 111001 \\
K_2 &= 111000\ 001011\ 011011\ 110110\ 100000\ 011001\ 100110\ 000110 \\
& \dots\dots\dots \text{XOR} \\
A_{15} &= 100100\ 001001\ 111011\ 111101\ 010011\ 100000\ 110101\ 111111 \\
B_{15} &= 0100\ 1100\ 0011\ 0001\ 1010\ 0111\ 1000\ 0111 \\
P(B_{15}) &= 10000101\ 00101010\ 10101000\ 10111101 \\
P(B_{15}) &= 10000101\ 00101010\ 10101000\ 10111101 \\
L_{14} &= 10000101\ 11010101\ 00000111\ 10111010 \\
& \dots\dots\dots \text{XOR} \\
R_{15} &= 00000000\ 11111111\ 10101100\ 00000111
\end{aligned}$$

Iterasi 16

$$\begin{aligned}
E(R_{15}) &= 100000\ 000001\ 011111\ 111111\ 110101\ 011000\ 000000\ 001110 \\
K_1 &= 111000\ 001011\ 111011\ 101110\ 110101\ 100100\ 011000\ 010010 \\
& \dots\dots\dots \text{XOR} \\
A_{16} &= 011000\ 001010\ 100100\ 010001\ 000000\ 111100\ 011000\ 011100 \\
B_{16} &= 0001\ 1000\ 1101\ 1000\ 0100\ 1011\ 0011\ 1111 \\
P(B_{16}) &= 00011110\ 00100111\ 00001101\ 01100010 \\
P(B_{16}) &= 00011110\ 00100111\ 00001101\ 01100010 \\
L_{15} &= 11100001\ 00000101\ 10011100\ 10011100 \\
& \dots\dots\dots \text{XOR} \\
R_{16} &= 11111111\ 00100010\ 10010001\ 11101110
\end{aligned}$$

$$L_{16} = R_{15} = 00000000\ 11111111\ 10101100\ 00000111$$

7. Menggabungkan R_{16} dan L_{16} dan dipermutasi dengan tabel IP^{-1}

Hasil yang didapatkan setelah digabungkan dan dipermutasi menggunakan tabel Inverse Initial Permutation (IP^{-1}) :

$R_{16}L_{16} = 11111111\ 00100010\ 10010001\ 11101110\ 00000000\ 11111111\ 10101100\ 00000111$

Menghasilkan output:

Plainteks (dalam biner) = 01100110 01110011 01101011 01101001 01100100 01111001 01100001 01101101

atau

Plainteks (dalam hexa) = 66 73 6B 69 64 79 61 6D

Plainteks (dalam karakter) = fskidyam

F. Kombinasi Dekripsi Kunci dengan Playfair Cipher

Hal pertama yang harus dilakukan yaitu menentukan Cipherteks yang akan dijadikan kunci dan kunci untuk Playfair Cipher. Misalnya, diambil contoh :

Cipherteks Kunci : himastik

Kunci Pembangkit : fskidyam

Playfair Cipher adalah salah satu algoritma klasik. Untuk proses pembuatannya sendiri. Yaitu, Kunci kriptografinya adalah 25 buah huruf yang disusun di dalam bujursangkar 5x5 dengan menghilangkan huruf J dari abjad. Setiap elemen bujursangkar berisi huruf yang berbeda satu sama lain. Pesan yang akan didekripsi diatur terlebih dahulu sebagai berikut :

1. Ganti huruf J (bila ada) dengan huruf I.
2. Tulis pesan dalam pasangan huruf (*bigram*).
3. Jangan sampai ada pasangan huruf yang sama. Jika ada, sisipkan Z ditengahnya
4. Jika jumlah huruf ganjil, tambahkan huruf Z di sekitar.

f	a	i	s	h
l	b	c	d	e
g	k	m	n	o
p	q	r	t	u
v	w	x	y	z

Berdasarkan tabel 1, didapatkan Hasil Dekripsi Cipherteks Playfair Cipher yaitu “himastik”.

G. Dekripsi Pesan dengan Algoritma DES

1. Menentukan Cipherteks Pesan dan Kunci Pesan

Cipherteks : sg↓@■p±

Kunci Pesan : İW>.\$÷A,

2. Mengubah Cipherteks Kunci dan Kunci Pesan menjadi bilangan biner.

Kunci										Pesan									
Teks	Hexa	Biner								Teks	Hexa	Biner							
s	73	1	2	3	4	5	6	7	8	İ	D8	1	2	3	4	5	6	7	8
		0	1	1	1	0	0	1	1			1	1	0	1	1	0	0	0
g	67	1	2	3	4	5	6	7	8	w	57	1	2	3	4	5	6	7	8
		0	1	1	0	0	1	1	1			0	1	0	1	0	1	1	1
↓	19	1	2	3	4	5	6	7	8	>	3E	1	2	3	4	5	6	7	8
		0	0	0	1	1	0	0	1			0	0	1	1	1	1	1	0
@	40	1	2	3	4	5	6	7	8	.	2E	1	2	3	4	5	6	7	8
		0	1	0	0	0	0	0	0			0	0	1	0	1	1	1	0
■	DB	1	2	3	4	5	6	7	8	\$	24	1	2	3	4	5	6	7	8
		1	1	0	1	1	0	1	1			0	0	1	0	0	1	0	0
p	ES	1	2	3	4	5	6	7	8	÷	F6	1	2	3	4	5	6	7	8
		1	1	1	0	1	0	0	0			1	1	1	1	0	1	1	0
T	C2	1	2	3	4	5	6	7	8	A	41	1	2	3	4	5	6	7	8
		1	1	0	0	0	0	1	0			0	1	0	0	0	0	0	1
±	F1	1	2	3	4	5	6	7	8	,	F7	1	2	3	4	5	6	7	8
		1	1	1	1	0	0	0	1			1	1	1	1	0	1	1	1

3. Initial Permutation (IP) pada Cipherteks Pesan

Cipherteks (X)								IP(X)							
1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
0	1	1	1	0	0	1	1	1	1	1	0	1	1	1	1

9	10	11	12	13	14	15	16	→	9	10	11	12	13	14	15	16
0	1	1	0	0	1	1	1		1	0	0	1	0	1	0	1
17	18	19	20	21	22	23	24	→	17	18	19	20	21	22	23	24
0	0	0	1	1	0	0	1		0	0	0	0	0	0	1	0
25	26	27	28	29	30	31	32	→	25	26	27	28	29	30	31	32
0	1	0	0	0	0	0	0		1	0	0	1	0	1	1	1
33	34	35	36	37	38	39	40	→	33	34	35	36	37	38	39	40
1	1	0	1	1	0	1	1		1	1	1	1	0	0	0	0
41	42	43	44	45	46	47	48	→	41	42	43	44	45	46	47	48
1	1	1	0	1	0	0	0		1	0	1	0	0	0	1	1
49	50	51	52	53	54	55	56	→	49	50	51	52	53	54	55	56
1	1	0	0	0	0	1	0		0	0	1	1	0	1	0	0
57	58	59	60	61	62	63	64	→	57	58	59	60	61	62	63	64
1	1	1	1	0	0	0	1		0	1	0	1	0	0	1	1

Hasil IP(X) :

11111011 10010101 00000010 10010111 11110000 10100011 00110100 01010011

Selanjutnya bit pada IP(X) dipecah menjadi 2 :

L₀ : 11111011 10010101 00000010 10010111

R₀ : 11110000 10100011 00110100 01010011

4. Generate Kunci Menggunakan Tabel Permutasi Kompresi PC-1

Kompresi 64-bit menjadi 56-bit dengan membuang 1 bit (*parity bit*) tiap blok kunci.

Kunci								→	Output						
1	2	3	4	5	6	7	8		1	2	3	4	5	6	7
1	1	0	1	1	0	0	0	→	1	0	1	0	0	0	0
9	10	11	12	13	14	15	16	→	8	9	10	11	12	13	14
0	1	0	1	0	1	1	1		1	1	1	1	0	0	0
17	18	19	20	21	22	23	24	→	15	16	17	18	19	20	21
0	0	1	1	1	1	1	0		1	1	1	0	1	1	1
25	26	27	28	29	30	31	32	→	22	23	24	25	26	27	28
0	0	1	0	1	1	1	0		1	0	0	1	0	1	0
33	34	35	36	37	38	39	40	→	29	30	31	32	33	34	35
0	0	1	0	0	1	0	0		1	0	1	0	1	1	1
41	42	43	44	45	46	47	48	→	36	37	38	39	40	41	42
1	1	1	1	0	1	1	0		0	1	0	1	1	1	1
49	50	51	52	53	54	55	56	→	43	44	45	46	47	48	49
0	1	0	0	0	0	0	1		1	0	0	0	0	0	1
57	58	59	60	61	62	63	64	→	50	51	52	53	54	55	56
1	1	1	1	0	1	1	1		1	0	1	0	1	1	1

Hasil $C_0 D_0 =$

1010000 1111000 1110111 1001010 1010111 1101111 1100001 1010110

Selanjutnya bit pada $C_0 D_0$ dipecah menjadi 2.

C_0 : 1010000 1111000 1110111 1001010

D_0 : 1010111 0101111 1000001 1010111

5. *Left Shift Operation*

Melakukan pergeseran pada C_0 dan D_0 menggunakan tabel pergeseran bit 16 putaran.

$C_0 D_0 =$ 1010000 1111000 1110111 1001010 1010111 0101111 1000001 1010111

C_0	=	1010000 1111000 1110111 1001010	D_0	=	1010111 0101111 1000001 1010111
C_1	=	0100001 1110001 1101111 0010101	D_1	=	0101110 1011111 0000011 0101111
C_2	=	1000011 1100011 1011110 0101010	D_2	=	1011101 0111110 0000110 1011110
C_3	=	0001111 0001110 1111001 0101010	D_3	=	1110101 1111000 0011010 1111010
C_4	=	0111100 0111011 1100101 0101000	D_4	=	1010111 1100000 1101011 1101011
C_5	=	1110001 1101111 0010101 0100001	D_5	=	1011111 0000011 0101111 0101110
C_6	=	1000111 0111100 1010101 0000111	D_6	=	1111100 0001101 0111101 0111010
C_7	=	0011101 1110010 1010100 0011110	D_7	=	1110000 0110101 1110101 1101011
C_8	=	1110111 1001010 1010000 1111000	D_8	=	1110000 0110101 1110101 1101011
C_9	=	1101111 0010101 0100001 1110001	D_9	=	0000011 0101111 0101110 1011111
C_{10}	=	0111100 1010101 0000111 1000111	D_{10}	=	0001101 0111101 0111010 1111100
C_{11}	=	1110010 1010100 0011110 0011101	D_{11}	=	0110101 1110101 1101011 1110000
C_{12}	=	1001010 1010000 1111000 1110111	D_{12}	=	1010111 1010111 0101111 1000001
C_{13}	=	0101010 1000011 1100011 1011110	D_{13}	=	1011110 1011101 0111110 0000110
C_{14}	=	0101010 0001111 0001110 1111001	D_{14}	=	1111010 1110101 1111000 0011010
C_{15}	=	0101000 0111100 0111011 1100101	D_{15}	=	1101011 1010111 1100000 1101011
C_{16}	=	1010000 1111000 1110111 1001010	D_{16}	=	1010111 0101111 1000001 1010111

Setiap hasil putaran digabungkan kembali menjadi C_iD_i dan diinput kedalam tabel Permutation Compression 2 (PC-2) dan terjadi kompresi data C_iD_i 56-bit menjadi C_iD_i 48-bit.

Berikut hasil outputnya :

$$\begin{aligned} C_1D_1 &= 0100001\ 1110001\ 1101111\ 0010101\ 0101110\ 1011111\ 0000011\ 0101111 \\ K_1 &= 100100\ 011011\ 010011\ 110101\ 100001\ 111011\ 011111\ 010101 \end{aligned}$$

$$\begin{aligned} C_2D_2 &= 1000011\ 1100011\ 1011110\ 0101010\ 1011101\ 0111110\ 0000110\ 1011110 \\ K_2 &= 110010\ 001100\ 110001\ 011110\ 111111\ 010011\ 001001\ 001011 \end{aligned}$$

$$\begin{aligned} C_3D_3 &= 0001111\ 0001110\ 1111001\ 0101010\ 1110101\ 1111000\ 0011010\ 1111010 \\ K_3 &= 011001\ 001110\ 101100\ 111010\ 011101\ 101111\ 001001\ 101110 \end{aligned}$$

$$\begin{aligned} C_4D_4 &= 0111100\ 0111011\ 1100101\ 0101000\ 1010111\ 1100000\ 1101011\ 1101011 \\ K_4 &= 101001\ 101011\ 110100\ 100011\ 001101\ 001011\ 110111\ 101110 \end{aligned}$$

$$\begin{aligned} C_5D_5 &= 1110001\ 1101111\ 0010101\ 0100001\ 1011111\ 0000011\ 0101111\ 0101110 \\ K_5 &= 111010\ 110010\ 111001\ 010011\ 101011\ 001011\ 110011\ 110011 \end{aligned}$$

$$\begin{aligned} C_6D_6 &= 1000111\ 0111100\ 1010101\ 0000111\ 1111100\ 0001101\ 0111101\ 0111010 \\ K_6 &= 011011\ 011111\ 011010\ 011000\ 011011\ 111110\ 111001\ 110011 \end{aligned}$$

$$\begin{aligned} C_7D_7 &= 0011101\ 1110010\ 1010100\ 0011110\ 1110000\ 0110101\ 1110101\ 1101011 \\ K_7 &= 010101\ 101001\ 010111\ 011010\ 001111\ 111100\ 110101\ 011010 \end{aligned}$$

$$\begin{aligned} C_8D_8 &= 1110111\ 1001010\ 1010000\ 1111000\ 1000001\ 1010111\ 1010111\ 0101111 \\ K_8 &= 011111\ 101100\ 100001\ 010011\ 100011\ 011101\ 010101\ 010110 \end{aligned}$$

$$\begin{aligned} C_9D_9 &= 1101111\ 0010101\ 0100001\ 1110001\ 0000011\ 0101111\ 0101110\ 1011111 \\ K_9 &= 100111\ 010111\ 101100\ 110001\ 110111\ 010001\ 101111\ 111000 \end{aligned}$$

$$\begin{aligned} C_{10}D_{10} &= 0111100 1010101 0000111 1000111 0001101 0111101 0111010 1111100 \\ K_{10} &= 100001 110011 011111 001101 010100 011111 101001 111001 \end{aligned}$$

$$\begin{aligned} C_{11}D_{11} &= 1110010 1010100 0011110 0011101 0110101 1110101 1101011 1110000 \\ K_{11} &= 010110 110101 011011 000101 011100 111011 110000 111100 \end{aligned}$$

$$\begin{aligned} C_{12}D_{12} &= 1001010 1010000 1111000 1110111 1010111 1010111 0101111 1000001 \\ K_{12} &= 010110 011101 100111 101000 101010 010011 110110 111110 \end{aligned}$$

$$\begin{aligned} C_{13}D_{13} &= 0101010 1000011 1100011 1011110 1011110 1011101 0111110 0000110 \\ K_{13} &= 100100 001110 000111 101111 001011 010111 101010 110111 \end{aligned}$$

$$\begin{aligned} C_{14}D_{14} &= 0101010 0001111 0001110 1111001 1111010 1110101 1111000 0011010 \\ K_{14} &= 101100 010100 111100 000111 011101 110100 100011 110111 \end{aligned}$$

$$\begin{aligned} C_{15}D_{15} &= 0101000 0111100 0111011 1100101 1101011 1010111 1100000 1101011 \\ K_{15} &= 011000 010011 101110 100101 100001 111000 100111 011111 \end{aligned}$$

$$\begin{aligned} C_{16}D_{16} &= 1010000 1111000 1110111 1001010 1010111 0101111 1000001 1010111 \\ K_{16} &= 011010 101011 010001 101100 111101 010010 011110 011010 \end{aligned}$$

6. Mengekspansi data

Data diekspansi R_{i-1} 32bit menjadi 48bit sebanyak 16 kali putaran dengan nilai perputaran $1 \leq i \leq 16$ menggunakan Tabel Ekspansi (E). Hasil $E(R_{i-1})$ kemudian di XOR dengan K_i dan menghasilkan vektor matriks A_i . Kemudian, vektor matriks A_i disubstitusikan ke 8 buah S-box (substitution box), Dimana blok ke-1 disubstitusikan ke S_1 , blok ke-2 disubstitusikan ke S_2 , hingga blok ke-8 disubstitusikan ke S_8 . Setiap blok dari A_i dipisahkan menjadi 2 blok, yaitu:

- Blok pertama dan terakhir (Sebagai pembanding baris)
- Blok kedua hingga kelima (Sebagai pembanding kolom)

Selanjutnya bandingkan dengan memeriksa diantara keduanya dan disesuaikan dengan tabel S-box yang akan menghasilkan output B_i 32bit. Setelah didapat vektor B_i , lalu permutasikan bit vektor B_i dengan tabel P-Box, lalu kelompokkan menjadi 4 blok Dimana tiap-tiap blok memiliki 32bit data. Hasil $P(B_i)$ di XOR kan dengan untuk mendapatkan nilai R_i . Sedangkan nilai L_i sendiri diperoleh dari nilai R_{i-1} dengan $1 \leq i \leq 16$. Lakukan langkah tersebut hingga mendapatkan R_{16} dan L_{16} .

Iterasi 1

```

E(R0) = 110110 100001 010000 000110 100110 101000 001110 100111
K16   = 011010 101011 010001 101100 111101 010010 011110 011010
.....XOR
A1    = 101100 001010 000001 101010 011011 111010 010000 111101

B1    = 0010 1011 1101 1011 1001 1101 0011 0110
P(B1)  = 11110111 01001011 01100111 01101000

P(B1)  = 11110111 01001011 01100111 01101000
L0    = 00111011 11110101 01100010 10010111
.....XOR
R1    = 11001100 10111110 00000101 11111111

```

Iterasi 2

```

E(R1) = 111001 011001 010111 111100 000000 001011 111111 111111
K15   = 011000 010011 101110 100101 100001 111000 100111 011111
.....XOR
A2    = 100001 001010 111001 011001 100001 110011 011000 100000

B2    = 1111 1011 1011 0001 1011 1110 1110 0111
P(B2)  = 11110101 10111010 11001111 10101111

P(B2)  = 11110101 10111010 11001111 10101111
L1    = 10110000 10000011 00110100 01110011
.....XOR
R2    = 01000101 00111001 11111011 11011100

```

Iterasi 3

```

E(R2) = 001000 001010 100111 110011 111111 110111 111011 111000
K14   = 101100 010100 111100 000111 011101 110100 100011 110111
.....XOR
A3    = 100100 011110 011011 110100 100010 000011 011000 001111

B3    = 1110 1010 1011 0011 0010 1111 0101 0100

P(B3)  = 11010110 11111000 10100011 10101100

```

$P(B_3) = 11010110 \ 11111000 \ 10100011 \ 10101100$
 $L_2 = 11001100 \ 10111110 \ 00000101 \ 11111111$
.....XOR
 $R_3 = 00011010 \ 01000110 \ 10100110 \ 01010011$

Iterasi 4

$E(R_3) = 100011 \ 110100 \ 001000 \ 001101 \ 010100 \ 001100 \ 001010 \ 100110$
 $K_{13} = 100100 \ 001110 \ 000111 \ 101111 \ 001011 \ 010111 \ 101010 \ 110111$
.....XOR
 $A_4 = 000111 \ 111010 \ 001111 \ 100010 \ 011111 \ 011011 \ 100000 \ 010001$

$B_4 = 0100 \ 0011 \ 1010 \ 0110 \ 0110 \ 1011 \ 0001 \ 1100$
 $P(B_4) = 01011010 \ 01100100 \ 11110001 \ 10100100$

$P(B_4) = 01011010 \ 01100100 \ 11110001 \ 10100100$
 $L_3 = 01000101 \ 00111001 \ 11111011 \ 11011100$
.....XOR
 $R_4 = 00011111 \ 01011101 \ 00001010 \ 01111000$

Iterasi 5

$E(R_4) = 000011 \ 111110 \ 101011 \ 111010 \ 100001 \ 010100 \ 001111 \ 110000$
 $K_{12} = 010110 \ 011101 \ 100111 \ 101000 \ 101010 \ 010011 \ 110110 \ 111110$
.....XOR
 $A_5 = 010101 \ 100011 \ 001100 \ 010010 \ 001011 \ 000111 \ 111001 \ 001110$

$B_5 = 1100 \ 1000 \ 1111 \ 0010 \ 0111 \ 0010 \ 1110 \ 0001$
 $P(B_5) = 00100100 \ 11111101 \ 10001101 \ 10000101$

$P(B_5) = 00100100 \ 11111101 \ 10001101 \ 10000101$
 $L_4 = 00011010 \ 01000110 \ 10100110 \ 01010011$
.....XOR
 $R_5 = 00111110 \ 10111011 \ 00101011 \ 11010110$

Iterasi 6

$E(R_5) = 000111 \ 111101 \ 010111 \ 110110 \ 100101 \ 010111 \ 111010 \ 101100$
 $K_{11} = 010110 \ 110101 \ 011011 \ 000101 \ 011100 \ 111011 \ 110000 \ 111100$
.....XOR
 $A_6 = 010001 \ 001000 \ 001100 \ 110011 \ 111001 \ 101100 \ 001010 \ 010000$

$B_6 = 1010 \ 0110 \ 1111 \ 0100 \ 1010 \ 1100 \ 0000 \ 1010$
 $P(B_6) = 01011101 \ 10000011 \ 00010011 \ 10011100$

$P(B_6) = 01011101 \ 10000011 \ 00010011 \ 10011100$
 $L_5 = 00011111 \ 01011101 \ 00001010 \ 01111000$
.....XOR

$$R_6 = 01000010 \ 11011110 \ 00011001 \ 11100100$$

Iterasi 7

$$E(R_6) = 001000 \ 000101 \ 011011 \ 111100 \ 000011 \ 110011 \ 111100 \ 001000$$

$$K_{10} = 100001 \ 110011 \ 011111 \ 001101 \ 010100 \ 011111 \ 101001 \ 111001$$

$$\dots\dots\dots \text{XOR}$$

$$A_7 = 101001 \ 110110 \ 000100 \ 110001 \ 010111 \ 101100 \ 010101 \ 110001$$

$$B_7 = 0100 \ 0110 \ 1001 \ 1001 \ 1010 \ 1100 \ 0101 \ 1111$$

$$P(B_7) = 11011111 \ 00010010 \ 10001001 \ 11111000$$

$$P(B_7) = 11011111 \ 00010010 \ 10001001 \ 11111000$$

$$L_6 = 00111110 \ 10111011 \ 00101011 \ 11010110$$

$$\dots\dots\dots \text{XOR}$$

$$R_7 = 11100001 \ 10101001 \ 10100010 \ 00101110$$

Iterasi 8

$$E(R_7) = 011100 \ 000011 \ 110101 \ 010011 \ 110100 \ 000100 \ 000101 \ 011101$$

$$K_9 = 100111 \ 010111 \ 101100 \ 110001 \ 110111 \ 010001 \ 101111 \ 111000$$

$$\dots\dots\dots \text{XOR}$$

$$A_8 = 111011 \ 010100 \ 011001 \ 100010 \ 000011 \ 010101 \ 101010 \ 100101$$

$$B_8 = 0000 \ 0010 \ 1100 \ 0110 \ 1011 \ 1101 \ 0011 \ 1110$$

$$P(B_8) = 01111011 \ 01000011 \ 00110101 \ 10101000$$

$$P(B_8) = 01111011 \ 01000011 \ 00110101 \ 10101000$$

$$L_7 = 01000010 \ 11011110 \ 00011001 \ 11100100$$

$$\dots\dots\dots \text{XOR}$$

$$R_8 = 00111001 \ 10011101 \ 00101100 \ 01001100$$

Iterasi 9

$$E(R_8) = 000111 \ 110011 \ 110011 \ 111010 \ 100101 \ 011000 \ 001001 \ 011000$$

$$K_8 = 011111 \ 101100 \ 100001 \ 010011 \ 100011 \ 011101 \ 010101 \ 010110$$

$$\dots\dots\dots \text{XOR}$$

$$A_9 = 011000 \ 011111 \ 010010 \ 101001 \ 000110 \ 000101 \ 011100 \ 001110$$

$$B_9 = 0101 \ 0101 \ 1101 \ 1010 \ 0001 \ 0100 \ 0110 \ 0001$$

$$P(B_9) = 00100100 \ 01010001 \ 11001101 \ 01011010$$

$$P(B_9) = 00100100 \ 01010001 \ 11001101 \ 01011010$$

$$L_8 = 11100001 \ 10101001 \ 10100010 \ 00101110$$

$$\dots\dots\dots \text{XOR}$$

$$R_9 = 11000101 \ 11111000 \ 01101111 \ 01110100$$

Iterasi 10

$$E(R_9) = 011000 \ 001011 \ 111111 \ 110000 \ 001101 \ 011110 \ 101110 \ 101001$$

$K_7 = 010101\ 101001\ 010111\ 011010\ 001111\ 111100\ 110101\ 011010$
.....XOR
 $A_{10} = 001101\ 100010\ 101000\ 101010\ 000010\ 100010\ 011011\ 110011$

 $B_{10} = 1101\ 1110\ 1000\ 1011\ 1100\ 1110\ 1111\ 1100$
 $P(B_{10}) = 11011011\ 11111100\ 10000101\ 01111011$

 $P(B_{10}) = 11011011\ 11111100\ 10000101\ 01111011$
 $L_9 = 00111001\ 10011101\ 00101100\ 01001100$
.....XOR
 $R_{10} = 11100010\ 01100001\ 10101001\ 00110111$

Iterasi 11

$E(R_{10}) = 111100\ 000100\ 001100\ 000011\ 110101\ 010010\ 100110\ 101111$
 $K_6 = 011011\ 011111\ 011010\ 011000\ 011011\ 111110\ 111001\ 110011$
.....XOR
 $A_{11} = 100111\ 011011\ 010110\ 011011\ 101110\ 101100\ 011111\ 011100$

 $B_{11} = 0010\ 1001\ 0111\ 1010\ 1000\ 1100\ 0110\ 1100$
 $P(B_{11}) = 00011101\ 01011001\ 01000110\ 01101100$

 $P(B_{11}) = 00011101\ 01011001\ 01000110\ 01101100$
 $L_{10} = 11000101\ 11111000\ 01101111\ 01110100$
.....XOR
 $R_{11} = 11011000\ 10100001\ 00101001\ 00011000$

Iterasi 12

$E(R_{11}) = 011011\ 110001\ 010100\ 000010\ 100101\ 010010\ 100011\ 110001$
 $K_5 = 111010\ 110010\ 111001\ 010011\ 101011\ 001011\ 110011\ 110011$
.....XOR
 $A_{12} = 100001\ 000011\ 101101\ 010001\ 001110\ 011001\ 010000\ 000010$

 $B_{12} = 1111\ 1101\ 1000\ 0100\ 0110\ 0000\ 0011\ 0010$
 $P(B_{12}) = 00000010\ 10001110\ 11010111\ 10010010$

 $P(B_{12}) = 00000010\ 10001110\ 11010111\ 10010010$
 $L_{11} = 11100010\ 01100001\ 10101001\ 00110111$
.....XOR
 $R_{12} = 11100000\ 11101111\ 01111110\ 10100101$

Iterasi 13

$E(R_{12}) = 111100\ 000001\ 011101\ 011110\ 101111\ 111101\ 010100\ 001011$
 $K_4 = 101001\ 101011\ 110100\ 100011\ 001101\ 001011\ 110111\ 101110$
.....XOR
 $A_{13} = 010101\ 101010\ 101001\ 111101\ 100010\ 110110\ 100011\ 100101$

B₁₃ = 1100 0100 0110 0010 0010 1010 1011 1110
P(B₁₃) = 00011010 11100011 10000100 10110101

P(B₁₃) = 00011010 11100011 10000100 10110101
L₁₂ = 11011000 10100001 00101001 00011000
.....XOR
R₁₃ = 11000010 01000010 10101101 10101101

Iterasi 14

E(R₁₃) = 111000 000100 001000 000101 010101 011011 110101 011011
K₃ = 011001 001110 101100 111010 011101 101111 001001 101110
.....XOR
A₁₄ = 100001 001010 100100 111111 001000 110100 111100 110101

B₁₄ = 1111 1011 0100 1110 0111 0100 1001 1001
P(B₁₄) = 01101010 11001101 11011010 11001011

P(B₁₄) = 01101010 11001101 11011010 11001011
L₁₃ = 11100000 11101111 01111110 10100101
.....XOR
R₁₄ = 10001010 00100010 10100100 01101110

Iterasi 15

E(R₁₄) = 010001 010100 000100 000101 010100 001000 001101 011101
K₂ = 110010 001100 110001 011110 111111 010011 001001 001011
.....XOR
A₁₅ = 100011 011000 110101 011011 101011 011011 000100 010110

B₁₅ = 1100 1100 1110 1010 1110 1011 0010 1110
P(B₁₅) = 11000010 10111101 11000001 10111100

P(B₁₅) = 11000010 10111101 11000001 10111100
L₁₄ = 11000010 01000010 10101101 10101101
.....XOR
R₁₅ = 00000000 11111111 01101100 00010001

Iterasi 16

E(R₁₅) = 100000 000001 011111 111110 101101 011000 000010 100010
K₁ = 100100 011011 010011 110101 100001 111011 011111 010101
.....XOR
A₁₆ = 000100 011010 001100 001011 001100 100011 011101 110111

B₁₆ = 1000 1101 1001 1000 0010 0101 1001 0001
P(B₁₆) = 01110101 00110010 11100101 11011000

P(B₁₆) = 01110101 00110010 11100101 11011000

$$\begin{array}{rcl}
 L_{15} & = & 10001010 \ 00100010 \ 10100100 \ 01101110 \\
 & & \dots\dots\dots \text{XOR} \\
 R_{16} & = & 11111111 \ 00010000 \ 01000001 \ 10110110
 \end{array}$$

$$L_{16} = R_{15} = 00000000 \ 11111111 \ 01101100 \ 00010001$$

7. Menggabungkan R_{16} dan L_{16} dan dipermutasi dengan tabel IP^{-1}

Hasil yang didapatkan setelah digabungkan dan dipermutasi menggunakan tabel Inverse Initial Permutation (IP^{-1}) :

$$R_{16}L_{16} = 11111111 \ 00010000 \ 01000001 \ 10110110 \ 00000000 \ 11111111 \ 01101100 \ 00010001$$

Menghasilkan output:

$$\text{Plainteks (dalam biner)} = 01100110 \ 01100001 \ 01101001 \ 01101000 \ 01110011 \ 01101001 \ 01101100 \ 01100001$$

atau

$$\text{Plainteks (dalam hexa)} = 66 \ 61 \ 69 \ 68 \ 73 \ 69 \ 6C \ 61$$

$$\text{Plainteks (dalam karakter)} = \text{faishila}$$

H. Kombinasi Dekripsi Pesan dengan Columnar Transposition Cipher

Hal pertama yang harus dilakukan yaitu menentukan Cipherteks untuk Columnar Transposition Cipher.

$$\text{Kunci pesan} : \text{fW>.\$/A,}$$

$$\text{Cipherteks Pesan} : \text{faishali}$$

Selanjutnya Cipherteks pesan dan kunci pesan akan ditulis dalam bentuk matriks dengan panjang tetap. Baris 1 diisi dengan kunci pesan, baris 2 dan seterusnya akan diisi dengan Cipherteks pesan. Jika Cipherteks pesan lebih panjang dari kunci pesan, maka Cipherteks pesan tersebut dilanjutkan ditulis ke baris selanjutnya hingga pesan Cipherteks tersebut selesai ditulis. Untuk kolom dan baris matriks yang belum terisi akan diisi dengan null atau dibiarkan kosong atau ditempatkan oleh suatu karakter. Untuk Cipherteks pesan akan dibacakan dari pesan paling atas sampai paling bawah yang berurutan berdasarkan kunci pesan yang dimiliki sehingga didapatlah hasil pada Tabel berikut.

I	W	>	.	\$	÷	A	,
f	a	i	h	s	i	l	a

Berdasarkan tabel diatas, didapatkan Plainteks Columnar Transposition Cipher yaitu faishali.

IV. PROGRAM

Bahasa yang digunakan untuk membuat program ini yaitu bahasa C++ dengan bantuan software Code.Blocks.

- Pendeklarasian Library C++ dan Variabel Global

```
#include<bits/stdc++.h>
#include <iostream>
#include <conio.h>
#include<string>
#include<math.h>
#include<windows.h>
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
#define blockSize 4
#define SIZE 30
using namespace std;

string round_keys[16];
string pt;
```

- Fungsi Konversi Teks ke Biner

```
string convertBinaryToText(const string& binary)
{
    string text = "";
    for (size_t i = 0; i < binary.length(); i += 8)
    {
        string byte = binary.substr(i, 8);
        char c = static_cast<char>(bitset<8>(byte).to_ulong());
        text += c;
    }
    return text;
}
```

- Fungsi Konversi Desimal ke Biner

```
string convertDecimalToBinary(int decimal)
{
    string binary;
    while (decimal != 0)
    {
        binary = (decimal % 2 == 0 ? "0" : "1") + binary;
        decimal = decimal / 2;
    }
    while (binary.length() < 4)
    {
        binary = "0" + binary;
    }
    return binary;
}
```

- Fungsi Konversi Biner ke Desimal

```
int convertBinaryToDecimal(const string& binary)
{
    int decimal = 0;
    int counter = 0;
    int size = binary.length();
    for (int i = size - 1; i >= 0; i--)
    {
        if (binary[i] == '1')
        {
            decimal += pow(2, counter);
        }
        counter++;
    }
    return decimal;
}
```

- Fungsi String Left Once

```
shift_left_once(string key_chunk)
{
    string shifted = "";
    for (int i = 1; i < 28; i++)
    {
        shifted += key_chunk[i];
    }
    shifted += key_chunk[0];
    return shifted;
}
```

- Fungsi String Left Twice

```
string shift_left_twice(string key_chunk)
{
    string shifted = "";
    for (int i = 0; i < 2; i++)
    {
        for (int j = 1; j < 28; j++)
        {
            shifted += key_chunk[j];
        }
        shifted += key_chunk[0];
        key_chunk = shifted;
        shifted = "";
    }
    return key_chunk;
}
```

- Fungsi XOR

```
string Xor(string a, string b)
{
    string result = "";
    int size = b.size();
    for (int i = 0; i < size; i++)
    {
        if (a[i] != b[i])
        {
            result += "1";
        }
        else
        {
            result += "0";
        }
    }
    return result;
}
```

- Fungsi Pembangkitan Kunci Algoritma DES

```
void generate_keys(string key)
{
    // The PC1 table
    int pc1[56] =
    {
        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,
7,62,54,46,38,30,22,
14,6,61,53,45,37,29,
21,13,5,28,20,12,4
};
// The PC2 table
int pc2[48] =
{
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
};
// 1. Compressing the key using the PC1 table
string perm_key = "";
for(int i = 0; i < 56; i++)
{
    perm_key += key[pc1[i]-1];
}
// 2. Dividing the key into two equal halves
string left= perm_key.substr(0, 28);
string right= perm_key.substr(28, 28);
for(int i=0; i<16; i++)
{
    // 3.1. For rounds 1, 2, 9, 16 the key_chunks
    // are shifted by one.
    if(i == 0 || i == 1 || i==8 || i==15 )
    {
        left= shift_left_once(left);
        right= shift_left_once(right);
    }
    // 3.2. For other rounds, the key_chunks
    // are shifted by two
    else
    {
        left= shift_left_twice(left);
        right= shift_left_twice(right);
    }
    // Combining the two chunks
    string combined_key = left + right;

```



```

string ciphertext = "";
// The inverse of the initial permuttaion is applied
for(int i = 0; i < 64; i++)
{
    ciphertext+= combined_text[inverse_permutation[i]-1];
}
//And we finally get the cipher text
return ciphertext;
}

```

- Fungsi Input

```

void getInputText(string& input, const string& prompt)
{
    cout << prompt << " : ";
    getline(cin, input);
}

```

- Fungsi Mengatur Panjang Teks

```

string adjustBitLength(const string& input)
{
    string adjustedInput = input;

    // If length is less than 64, add 0s to the end
    while (adjustedInput.length() < 64)
    {
        adjustedInput += "0";
    }

    // If length is more than 64, trim to 64
    if (adjustedInput.length() > 64)
    {
        adjustedInput = adjustedInput.substr(0, 64);
    }

    return adjustedInput;
}

```

- Fungsi Power

```

int power(int a, int b, int P)
{
    if (b == 1)
        return a;

    else

```

```
return (((int)pow(a, b)) % P);
}
```

- Fungsi Mengubah Huruf Kapital Menjadi Huruf Kecil

```
void toLowerCase(char plain[], int ps)
{
    int i;
    for (i = 0; i < ps; i++) {
        if (plain[i] > 64 && plain[i] < 91)
            plain[i] += 32;
    }
}
```

- Fungsi Menghapus Spasi

```
int removeSpaces(char* plain, int ps)
{
    int i, count = 0;
    for (i = 0; i < ps; i++)
        if (plain[i] != ' ')
            plain[count++] = plain[i];
    plain[count] = '\0';
    return count;
}
```

- Fungsi Persiapan Dekripsi Playfair Cipher

```
void generateKeyTable(char key[], int ks, char keyT[5][5])
{
    int i, j, k, flag = 0;

    // a 26 character hashmap
    // to store count of the alphabet
    int dicty[26] = { 0 };
    for (i = 0; i < ks; i++) {
        if (key[i] != 'j')
            dicty[key[i] - 97] = 2;
    }

    dicty['j' - 97] = 1;

    i = 0;
    j = 0;

    for (k = 0; k < ks; k++) {
        if (dicty[key[k] - 97] == 2) {
```

```

dicty[key[k] - 97] -= 1;
    keyT[i][j] = key[k];
    j++;
    if (j == 5) {
        i++;
        j = 0;
    }
}

for (k = 0; k < 26; k++) {
    if (dicty[k] == 0) {
        keyT[i][j] = (char)(k + 97);
        j++;
        if (j == 5) {
            i++;
            j = 0;
        }
    }
}

void decrypt(char str[], char keyT[5][5], int ps)
{
    int i, a[4];
    for (i = 0; i < ps; i += 2) {
        search(keyT, str[i], str[i + 1], a);
        if (a[0] == a[2]) {
            str[i] = keyT[a[0]][mod5(a[1] - 1)];
            str[i + 1] = keyT[a[0]][mod5(a[3] - 1)];
        }
        else if (a[1] == a[3]) {
            str[i] = keyT[mod5(a[0] - 1)][a[1]];
            str[i + 1] = keyT[mod5(a[2] - 1)][a[1]];
        }
        else {
            str[i] = keyT[a[0]][a[3]];
            str[i + 1] = keyT[a[2]][a[1]];
        }
    }
}

```

- Fungsi Dekripsi Playfair Cipher

```
void decryptByPlayfairCipher(char str[], char key[])
{
    char ps, ks, keyT[5][5];

    // Key
    ks = strlen(key);
    ks = removeSpaces(key, ks);
    toLowerCase(key, ks);

    // ciphertext
    ps = strlen(str);
    toLowerCase(str, ps);
    ps = removeSpaces(str, ps);

    generateKeyTable(key, ks, keyT);

    decrypt(str, keyT, ps);
}
```

- Fungsi Persiapan Enkripsi Playfair Cipher

```
int prepare(char str[], int ptrs)
{
    if (ptrs % 2 != 0) {
        str[ptrs++] = 'z';
        str[ptrs] = '\0';
    }
    return ptrs;
}

void encrypt(char str[], char keyT[5][5], int ps)
{
    int i, a[4];

    for (i = 0; i < ps; i += 2) {

        search(keyT, str[i], str[i + 1], a);

        if (a[0] == a[2]) {
            str[i] = keyT[a[0]][mod5(a[1] + 1)];
            str[i + 1] = keyT[a[0]][mod5(a[3] + 1)];
        }
        else if (a[1] == a[3]) {
            str[i] = keyT[mod5(a[0] + 1)][a[1]];
            str[i + 1] = keyT[mod5(a[2] + 1)][a[1]];
        }
    }
}
```

```

    }
    else {
        str[i] = keyT[a[0]][a[3]];
        str[i + 1] = keyT[a[2]][a[1]];
    }
}
}

```

- Fungsi Enkripsi Playfair Cipher

```

void encrypt(char str[], char keyT[5][5], int ps)
{
    int i, a[4];

    for (i = 0; i < ps; i += 2) {

        search(keyT, str[i], str[i + 1], a);

        if (a[0] == a[2]) {
            str[i] = keyT[a[0]][mod5(a[1] + 1)];
            str[i + 1] = keyT[a[0]][mod5(a[3] + 1)];
        }
        else if (a[1] == a[3]) {
            str[i] = keyT[mod5(a[0] + 1)][a[1]];
            str[i + 1] = keyT[mod5(a[2] + 1)][a[1]];
        }
        else {
            str[i] = keyT[a[0]][a[3]];
            str[i + 1] = keyT[a[2]][a[1]];
        }
    }
}

```

```

void encryptByPlayfairCipher(char str[], char key[])
{
    char ps, ks, keyT[5][5];

    // Key
    ks = strlen(key);
    ks = removeSpaces(key, ks);
    toLowerCase(key, ks);

    // Plaintext
    ps = strlen(str);
    toLowerCase(str, ps);
}

```

```

ps = prepare(str, ps);

    generateKeyTable(key, ks, keyT);

    encrypt(str, keyT, ps);
}

```

- Fungsi Mengubah Huruf Kecil Menjadi Kapital

```

string to_upper(string &in)
{
    for (int i = 0; i < in.length(); i++)
        if (islower(in[i]))
            if ('a' <= in[i] <= 'z')
                in[i] = in[i] - 'a' + 'A';
    return in;
}

```

- Fungsi Enkripsi dan Dekripsi Algoritma DES

```

string DES()
{
    // The initial permutation table
    int initial_permutation[64] =
    {
        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
    };
    // The expansion table
    int expansion_table[48] =
    {
        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
    };
    // The substitution boxes. The should contain values
    // from 0 to 15 in any order.
    int substitution_boxes[8][4][16]=

```

```

{
    {
        14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7,
        0,15,7,4,14,2,13,1,10,6,12,11,9,5,3,8,
        4,1,14,8,13,6,2,11,15,12,9,7,3,10,5,0,
        15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13
    },
    {
        15,1,8,14,6,11,3,4,9,7,2,13,12,0,5,10,
        3,13,4,7,15,2,8,14,12,0,1,10,6,9,11,5,
        0,14,7,11,10,4,13,1,5,8,12,6,9,3,2,15,
        13,8,10,1,3,15,4,2,11,6,7,12,0,5,14,9
    },
    {
        10,0,9,14,6,3,15,5,1,13,12,7,11,4,2,8,
        13,7,0,9,3,4,6,10,2,8,5,14,12,11,15,1,
        13,6,4,9,8,15,3,0,11,1,2,12,5,10,14,7,
        1,10,13,0,6,9,8,7,4,15,14,3,11,5,2,12
    },
    {
        7,13,14,3,0,6,9,10,1,2,8,5,11,12,4,15,
        13,8,11,5,6,15,0,3,4,7,2,12,1,10,14,9,
        10,6,9,0,12,11,7,13,15,1,3,14,5,2,8,4,
        3,15,0,6,10,1,13,8,9,4,5,11,12,7,2,14
    },
    {
        2,12,4,1,7,10,11,6,8,5,3,15,13,0,14,9,
        14,11,2,12,4,7,13,1,5,0,15,10,3,9,8,6,
        4,2,1,11,10,13,7,8,15,9,12,5,6,3,0,14,
        11,8,12,7,1,14,2,13,6,15,0,9,10,4,5,3
    },
    {
        12,1,10,15,9,2,6,8,0,13,3,4,14,7,5,11,
        10,15,4,2,7,12,9,5,6,1,13,14,0,11,3,8,
        9,14,15,5,2,8,12,3,7,0,4,10,1,13,11,6,
        4,3,2,12,9,5,15,10,11,14,1,7,6,0,8,13
    },
    {
        4,11,2,14,15,0,8,13,3,12,9,7,5,10,6,1,
        13,0,11,7,4,9,1,10,14,3,5,12,2,15,8,6,
        1,4,11,13,12,3,7,14,10,15,6,8,0,5,9,2,
        6,11,13,8,1,4,10,7,9,5,0,15,14,2,3,12
    },
    {
        13,2,8,4,6,15,11,1,10,9,3,14,5,0,12,7,
        1,15,13,8,10,3,7,4,12,5,6,11,0,14,9,2,
        7,11,4,1,9,12,14,2,0,6,10,13,15,3,5,8,

```

```

    2,1,14,7,4,10,8,13,15,12,9,0,3,5,6,11
    }
};
// The permutation table
int permutation_tab[32] =
{
    16,7,20,21,29,12,28,17,
    1,15,23,26,5,18,31,10,
    2,8,24,14,32,27,3,9,
    19,13,30,6,22,11,4,25
};
// The inverse permutation table
int inverse_permutation[64]=
{
    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
};
//1. Applying the initial permutation
string perm = "";
for(int i = 0; i < 64; i++)
{
    perm += pt[initial_permutation[i]-1];
}
// 2. Dividing the result into two equal halves
string left = perm.substr(0, 32);
string right = perm.substr(32, 32);
// The plain text is encrypted 16 times
for(int i=0; i<16; i++)
{
    string right_expanded = "";
    // 3.1. The right half of the plain text is expanded
    for(int i = 0; i < 48; i++)
    {
        right_expanded += right[expansion_table[i]-1];
    }; // 3.3. The result is xored with a key
    string xored = Xor(round_keys[i], right_expanded);
    string res = "";
    // 3.4. The result is divided into 8 equal parts and passed
    // through 8 substitution boxes. After passing through a
    // substitution box, each box is reduces from 6 to 4 bits.
    for(int i=0;i<8; i++)

```



```

{
    // Finding row and column indices to lookup the
    // substitution box
    string row1= xored.substr(i*6,1) + xored.substr(i*6 + 5,1);
    int row = convertBinaryToDecimal(row1);
    string col1 = xored.substr(i*6 + 1,1) + xored.substr(i*6 + 2,1) + xored.substr(i*6 +
3,1) + xored.substr(i*6 + 4,1);;
    int col = convertBinaryToDecimal(col1);
    int val = substitution_boxes[i][row][col];
    res += convertDecimalToBinary(val);
}
// 3.5. Another permutation is applied
string perm2="";
for(int i = 0; i < 32; i++)
{
    perm2 += res[permutation_tab[i]-1];
}
// 3.6. The result is xored with the left half
xored = Xor(perm2, left);
// 3.7. The left and the right parts of the plain text are swapped
left = xored;
if(i < 15)
{
    string temp = right;
    right = xored;
    left = temp;
}
}
// 4. The halves of the plain text are applied
string combined_text = left + right;
string ciphertext="";
// The inverse of the initial permuttaion is applied
for(int i = 0; i < 64; i++)
{
    ciphertext+= combined_text[inverse_permutation[i]-1];
}
//And we finally get the cipher text
return ciphertext;
}

```

- Fungsi Enkripsi Pesan dengan Columnar Transposition Cipher dan DES

```

void encryptMessage() {
    int row,col,j;
    string key;
    cout<<" Masukkan Kunci Pesan : ";

```

```

cin>>key;
map<int,int> keyMap;
string msg;
cout<<" Masukkan Pesan :";
cin>>msg;
string cipher = "";

// Add the permutation order into map
for(int i=0; i < key.length(); i++)
{
    keyMap[key[i]] = i;
}
/* calculate column of the matrix*/
col = key.length();

/* calculate Maximum row of the matrix*/
row = msg.length()/col;

if (msg.length() % col)
    row += 1;

char matrix[row][col];

for (int i=0,k=0; i < row; i++)
{
    for (int j=0; j<col; )
    {
        if(msg[k] == '\0')
        {
            /* Adding the padding character '_' */
            matrix[i][j] = '_';
            j++;
        }

        if( isalpha(msg[k]) || msg[k]==' ')
        {
            /* Adding only space and alphabet into matrix*/
            matrix[i][j] = msg[k];
            j++;
        }
        k++;
    }
}

for (map<int,int>::iterator ii = keyMap.begin(); ii!=keyMap.end(); ++ii)
{
    j=ii->second;

```

```

        // getting cipher text from matrix column wise using permuted key
        for (int i=0; i<row; i++)
        {
            if( isalpha(matrix[i][j]) || matrix[i][j]==' ' || matrix[i][j]=='_')
                cipher += matrix[i][j];
        }
    }
    // Calling encryption function
    cout << "Pesan yang Sudah di Columnar : " << cipher << endl<<endl;

    string binaryKey = convertTextToBinary(key); // Convert key and plain text to binary
    string binaryPlainText = convertTextToBinary(cipher); // Convert key and plain text to binary
    binaryKey = adjustBitLength(binaryKey); // Adjust bit length
    binaryPlainText = adjustBitLength(binaryPlainText); // Adjust bit length

    generate_keys(binaryKey);

    cout << "Plain Text: " << cipher << endl;

    pt = binaryPlainText;

    string cipherText = DES();

    cout << "Hasil Akhir Enkripsi Pesan : " << convertBinaryToText(cipherText) << endl;

}

```

- Fungsi Dekripsi Pesan dengan DES dan Columnar Transposition Cipher

```

void decryptMessage()
{
    string key, plainText, keyd, plainTextd;
    getInputText(key, "Masukkan Kunci Untuk DES ");

    // Get plain text input from the user
    getInputText(plainTextd, "Masukkan Plain Teks ");

    // Convert key and plain text to binary
    string binaryKeyd = convertTextToBinary(key);
    string binaryPlainTextd = convertTextToBinary(plainTextd);

    // Adjust bit length
    binaryKeyd = adjustBitLength(binaryKeyd);
    binaryPlainTextd = adjustBitLength(binaryPlainTextd);
    generate_keys(binaryKeyd);
}

```

```

cout << "Plain Text: " << plainTextd << endl;

pt = binaryPlainTextd;

int i = 15;
int j = 0;
while (i > j)
{
    string temp = round_keys[i];
    round_keys[i] = round_keys[j];
    round_keys[j] = temp;
    i--;
    j++;
}

string decryptedText = DES();

// Convert binary to text
string decryptedTextInText = convertBinaryToText(decryptedText);

// Output in text format
cout << "Hasil Dekripsi DES : " << decryptedTextInText << endl<< endl;

map<int,int> keyMap;
string plaintext = "";

// Add the permutation order into map
for(int i=0; i < key.length(); i++)
{
    keyMap[key[i]] = i;
}

/* calculate row and column for cipher Matrix */
int col = key.length();

int row = decryptedTextInText.length()/col;
char cipherMat[row][col];

/* add character into matrix column wise */
for (int j=0,k=0; j<col; j++)
    for (int i=0; i<row; i++)
        cipherMat[i][j] = decryptedTextInText[k++];

/* update the order of key for decryption */
int index = 0;
for( map<int,int>::iterator ii=keyMap.begin(); ii!=keyMap.end(); ++ii)
    ii->second = index++;

```

```

/* Arrange the matrix column wise according
to permutation order by adding into new matrix */
char decCipher[row][col];
map<int,int>::iterator ii=keyMap.begin();
int k = 0;
for (int l=0;j; key[l]!='\0'; k++)
{
    j = keyMap[key[l++]];
    for (int i=0; i<row; i++)
    {
        decCipher[i][k]=cipherMat[i][j];
    }
}

/* getting Message using matrix */
for (int i=0; i<row; i++)
{
    for(int j=0; j<col; j++)
    {
        if(decCipher[i][j] != '_')
            plaintext += decCipher[i][j];
    }
}

// Calling encryption function
cout << "Hasil Dekripsi Akhir : " << plaintext << endl;
}

```

- Fungsi Utama

```

int main()
{
    int jawab;
    menu:
        cout<<"          *****
"<<endl;
        cout<<"          PROGRAM ALGORITMA DES DAN PLAYFAIR DAN          "<<endl;
        cout<<"          PEMBANGKITAN   KUNCI   MENGGUNAKAN   COLUMNAR
TRANSPPOSITION DAN DES   "<<endl;
        cout<<"          *****
"<<endl;
        cout<<"\t1.) Enkripsi Kunci\n";
        cout<<"\t2.) Enkripsi Pesan\n";
        cout<<"\t3.) Dekripsi Kunci\n";
        cout<<"\t4.) Dekripsi Pesan\n";
        cout<<"\t5.) Selesai\n";

```

```

cout<<"\tSilahkan memilih menu sesuai nomor... "<<endl;
cout<<"\tPilihan Anda: ";
    cin>>jawab;
cin.ignore(); // mengabaikan input spasi yang terjadi ketika input jawab tadi
system("cls");
    switch(jawab)
    {
case 1:
    {
        string kunci,strply,keyd;
        char str[SIZE], key[SIZE];
        cout<<"
***** " <<endl;
        cout<<"                ENKRIPSI KUNCI                " <<endl;
        cout<<"
***** " <<endl;
        cout<<"Masukkan Kunci Untuk Playfair : ";
        cin>>kunci;
        const int length = kunci.length();
        char* strkunci = new char[length + 1];
        strcpy(strkunci, kunci.c_str());
        strcpy(key, strkunci);

        cout<<"Masukkan Plainteks Kunci yang Ingin Dibangkitkan : ";
        cin>>strply;
        const int panjang = strply.length();
        char* strteks = new char[panjang + 1];
        strcpy(strteks, strply.c_str());
        strcpy(str, strteks);

        encryptByPlayfairCipher(str, key);

        cout<<"Hasil Pembangkitan Kunci Menggunakan Playfair: "<< str <<endl<<endl;

    cout << "Masukkan Kunci Untuk DES : ";
    cin >> keyd;

    string binaryKey = convertTextToBinary(keyd);
    string binaryPlainText = convertTextToBinary(str);

    binaryKey = adjustBitLength(binaryKey);
    binaryPlainText = adjustBitLength(binaryPlainText);

    generate_keys(binaryKey);
    pt = binaryPlainText;
    string cipherText = DES();

```

```

cout << "Hasil Pembangkitan Kunci Akhir : " << convertBinaryToText(cipherText) << endl;

    getch();
    system("cls");
    goto menu;
}
case 2:
{
    cout<<"
***** " <<endl;

    cout<<"                ENKRIPSI PESAN                " <<endl;
    cout<<"
***** " <<endl;

    encryptMessage();
    getch();
    system("cls");
    goto menu;
}
case 3:
{
    string kuncid,strd,keywordd;
    char str[SIZE], key[SIZE];
    cout<<"
***** " <<endl;

    cout<<"                DEKRIPSI KUNCI                " <<endl;
    cout<<"
***** " <<endl;

    cout<<"Masukkan Kunci Untuk Dekripsi DES : ";
    cin>>kuncid;
    cout<<"Masukkan Plainteks Kunci : ";
    cin>>strd;

    string binaryKeyd = convertTextToBinary(kuncid);
    string binaryPlainTextd = convertTextToBinary(strd);

    binaryKeyd = adjustBitLength(binaryKeyd);
    binaryPlainTextd = adjustBitLength(binaryPlainTextd);

    generate_keys(binaryKeyd);

    pt = binaryPlainTextd;

    int i = 15;
    int j = 0;
    while (i > j)
    {
        string temp = round_keys[i];

```

```

        round_keys[i] = round_keys[j];
        round_keys[j] = temp;
        i--;
        j++;
    }

    string decryptedText = DES();

    // Convert binary to text
    string decryptedTextInText = convertBinaryToText(decryptedText);

    cout << "Hasil Dekripsi DES : " << decryptedTextInText << endl<<endl;

    string kuncip;
    cout<< "Masukkan Kunci Dekripsi Playfair : ";
    cin>>kuncip;

    const int length = decryptedTextInText.length();
    char* strp = new char[length + 1]; // declaring character array (+1 for null terminator)
    strcpy(strp, decryptedTextInText.c_str()); // copying the contents of the string to char array

    const int lengthk = kuncip.length();
    char* keyp = new char[lengthk + 1]; // declaring character array (+1 for null terminator)
    strcpy(keyp, kuncip.c_str());

    decryptByPlayfairCipher(strp, keyp);
    cout<<"Hasil Dekripsi Akhir : "<< strp <<endl;

    getch();
    system("cls");
    goto menu;
}
case 4:
{
    cout<<"
***** " <<endl;
        cout<<"                DEKRIPSI PESAN                " <<endl;
        cout<<"
***** " <<endl;
    decryptMessage();
    getch();
    system("cls");
    goto menu;
}
case 5:
{

```



```

    getch();
    system("cls");
    goto menu;
}
default:
cout<<"\tPilihan Anda salah, silahkan coba kembali";
system("cls");
goto menu;
}
}

```

V. HASIL PROGRAM

Tampilan pertama pada program yang sudah dibuat yaitu pilihan yang bisa dipilih oleh pengguna. Terdapat 5 pilihan yaitu Enkripsi Kunci, Enkripsi Pesan, Dekripsi Kunci, Dekripsi Pesan, dan Keluar. Adapun tampilannya adalah sebagai berikut :

```

"C:\Users\faish\Downloads\Projek Akhir.exe"
*****
                PROGRAM ALGORITMA DES DAN PLAYFAIR DAN
                PEMBANGKITAN KUNCI MENGGUNAKAN COLUMNAR TRANSPOSITION DAN DES
                *****
                1.) Enkripsi Kunci
                2.) Enkripsi Pesan
                3.) Dekripsi Kunci
                4.) Dekripsi Pesan
                5.) Selesai
                Silahkan memilih menu sesuai nomor...
                Pilihan Anda: _

```

1. Enkripsi Kunci

Enkripsi Kunci akan berjalan apabila pengguna memasukkan angka 1 pada tampilan awal. Hal pertama yang dilakukan pada enkripsi kunci yaitu menginput kunci dan plainteks untuk Playfair Cipher. Lalu program akan mengenkripsi kunci dan plainteks menjadi cipherteks Playfair Cipher.

```
"C:\Users\faish\Downloads\Projek Akhir.exe"
*****
                        ENKRIPSI KUNCI
*****
Masukkan Kunci Untuk Playfair : faishal
Masukkan Plainteks Kunci yang Ingin Dibangkitkan : himastik
Hasil Pembangkitan Kunci Menggunakan Playfair: fskidyam
```

Selanjutnya pengguna akan diminta untuk menginput kunci untuk algoritma DES, maka program akan memproses kunci DES dan hasil cipherteks Playfair Cipher menjadi cipher teks yang baru.

```
"C:\Users\faish\Downloads\Projek Akhir.exe"
*****
                        ENKRIPSI KUNCI
*****
Masukkan Kunci Untuk Playfair : faishal
Masukkan Plainteks Kunci yang Ingin Dibangkitkan : himastik
Hasil Pembangkitan Kunci Menggunakan Playfair: fskidyam

Masukkan Kunci Untuk DES : sempurna
Hasil Pembangkitan Kunci Akhir : ÌW>.$÷A.
```

2. Enkripsi Pesan

Enkripsi Pesan akan dilakukan apabila pengguna memilih angka 2 pada tampilan awal. Pada enkripsi kunci, pengguna akan diminta memasukkan kunci untuk Algoritma Columnar Transposition Cipher dan DES serta plainteks untuk Algoritma Columnar Transposition Cipher. Setelah pengguna menginput, program akan memproses enkripsi Algoritma Columnar Transposition Cipher menggunakan kunci dan plainteks yang sudah diinput dan menampilkan Cipherteks Algoritma Columnar Transposition Cipher.

```
"C:\Users\faish\Downloads\Projek Akhir.exe"
*****
                        ENKRIPSI PESAN
*****
Masukkan Kunci Pesan : ÌW>.$÷A.
Masukkan Pesan :faishali
Pesan yang Sudah di Columnar : faihsila
```

Setelah menampilkan cipherteks, program akan melanjutkan enkripsi pesan Algoritma DES menggunakan kunci yang sudah diinput sebelumnya dan cipherteks dari Algoritma Columnar Transposition Cipher. Hasil Cipherteks DES lalu ditampilkan ke layar.

```
"C:\Users\faish\Downloads\Projek Akhir.exe"
*****
                        ENKRIPSI PESAN
*****
Masukkan Kunci Pesan : ĪW>.$÷A,
Masukkan Pesan : faishali
Pesan yang Sudah di Columnar : faihsila

Plain Text: faihsila
Hasil Akhir Enkripsi Pesan : sg↓@P_T±
```

3. Dekripsi Kunci

Program Dekripsi Kunci akan dilakukan apabila pengguna menginput angka 3 pada halaman awal. Pengguna akan diminta untuk menginput kunci dan cipherteks untuk dekripsi Algoritma DES.

```
"C:\Users\faish\Downloads\Projek Akhir.exe"
*****
                        DEKRIPSI KUNCI
*****
Masukkan Kunci Untuk Dekripsi DES : sempurna
Masukkan Plainteks Kunci : ĪW>.$÷A,
Hasil Dekripsi DES : fskidyam
```

Selanjutnya pengguna akan diminta untuk memasukkan kunci untuk Playfair Cipher, lalu program akan memproses dekripsi Playfair Cipher menggunakan kunci yang sudah diinput dan cipherteks hasil dekripsi DES.

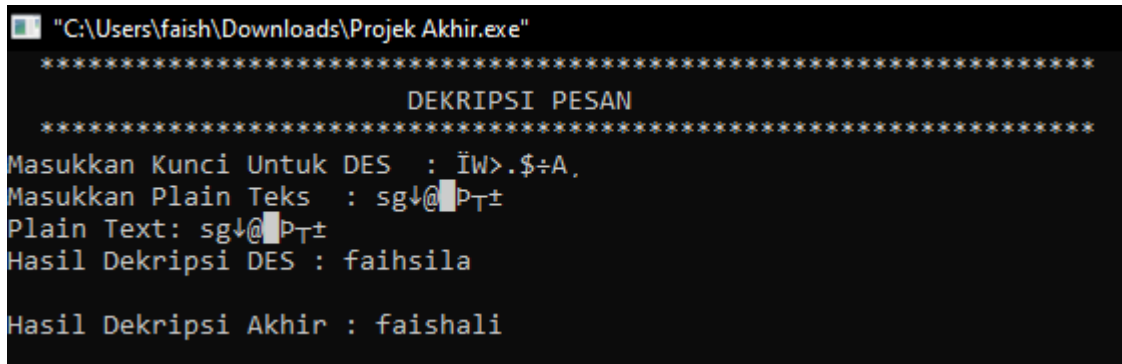
```
"C:\Users\faish\Downloads\Projek Akhir.exe"
*****
                        DEKRIPSI KUNCI
*****
Masukkan Kunci Untuk Dekripsi DES : sempurna
Masukkan Plainteks Kunci : ĪW>.$÷A,
Hasil Dekripsi DES : fskidyam

Masukkan Kunci Dekripsi Playfair : faishal
Hasil Dekripsi Akhir : himastik
```

4. Dekripsi Pesan

Program Dekripsi Pesan akan dilakukan apabila pengguna memasukkan angka 4 di tampilan awal. Pada dekripsi pesan, pengguna akan diminta untuk memasukkan kunci dan plainteks untuk dekripsi DES dan Columnar Transposition. Setelah dimasukkan maka program akan memproses enkripsi DES dan cipherteks DES diproses kembali dengan dekripsi

Columnar Transposition Cipher. Lalu, hasil dekripsi DES dan Columnar Transposition ditampilkan ke layar.



```
"C:\Users\faish\Downloads\Projek Akhir.exe"
*****
                        DEKRIPSI PESAN
*****
Masukkan Kunci Untuk DES : IW>.$÷A.
Masukkan Plain Teks   : sg↓@P_T±
Plain Text: sg↓@P_T±
Hasil Dekripsi DES : faihsila

Hasil Dekripsi Akhir : faishali
```

VI. KESIMPULAN

Program yang dibentuk untuk mengamankan pesan pada penelitian ini yaitu dengan mengkombinasikan algoritma Playfair Cipher, Columnar Transposition Cipher, dan DES sehingga pesan yang akan diamankan terjaga kerahasiaannya. Pada proses enkripsi kunci digunakan dengan Algoritma Playfair Cipher dan Algoritma DES. Kemudian hasil enkripsi tersebut akan dijadikan kunci pada proses enkripsi pesan dengan menggunakan Algoritma Columnar Transposition Cipher dan Algoritma DES. Pada program juga terdapat dekripsi pesan dan dekripsi kunci, di mana jika pengguna memasukkan hasil enkripsi yang telah didapat pada proses sebelumnya maka akan dihasilkan plaintext awal. Dari penelitian ini dapat disimpulkan bahwa ketiga algoritma yang digunakan dapat meningkatkan keamanan pesan yang dienkrapsikan agar tidak mudah ditebak oleh pihak manapun.

DAFTAR PUSTAKA

- Amin, M. M. (2017). Implementasi Kriptografi Klasik Pada Komunikasi Berbasis Teks. *Pseudocode*, 3(2), 129–136. <https://doi.org/10.33369/pseudocode.3.2.129-136>
- Basri. (2016). Kriptografi Simetris dan Asimetris dalam Perspektif Keamanan Data dan Kompleksitas Komputasi. *Jurnal Ilmiah Ilmu Komputer*, 2(2). <http://ejournal.fikom-unasman.ac.id/index.php/jikom/article/view/82%0Ahttp://ejournal.fikom-unasman.ac.id/index.php/jikom/article/download/82/55>
- Dar, J. A. (2014). Enhancing the Data Security of Simple Columnar Transposition Cipher By Caesar Cipher and Rail Fence Cipher Technique . *International Journal of Computer Science & Engineering Technology (IJCSET)*, 5(11), 1054–1061.
- Nugroho, A. R., & Pramusinto, W. (2018). Implementasi Kriptografi dengan Algoritma Caesar Ccipher, AES 192 dan DES untuk Aplikasi Pesan Instan Berbasis Android. *Skanika*, 1(1), 14–19. <https://jom.fti.budiluhur.ac.id/index.php/SKANIKA/article/download/153/606>
- Saputro, T. H., Hidayati, N. H., & Ujianto, E. I. H. (2020). Survei Tentang Algoritma Kriptografi Asimetris. *Jurnal Informatika Polinema*, 6(2), 67–72. <https://doi.org/10.33795/jip.v6i2.345>
- Siregar, S., Fadlina, F., & Nasution, S. (2020). *Enhancing Data Security of Columnar Transposition Cipher by Fibonacci Codes Algorithm*. <https://doi.org/10.4108/eai.11-12-2019.2290839>
- Susanti, D. (2020). Analisis Modifikasi Metode Playfair Cipher Dalam Pengamanan Data Teks. *Indonesian Journal of Data and Science*, 1(1), 11–18. <https://doi.org/10.33096/ijodas.v1i1.4>
- Tendean, S. (n.d.). Perancangan Aplikasi Kriptografi E-Mail Dengan Metode Columnar Transposition Cipher. *Jurnal InTekSis*, 4(2), 76.