PENERAPAN KOMBINASI ALGORITMA RFC, RSA, RC4 DAN OTP DALAM MENGAMANKAN PESAN

D

I

S

U

S

U

N

Oleh :

Nama	: Fitri Salamah
NIM	: 08011182025003
Fakultas / Jurusan	: MIPA / Matematika
Mata Kuliah	: Kriptografi
Dosen Pengajar	: Dr. Anita Desiani, S.Si., M.Kom

JURUSAN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS SRIWIJAYA

2022

BAB I
PENDAHULUAN

Keamanan dan privasi data adalah salah satu aspek penting dari sistem pengiriman informasi. Perkembangan teknologi media komunikasi yang semakin canggih, banyak pihak yang mencari celah untuk mencuri informasi dan digunakan untuk hal-hal yang tidak baik (Sutoyo et al., 2019). Informasi tidak akan berguna lagi jika di tengah pengiriman informasi dicegat atau disalahgunakan oleh orang yang tidak berwenang (Irawan, 2017). Dalam hal ini, sangat berkaitan dengan pentingnya informasi dikirim dan diterima oleh orang-orang yang berkepentingan . Salah satu cara untuk mencegah pengungkapan informasi kepada siapa pun yang tidak memiliki akses adalah dengan membuat kerahasiaan data. Kriptografi dapat digunakan untuk menjaga keamanan data (Amrulloh and Ujianto, 2019).

Kriptografi adalah ilmu pengetahuan untuk melindungi pengiriman data dengan mengubahnya menjadi kode tertentu dan hanya tersedia untuk pengguna yang memiliki kunci untuk mengubah kode tersebut (Arya et al., 2022). Dalam kriptografi dikenal dua konsep utama yaitu enkripsi dan dekripsi. Enkripsi adalah proses di mana informasi atau data dikirim dan diubah menjadi bentuk yang hampir tidak dapat dikenali sebagai informasi asli oleh algoritma tertentu. Dekripsi adalah kebalikan dari enkripsi, yang mengubah kembali informasi atau data yang disamarkan menjadi informasi asli (Prabowo and Pramusinto, 2018). Adapun contoh dari algoritma klasik, yaitu Caesar Cipher, Vigenere Cipher, dan Hill Cipher. Contoh algoritma kunci simetris, yaitu DES (*Data Encryption Standar*), 3DES, RC4 (*Riverst Code 4*), IDEA, Blowfish, Twofish, dan AES (*Advanced Encryption Standard*). Dan contoh dari algoritma kunci asimetris, yaitu RSA (*Riverst Shamir Adleman*) dan ECC (*Elliptic Curve Cryptography*) (Irawan, 2017). Pada pembahasan ini, akan digunakan 4 algoritma, yaitu algoritma RFC (*Rail Fence Cipher*), algoritma RSA (*Riverst Shamir Adleman*), algoritma RC4 (*Riverst Code 4*) dan algoritma OTP (*One Time Pad*).

Algoritma RFC (*Rail Fence Cipher*) merupakan salah satu variasi implementasi cipher transposisi. Pada Rail Fence Cipher, plainteks dituliskan secara vertikal kebawah sepanjang n-rails, dan menulis lagi kekolom baru ketika telah mencapai karakter ke-n (Ratna, 2018). Algoritma RSA ditemukan pada tahun 1987 oleh Ron Rivest, Adi Shamir, dan Leonard Adleman. Algoritma ini merupakan algoritma kriptografi kunci-publik yang paling popular dan sering digunakan (Ginting et al., 2015). Keamanan algoritma RSA terletak pada sulitnya memfaktorkan bilangan besar menjadi faktor-faktor prima yang dilakukan untuk memperoleh kunci privat (Jayusman et al., 2017). Algoritma OTP pertama kali diperkenalkan oleh Gilbert Vernam dalam perang dunia pertama. OTP merupakan salah satu variasi dari metode penyandian substitusi dengan cara memberikan syarat-syarat khusus terhadap kunci yang

digunakan. Kunci yang digunakan terbuat dari karakter/huruf yang acak (kunci acak atau pad), dan pengacakannya tidak menggunakan rumus tertentu (Eko Tinikar et al., 2014).

Beberapa penelitian sebelumnya telah dilakukan dilakukan penerapan algoritma RSA pada platform android (Jayusman et al., 2017) sehingga dapat melindungi pesan yang dikirim dan tidak bisa dibaca oleh pihak ketiga sehingga Kerahasiaan email dapat terjaga dengan adanya dan telah diterapkan juga algoritma RC4 pada pengamanan citra digital (Taronisokhi and Ndruru, 2017), yaitu dengan cara menambahkan sebuah nilai inisialisasi awal pada setiap operasi XOR antar biner dan pixel citra. Pada penelitian sebelumnya telah dilakukan dilakukan penerapan algoritma OTP pada pengamanan data file (Eko Tinikar et al., 2014), penyandian OTP ini sangat kuat dan tidak dapat dipecahkan sehingga data file terjaga kerahasiannya dengan baik. Pada pembahasan kali ini akan dilakukan kombinasi antara algoritma RFC, RSA algoritma RC4 dan algoritma OTP untuk keamanan pesan. Hal ini dilakukan dengan mengenkripsikan plaintext pesan dengan langkah pertama mengenkripsikan plaintext kunci dengan menggunakan algoritma Rail Fence Cipher dan RSA Selanjutnya, akan melakukan pergeseran plaintext ke kanan dan hasil tersebut akan menjadi kunci pada algoritma Rail Fence Cipher. Kemudian, plaintext dan hasil plaintext terbaru di enkripsikan menggunakan algoritma Rail Fence Cipher. Setelah, mendapatkan hasil enkripsi maka hasil tersebut akan digunakan pada algoritma RSA dan menggunakan kunci yang telah dimasukkan. Kemudian, hasil enkripsi tersebut akan digunakan sebagai kunci untuk proses enkripsi pesan. Pada enkripsi pesan, akan dimasukkan plaintext pesan dan kunci yang sudah didapatkan dari enkripsi kunci. Selanjutnya, plaintext dan kunci akan dienkripsi menggunakan algoritma RC4. Selanjutnya, hasil enkripsi dari algoritma RC4 akan digunakan sebagai plaintext pada algoritma OTP dan kunci yang digunkan masih tetap kunci sebelumnya. Kemudian, hasil enkripsi algoritma OTP dienkripsi kembali menggunakan Algoritma OTP. Dan akan didapatkan ciphertext yang telah melewati beberapa proses enkripsi yaitu dengan algoritma RSA, algoritma RC4 dan algoritma OTP. Hal ini dilakukan untuk meningkatkan keamanan pesan yang dienkripsikan agar tidak mudah ditebak oleh pihak manapun.

BAB II
METODE

Langkah-langkah yang digunakan dalam mengkombinasikan algoritma RFC, RSA, RC4 dan OTP adalah sebagai berikut:

1. Plaintext

Plaintext merupakan pesan asli yang belum disandikan atau informasi yang ingin dikirimkan atau dijaga keamanannya. Bentuk plaintext beraneka ragam berdasarkan media yang

digunakan antara lain yaitu teks, gambar, suara, video maupun IP protokol (Maulana and Simanjorang, 2021).

2. RSA

Sandi RSA merupakan algoritma kriptografi kunci publik (asimetris). Ditemukan pertama kali pada tahun 1977 oleh Ron Rivest, Adi Shamir, dan Len Adleman (Zulfikar et al., 2019). Langkah-langkah yang dilakukan dalam algoritma RSA adalah membangkitkan kunci dengan cara memilih 2 bilangan prima yang disimbolkan dengan nilai p dan q, lalu menghitung nilai modulusnya dengan hitung menggunakan fungsi eulernya dengan $\phi(n) = (p-1)(q-1)$, selanjutnya menentukan nilai e acak sebagai kunci public dengan syarat memenuhi $gcd(\phi(n) = 1, \quad 1 < \mathrm{e} < \phi(n)$ dan hitung kunci privat d dengan $d \times e = 1(mod\ \phi(n))$. Setelah itu dilakukan enkripsi dengan $C = M^e\ mod\ n$ dan deskripsi dengan $M = C^d\ mod\ n$ (Arief and Saputra, 2016).

3. RFC

Algoritma RFC merupakan salah satu variasi implementasi cipher transposisi. Pada Rail Fence Cipher, plainteks dituliskan secara vertikal kebawah sepanjang n-rails, dan menulis lagi kekolom baru ketika telah mencapai karakter ke-n (Ratna, 2018). Cipherteks yang dihasilkan adalah urutan karakter yang dibaca secara horizontal. Sebagai contoh, kita mempunyai n=3 dan sebuah pesan. Pada Rail Fence Cipher, plainteks dituliskan secara vertikal kebawah sepanjang n-rails, dan menulis lagi kekolom baru ketika telah mencapai karakter ke-n. Cipherteks yang dihasilkan adalah urutan karakter yang dibaca secara horizontal (Febriani et al., 2020).

4. Algoritma RC4

RC4 diciptakan oleh Ron Rivest di laboratorium RSA dengan RC adalah singkatan dari *Ron's Code*. RC4 membangkitkan *keystream* yang kemudian di XOR-kan dengan plaintext pada waktu enkripsi (atau di XOR kan dengan bit ciphertext pada waktu deskripsi). RC4 bukan seperti cipher aliran biasa yang memproses data yang terdapat dalam bit. RC4 memproses data yang ada dalam waktu hanya ukuran byte (1 byte = 8 bit). RC4 menggunakan 2 kotaksubstitusi atau (S-box) dengan array 256 byte dan yang berisi permutasi daripada bilangan 0 sampai dengan 255 dan S-box kedua berisi permutasi fungsi daripada kunci sepanjang variabel (Purba et al., 2020).

5. *One Time Pad* (OTP)

Algoritma *One Time Pad* (OTP) adalah stream cipher yang melakukan enkripsi dan dekripsi satu karakter setiap kali. Algoritma ini merupakan perbaikan dari Vernam cipher untuk menghasilkan keamanan yang sempurna. Cipher ini termasuk ke dalam kelompok algoritma kriptografi simetri. *One Time Pad* (*pad* = kertas bloknot) berisi barisan karakter-karakter kunci yang dibangkitkan secara acak. Aslinya, satu buah *One Time Pad* adalah sebuah pita (tape)

yang berisi barisan karakter-karakter kunci. Satu pad hanya digunakan sekali (*one time*) saja untuk mengenkripsi pesan, setelah itu pad yang telah digunakan dihancurkan supaya tidak dipakai kembali untuk mengenkripsi pesan yang lain (Harahap and Khairina, 2018). Penggunaan *One Time Pad* berikut yang digunakan penulis menggunakan tabel ASCII, berikut merupakan table ASCII.

Algoritma enkripsi menggunakan XOR adalah dengan meng-XOR-kan *plaintext* (P) dengan kunci (K) menghasilkan *ciphertext* (C) terlihata pada Persamaan (1).

$$C_i = P \oplus K \quad \text{... (1)}$$

Algoritma dekripsi menggunakan XOR adalah dengan meng-XOR-kan *plaintext* (P) dengan kunci (K) menghasilkan *ciphertext* (C):

$$P_i = C \oplus K \quad \text{... (2)}$$

Keterangan rumus :

Ci = Cipherteks(*Ciphertext*),

Pi = Plainteks (*Plaintext*),

Ki = kunci (*Key*)

BAB III

PEMBAHASAN

3.1 Perhitungan Manual

a. Enkripsi

- **Enkripsi Kunci**

Pada pembahasan akan dilakukan perhitungan manual untuk mengenkripsi kunci dengan algoritma RSA dan algoritma Rail Fence Cipher lalu dienkripsikan sehingga akan didapatkan *ciphertext* yang telah melalui proses enkripsi. Plainteks yang akan digunakan yaitu :

Tahap 1 (Dekripsi Kunci menggunkan Rail Fence Cipher dan RSA)

Plaintext : s@1mhfit12

Key RSA : $(p, q) = (11,13)$

Shift k : 3

Plaintext Rail Fence Cipher : t12s@lmhfi

Plaintext diubah ke dalam bentuk desimal berdasarkan tabel ASCII sehingga didapatkan hasil pada Tabel 1 sebagai berikut.

Tabel 1. Tabel Hasil Plaintext String ke ASCII

t	1	2	s	@	1	m	h	f	i
116	49	50	115	64	49	109	104	102	105

Dari Tabel 1, akan dicari nilai rata-rata dari hasil plaintext ke ascii sehingga didapatkan perhitungan berikut ini.

$$rata - rata = \frac{116 + 49 + 50 + 115 + 64 + 49 + 109 + 104 + 102 + 105}{10} = 86$$

Selanjutnya, setelah mendapatkan nilai rata-rata, diambil angka terbesar dari rata-rata tersebut. Sehingga, diambil angka 8 untuk digunakan sebagai kunci di algoritma Rail Fence Cipher.

Tabel 2. Proses Enkripsi Algoritma Rail Fence Cipher

t									
	1								
		2							
			S						
				@					
					1				i
						m		f	
							h		

Dari Tabel 2 didapatkan hasil plaintext Rail Fence Cipher, yaitu t12s@lmhfi. Selanjutnya, akan digunakan algoritma RSA untuk mengenkripsi kunci dengan langkah-langkah sebagi berikut.

a. Pilih p=11 dan q=13
b. Hitung $n = p \times q = 11 \times 13 = 143$
c. Hitung fungsi euler

$$\phi(n) = (p-1)(q-1)$$
$$\phi(n) = (11-1)(13-1)$$
$$\phi(n) = (10)(12)$$
$$\phi(n) = 120$$

d. Pilih e sedemikian sehingga gcd (e,120)
   Diambil e=19, karena

$$120 = 19.6 + 6$$
$$19 = 6.3 + 1$$
$$6 = 1.6 + 0$$

   Sehingga, $\gcd(e, 120)$

e. Hitung kunci d dengan $d = \frac{1+(k \times 120)}{19}$

$$d = \frac{1 + (k \times 120)}{19}$$

   Hitunglah nilai d dengan periksa k=1,2,3,... dan akan diperoleh nilai bulat d pada saat k=3

$$d = \frac{1 + (3 \times 120)}{19}$$

$$d = \frac{361}{19}$$

$$d = 19$$

Sehingga didapatkan kunci public adalah (e=19,n=143) dan kunci privatnya adalah (d=19, n=143). Setelah mendapatkan kunci publik dan kunci privat, akan dilakukan enkripsi dengan proses sebagai berikut:

Plaintext diubah ke dalam bentuk desimal berdasarkan tabel ASCII sehingga menjadi:

t	1	2	s	@	1	i	m	f	h
116	49	50	115	64	49	105	109	102	104

Sehingga "s@1zh" dalam bentuk desimal adalah 115 644 912 210 004, kemudian dipecah menjadi lima blok yang berukuran 2 digit sedemikian sehingga.

$$m_1 = 116 \qquad m_6 = 49$$
$$m_2 = 49 \qquad m_7 = 105$$
$$m_3 = 50 \qquad m_8 = 109$$
$$m_4 = 115 \qquad m_9 = 102$$
$$m_5 = 64 \qquad m_{10} = 104$$

Lalu, proses enkripsi dilakukan dengan ci = mi e mod n pada setiap blok. Selanjutnya, akan dilakukan enkripsi dengan algoritma RSA dengan cara sebagai berikut:

$$c_1 = 116^{19}\ mod\ 143 = 90$$
$$c_2 = 49^{19}\ mod\ 143 = 75$$
$$c_3 = 50^{19}\ mod\ 143 = 2$$
$$c_4 = 115^{19}\ mod\ 143 = 119$$
$$c_5 = 64^{19}\ mod\ 143 = 38$$
$$c_6 = 49^{19}\ mod\ 143 = 75$$
$$c_7 = 105^{19}\ mod\ 143 = 79$$
$$c_8 = 109^{19}\ mod\ 143 = 21$$
$$c_9 = 102^{19}\ mod\ 143 = 15$$
$$c_{10} = 104^{19}\ mod\ 143 = 69$$

Sehingga ciphertext dari "s@1mhfit12" dengan algoritma RSA adalah atau 90 75 2 119 38 75 79 21 15 69. Selanjutnya, cipherteks akan di enkripsikan lagi dengan algoritma RC4 dengan tahap-tahap sebagai berikut.

- **Enkripsi Pesan**

Pada pembahasan ini akan dilakukan perhitungan manual untuk mengenkripsi pesan dengan algoritma RC4 lalu dienkripsikan lagi dengan algoritma OTP sebanyak dua kali sehingga akan didapatkan *ciphertext* yang telah melalui proses enkripsi. *Plaintext* pesan yang akan digunakan yaitu: F~T#0. Selanjutnya, *plaintext* kunci diambil dari proses enkripsi kunci pada algoritma RSA dengan tahap sebagai berikut.

Plaintext : [83 126 84 35 48]

Kunci : [90 75 2 119 38 75 79 21 15 69]

- **Langkah 1**

Inisialisasi larik S:$S_0 = 0, S_1 = 1, \dots, S_{255} = 255$. S-box yang diketahui memiliki panjang 5 *byte*, dengan S[0] = 0, S[1] = 1, S[2] = 2, S[3] = 3, S[4] = 4, S[5] = 5 sehingga array S menjadi: 0 1 2 3 4 5

Tabel 3. S-Box 5 Byte

INDEX	0	1	2	3	4
S	0	1	2	3	4

- **Langkah 2:**

Memasukkan nilai kunci, misalkan 5 *byte* kunci array: K = 9 0 7 5 2

Tabel 4. Kunci Array 5 Byte

INDEX	0	1	2	3	4
K	9	0	7	5	2

- **Langkah 3:**

Melakukan permutasi nilai-nilai dalam larik S. Lakukan KSA (*Key State Array*) dengan rumus:

$j \leftarrow 0$

$for\ i \leftarrow 0\ to\ 255\ do$

$\quad j \leftarrow (j + [i] + [i])\ mod\ 256$

$\quad swap(S[i], S[j])$

$endfor$

Dari rumus tersebut kita akan melakukan perhitungan menggunakan 5 *byte*, dimana iterasi yang akan dijalani adalah sebanyak 5 kali perulangan.

❖ Iterasi 1

i = 0

j = (0 + S[0] + K[0]) mod 5

j = (0 + 0 + 9) mod 5

j = 9 mod 5

j = 4

$Swap$ ([0], [4])

Tabel 5. Hasil Array Iterasi 1

INDEX	0	1	2	3	4
S	4	1	2	3	0

❖ Iterasi 2

i = 1

j = (1 + S[1] + K[1]) mod 5

j = (1 + 1+ 0) mod 5

j = 2 mod 5

j = 2

$Swap$ ([1], [2])

Tabel 6. Hasil Array S Iterasi 2

INDEX	0	1	2	3	4
S	4	2	1	3	0

❖ Iterasi 3

i = 2

j = (2 + S[2] + K[2]) mod 5

j = (2 + 1 + 7) mod 5

j = 10 mod 5

j = 0

$Swap$ ([2], [0])

Tabel 7. Hasil Array S Iterasi 3

INDEX	0	1	2	3	4
S	1	2	4	3	0

❖ Iterasi 4

i = 3

j = (3 + S[3] + K[3]) mod 5

j = (3 + 3 + 5) mod 5

j = 11 mod 5

j = 1

$Swap$ ([3], [1])

Tabel 8. Hasil Array S Iterasi 4

INDEX	0	1	2	3	4
S	1	3	4	2	0

❖ Iterasi 5

i = 4

j = (4 + S[4] + K[4]) mod 5

j = (4 + 0 + 2) mod 5

j = 6 mod 5

j = 1

$Swap$ ([4], [1])

Tabel 9. Hasil Array S Iterasi 5

INDEX	0	1	2	3	4
S	1	0	4	2	3

- **Langkah 4**

Selanjutnya, membangkitkan aliran-kunci. Lakukan PRGA (*Pseudo-Random Generation Algorithm*) dengan rumus:

$i \leftarrow 0$

$j \leftarrow 0$

$for\ idx \leftarrow 0\ to$

$PanjangPlainteks - 1\ do$

$i \leftarrow (i + 1)\ mod\ 256$

$j \leftarrow (j + S[i])\ mod\ 256$

$Swap(S[i], S[j])$

$t \leftarrow ([i] + [j])\ mod\ 256$

$K \leftarrow S[t]$ $(* keystream)$

Dari rumus tersebut kita akan melakukan perhitungan menggunakan 5 *byte*, dimana iterasi yang akan dijalani adalah sebanyak 5 kali perulangan.

a. Iterasi 1

$i = 0$

$j = 0$

$i = (0 + 1)\ mod\ 5$

$i = 1\ \mathrm{mo}d\ 5$

$i = 1$

$j = (0 + \mathrm{S}[1])\ mod\ 5$

$j = (0 + 0)\ mod\ 5$

$j = 0\ mod\ 5$

$j = 0$

$(\mathrm{S}[1], S[0])$

Tabel 10. Hasil Array S Iterasi 1

INDEX	0	1	2	3	4
S	0	1	4	2	3

$t = (S[1] + S[0])\ mod\ 5$

$t = (1 + 0)\ mod\ 5$

$t = 1\ mod\ 5$

$t = 1$

$\mathrm{K} = \mathrm{S}[1] = 1$

$\mathrm{K} = 00000001$

b. Iterasi 2

$i = 1$

$j = 0$

$i = (1 + 1)\ mod\ 5$

$i = 2\ mod\ 5$

$i = 2$

$j = (0 + \mathrm{S}[2])\ mod\ 5$

$j = (0 + 4)\ mod\ 5$

$j = 4\ \ \mathrm{mo}d\ 5$

$j = 4$

$Swap(S[2], S[4])$

Tabel 11. Hasil Array S Iterasi 2

INDEX	0	1	2	3	4
S	0	1	3	2	4

$t = (S[2] + S[4])\ mod\ 5$

$t = (3 + 4)\ mod\ 5$

$t = 7\ mod\ 5$

$t = 2$

$\mathrm{K} = \mathrm{S}[2] = 3$

$\mathrm{K} = 00000011$

c. Iterasi 3

$i = 2$

$j = 4$

$i = (2 + 1)\ mod\ 5$

$i = 3\ mod\ 5$

$i = 3$

$j = (4 + \mathrm{S}[3])\ mod\ 5$

$j = (4 + 3)\ \mathrm{mod}\ 5$

$j = 7\ mod\ 5$

$j = 2$

$Swap(S[3], S[2])$

Tabel 12. Hasil Array S Iterasi 3

INDEX	0	1	2	3	4
S	0	1	2	3	4

$t = (S[3] + S[2])\ mod\ 5$

$t = (3 + 2)\ mod\ 5$

$t = 5\ mod\ 5$

$t = 0$

$\mathrm{K} = \mathrm{S}[0] = 0$

$\mathrm{K} = 00000000$

d. Iterasi 4

$i = 3$

$j = 2$

$i = (3 + 1)\ mod\ 5$

$i = 4\ mod\ 5$

$i = 4$

$j = (2 + \mathrm{S}[4])\ mod\ 5$

$j = (2 + 4)\ mod\ 5$

$j = 6\ mod\ 5$

$j = 1$

$Swap(S[4], S[1])$

Tabel 13. Hasil Array S Iterasi 4

INDEX	0	1	2	3	4
S	0	4	2	3	1

$t = (S[4] + S[1])\ mod\ 5$

$t = (1 + 4)\ mod\ 5$

$t = 5\ mod\ 5$

$t = 0$

$\mathrm{K} = \mathrm{S}[0] = 0$

$\mathrm{K} = 00000000$

e. Iterasi 5

$i = 4$

$j = 1$

$i = (4 + 1)\ mod\ 5$

$i = 5\ mod\ 5$

$i = 0$

$j = (1 + \mathrm{S}[0])\ mod\ 5$

$j = (1 + 0)\ mod\ 5$

$j = 1\ mod\ 5$

$j = 1$

$Swap(S[0], S[1])$

Tabel 14. Hasil Array S Iterasi 5

INDEX	0	1	2	3	4
S	4	0	2	3	1

$t = (S[0] + S[1])\ mod\ 5$

$t = (4\ + 0)\ mod\ 5$

$t = 4\ mod\ 5$

$t = 4$

$\mathrm{K} = \mathrm{S}[4] = 1$

$\mathrm{K} = 00000001$

Sehingga diperoleh

K = 1 3 0 0 1

K = 00000001 00000011 00000000 00000000 00000001

- **Langkah 5:**

Selanjutnya, kita akan meelakukan enkripsi dengan melakukan operasi XOR antara karakter pada plaintext dengan kunci yang dihasilkan. Tiap-tiap karakter pada plaintext yang digunakan diubah menjadi biner 8 bit ASCII sebagai berikut.

Plaintext : F~T#0

F = 83 = 01000110

~ = 126 = 01111110

T = 84 = 01010100

= 35 = 00100011

0 = 48 = 00110000

Proses XOR dari Plaintext dengan kunci yang telah didapat:

Plaintext	01000110	01111110	01010100	00100011	00110000	XOR
Kunci	00000001	00000011	00000000	00000000	00000001	
Ciphertext	01000111	01111101	01010100	00100011	00110001	

Sehingga, hasil ciphertextnya diubah menjadi ASCII sebagai berikut.

01000111=71= G

01111101= 125= }

01010100 = 84= T

00100011= 35= #

00110001= 49= 1

Diperoleh ciphertextnya dari hasil enkripsi algoritma RC4 menggunakan mode 5 byte adalah G}T#1.

- **Langkah 6**

Plaintext : [71 125 84 35 49]

Kunci : [90 75 2 119 38]

$P_1 = 01000111$

$K_1 = 01011010$ XOR

$C_1 = 00011101$

$P_2 = 01111101$

$K_2 = 01001011$ XOR

$C_2 = 00110110$

$P_3 = 01010100$

$K_3 = 00000010$ XOR

$C_3 = 01010110$

$P_4 = 00100011$

$K_4 = 01110111$ XOR

$C_4 = 01010100$

$P_5 = 00110001$

$K_5 = 00100110$ XOR

$C_5 = 00010111$

- **Langkah 7**

Plaintext : [29 54 86 84 23]

Kunci : [90 75 2 119 38]

$P_1 = 00011101$

$K_1 = 01011010$ XOR

$C_1 = 01000111$

$P_2 = 00110110$

$K_2 = 01001011$ XOR

$C_2 = 01111101$

$P_3 = 01010110$

$K_3 = 00000010$ XOR

$C_3 = 01010100$

$P_4 = 01010100$

$K_4 = 01110111$ XOR

$C_4 = 00100011$

$P_5 = 00010111$

$K_5 = 00100110$ XOR

$C_5 = 00110001$

2. Dekripsi

- **Dekripsi Pesan**

Selanjutnya, untuk proses dekripsi akan dicari plaintext menggunakan OTP putaran 2 terlebih dahulu.Algoritma dekripsi menggunakan XOR adalah dengan meng-XOR-kan *plaintext* (P) dengan kunci (K) menghasilkan *ciphertext* (C) dengan menggunakan Persamaan (2).

Langkah 1 :

Ciphertext : [71 125 84 35 49]

Kunci : [90 75 2 119 38]

$C_1 = 01000111$

$K_1 = 01011010$ XOR

$P_1 = 00011101$

$C_2 = 01111101$

$K_2 = 01001011$ XOR

$P_2 = 00110110$

$C_3 = 01010100$

$K_3 = 00000010$ XOR

$P_3 = 01010110$

$C_4 = 00100011$

$K_4 = 01110111$ XOR

$P_4 = 0101010$

$K_5 = 00100110$ XOR

$C_5 = 00110001$ $P_5 = 00010111$

- **Langkah 2**

Plaintext : [29 54 86 84 23]

Kunci : [90 75 2 119 38]

$C_1 = 00011101$

$K_1 = 01011010$ XOR

$P_1 = 01000111$

$C_2 = 00110110$

$K_2 = 01001011$ XOR

$P_2 = 01111101$

$C_3 = 01010110$

$K_3 = 00000010$ XOR

$P_3 = 01010100$

$C_4 = 01010100$

$K_4 = 01110111$ XOR

$P_4 = 00100011$

$C_5 = 00010111$

$K_5 = 00100110$ XOR

$P_5 = 00110001$

- **Langkah 3**

Plaintext dapat diperoleh dengan proses XOR terhadap ciphertext dengan Keystream (karena dalam proses pencarian Keystream baik enkripsi maupun dekripsi, prosesnya sama) yang telah didapat:

Ciphertext	01000111	01111101	01010100	00100011	00110001	
Kunci	00000001	00000011	00000000	00000000	00000001	XOR
Plaintext	01000110	01111110	01010100	00100011	00110000	

F = 83 = 01000110

~ = 126 = 01111110

T = 84 = 01010100

= 35 = 00100011

0 = 48 = 00110000

Sehingga, didapatkan plaintext pesan awal, yaitu F~T#0.

- **Dekripsi Kunci**

Tahap 1 (Dekripsi Kunci mrnggunkan Rail Fence Cipher)

Enkripsi Kunci : 90 75 2 119 38 75 79 21 15 69

Selanjutnya, akan dilakukan deskripsi dengan algoritma RSA dengan cara sebagai berikut:

$$P_1 = 90^{19}\ mod\ 143 = 116$$

$$P_2 = 75^{19}\ mod\ 143 = 49$$

$$P_3 = 2^{19}\ mod\ 143 = 50$$

$$P_4 = 119^{19}\ mod\ 143 = 115$$
$$P_5 = 38^{19}\ mod\ 143 = 64$$
$$P_6 = 75^{19}\ mod\ 143 = 49$$
$$P_7 = 79^{19}\ mod\ 143 = 105$$
$$P_8 = 21^{19}\ mod\ 143 = 109$$
$$P_9 = 15^{19}\ mod\ 143 = 102$$
$$P_{10} = 69^{19}\ mod\ 143 = 104$$

Tabel 15. Hasil Dekripsi RSA

t	1	2	s	@	1	i	m	f	h
116	49	50	115	64	49	105	109	102	104

Mencari nilai rata-rata dari ascii di atas:

$$rata - rata = \frac{116 + 49 + 50 + 115 + 64 + 49 + 105 + 109 + 102 + 104}{10} = 86$$

Diambil angka terbesar dari rata-rata, yaitu 8. Sehinggan, dilakukan dekiripsi dan didapatkan, yaitu t12s@lmhfi. Selanjutnya, diubah ke dalam bentuk desimal berdasarkan tabel ASCII sehingga menjadi:

Tabel 15. Hasil Dekripsi RFC

t	1	2	s	@	1	m	h	f	i
116	49	50	115	64	49	109	104	102	105

Selanjutnya, akan di geseran ke kanan sebanyak k=3 sehingga hasil dekripsi kunci adalah s@1mhfit (Plaintext awal).

b. Program

- **Source Code dan Hasil Program**

Pembentukan program dilakukan dengan menggunakan Software C++. Dalam pembentukan program akan disusun syntax untuk memproses algoritma RSA, Rail Fence Cipher, RC4 dan OTP.

1. Pertama akan dideklarasikan terlebih dahulu seluruh fungsi dan variabel yang akan digunakan sebagai berikut.

```
#include <iostream>
#include <cmath>
#include <algorithm>
#include<cstdlib>
#include<winsock.h>
#include<math.h>
#include<windows.h>
#include<conio.h>
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
#include <bits/stdc++.h>
```

2. Berikut adalah *source code* untuk *left shift* dan *right shift* sebelum proses enkripsi algoritma RFC dan RSA.

```
  void leftrotate(string &s, int d)
{
   reverse(s.begin(),
   s.begin()+d);
   reverse(s.begin()+d, s.end());
   reverse(s.begin(), s.end());
}
void rightrotate(string &s, int d)
{
  leftrotate(s, s.length()-d);
}
```

3. Berikut adalah *source code* untuk proses enkripsi dan dekripsi dari Rail Fence Cipher.

```
string encryptRailFenceCipher(string plaintext, int k) {
  // Buat matriks untuk menyimpan pesan yang akan dienkripsi
  char matrix[k][plaintext.length()];
  // Inisialisasi matriks dengan karakter spasi
  for (int i = 0; i < k; i++) {
    for (int j = 0; j < plaintext.length(); j++) {
      matrix[i][j] = ' ';
    } }// Isi matriks dengan karakter dari plaintext
  int direction = 1;
  int row = 0;
  for (int i = 0; i < plaintext.length(); i++) {
    matrix[row][i] = plaintext[i];
    // Ubah arah jika sudah mencapai ujung atas atau bawah matriks
    if (row == k - 1) {
      direction = -1;
    } else if (row == 0) {
      direction = 1;
    } // Geser ke baris berikutnya
    row += direction;
  }
  // Buat ciphertext dengan menggabungkan karakter di setiap baris
  string ciphertext = "";
  for (int i = 0; i < k; i++) {
    for (int j = 0; j < plaintext.length(); j++) {
      if (matrix[i][j] != ' ') {
        ciphertext += matrix[i][j];

   } }}
  return ciphertext;
}
```

```
 // Fungsi untuk mendekripsi ciphertext menggunakan Rail Fence Cipher
string decryptRailFenceCipher(string decryptedMessage, int key) {
    // Buat matriks untuk menyimpan plaintext
    char matrix[key][decryptedMessage.length()];
    // Inisialisasi matriks dengan karakter spasi
    for (int i = 0; i < key; i++) {
        for (int j = 0; j < decryptedMessage.length(); j++) {
            matrix[i][j] = ' ';
        }}// Isi matriks dengan ciphertext
    int row = 0, col = 0;
    bool down = true;
    for (int i = 0; i < decryptedMessage.length(); i++) {
        matrix[row][col] = '*';
        if (down) {row++;
            if (row == key) {
                row = key - 2;
                down = false;
            } } else {
            row--;
            if (row == -1) {
                row = 1;
                down = true;}}
        col++;} // Rekonstruksi plaintext dari matriks
    int index = 0;
    for (int i = 0; i < key; i++) {
        for (int j = 0; j <decryptedMessage.length(); j++) {
if (matrix[i][j] == '*') {matrix[i][j] = decryptedMessage[index++];}}}
// Buat string untuk menyimpan hasil dekripsi
string plaintext;
row = 0, col = 0;
down = true;
for (int i = 0; i < decryptedMessage.length(); i++) {
    plaintext += matrix[row][col];
    if (down) { row++;
        if (row == key) {
            row = key - 2;
            down = false;}} else {
        row--;
        if (row == -1) {
            row = 1;
            down = true;
        }}
    col++;}
return plaintext;}
```

4. Berikut adalah *source code* untuk proses enkripsi dan dekripsi dari RSA

```
// Fungsi untuk mengenkripsi pesan dengan menggunakan kunci publik
string encrypt(int e, int n, string ciphertext) {
    string encryptedMessage = "";
    for (char c : ciphertext) {
        int x = (int)c;
        int y = mod(x, e, n);
        encryptedMessage += to_string(y) + " ";}
    return encryptedMessage;}
string decrypt(int d1, int n, string encryptedMessage)
    encryptedMessage+=" ";
    string decryptedMessage = "";
    string number = "";
    for (char c : encryptedMessage){
        if (c == ' ') {
            int x = stoi(number);
            int y = mod(x, d1, n);
            char z = (char)y;
            decryptedMessage += z;
            number = "}
        else {number += c;
        }return decryptedMessage;}
```

5. Berikut adalah *source code* untuk proses enkripsi dan dekripsi dari RC4

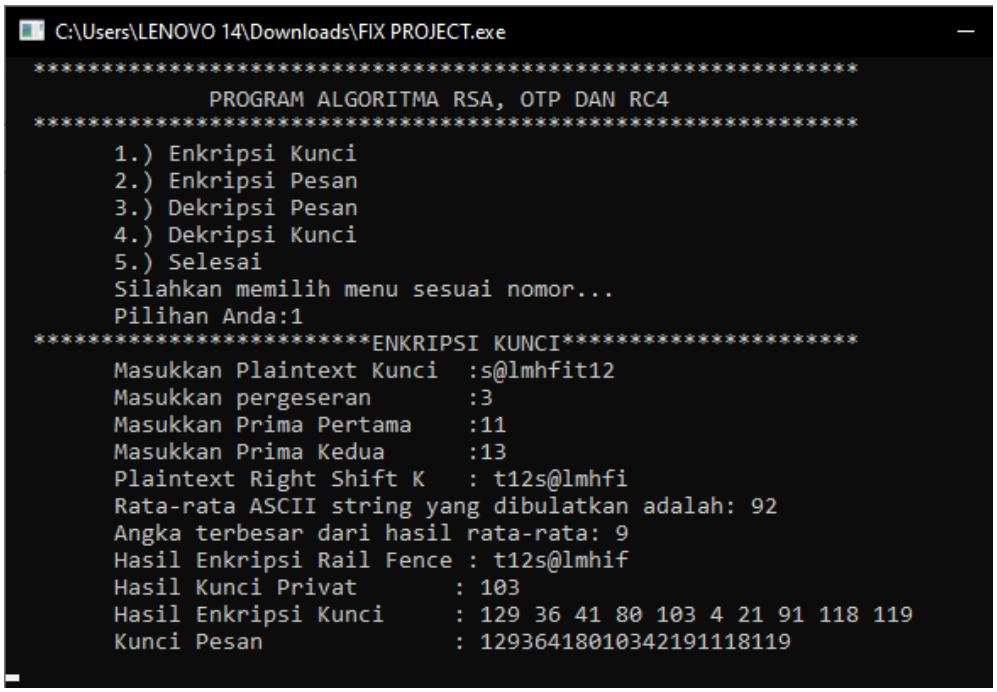
```
string RC4_decrypt(string decrypted1, string str5)
{ // Inisialisasi variabel
    int S[256], T[256];
    for (int i = 0; i < 256; i++)
     S[i] = i; int j = 0;
    for (int i = 0; i < 256; i++)
   {  j = (j + S[i] + str5[i % str5.length()]) % 256;
        swap(S[i], S[j]); } int i = 0, k = 0;
    string hasil_dekripsi = "";
    for (int n = 0; n < decrypted1.length(); n++){
        i = (i + 1) % 256;j = (j + S[i]) % 256;
        swap(S[i], S[j]);
        k = S[(S[i] + S[j]) % 256];
        hasil_dekripsi += char(k ^ decrypted1[n]);
    }return hasil_dekripsi;}
```

```
string RC4(string str2, string str3)
{ // Inisialisasi variabel
    int S[256], T[256];
    for (int i = 0; i < 256; i++)
        S[i] = i;
    int j = 0;
    for (int i = 0; i < 256; i++)
    {j = (j + S[i] + str3[i % str3.length()]) % 256;
swap(S[i], S[j]);
    }int i = 0, k = 0;
    string hasil_enkripsi = "";
    for (int n = 0; n < str2.length(); n++)
    { i = (i + 1) % 256;
        j = (j + S[i]) % 256;
        swap(S[i], S[j]);
        k = S[(S[i] + S[j]) % 256];
hasil_enkripsi += char(k ^ str2[n]);}
       return hasil_enkripsi;}
```

6. Berikut adalah *source code* untuk proses enkripsi dan dekripsi dari OTP

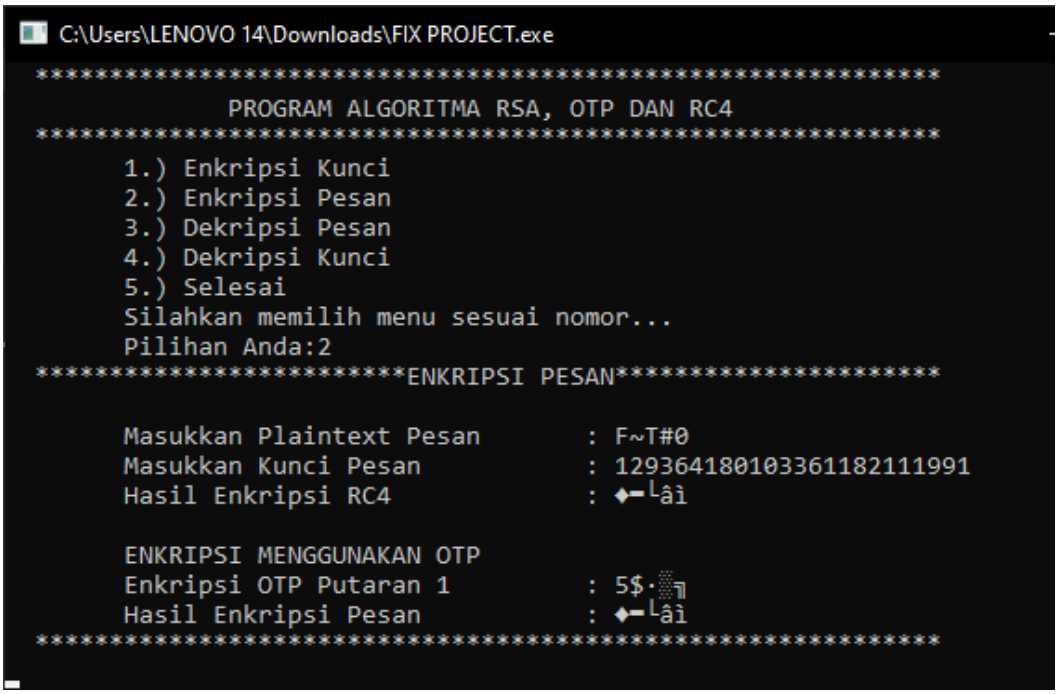
```
string encript(string pesan_enkripsi, string str3)
{
    // Create a result string the same size as the message
    string result(pesan_enkripsi.size(), ' ');

    // Loop through each character in the message
    for (int i = 0; i < pesan_enkripsi.size(); i++)
        {// Encrypt each character by XORing it with the corresponding character in the key
        result[i] = pesan_enkripsi[i] ^ str3[i];
    } return result;
}
string decript(string str4, string str5)
{// Decryption is the same as encryption, just XORing the ciphertext with the key
    return encript(str4, str5);}
```

7. Selanjutnya, setelah menuliskan syntax di C++ maka program akan di build and run untuk melihat hasil program enkripsi kunci dengan memasukkan plaintext kunci, bilangan prima pertama, bilangan prima kedua, dan nilai k pergesaran. Plaintext dan kunci yang diinputkan akan dienkripsi menggunakan algoritma RFC dan RSA sehingga didapatkan hasil enkripsi seperti pada Gambar 1.



Gambar 1. Hasil Program Enkripsi Kunci

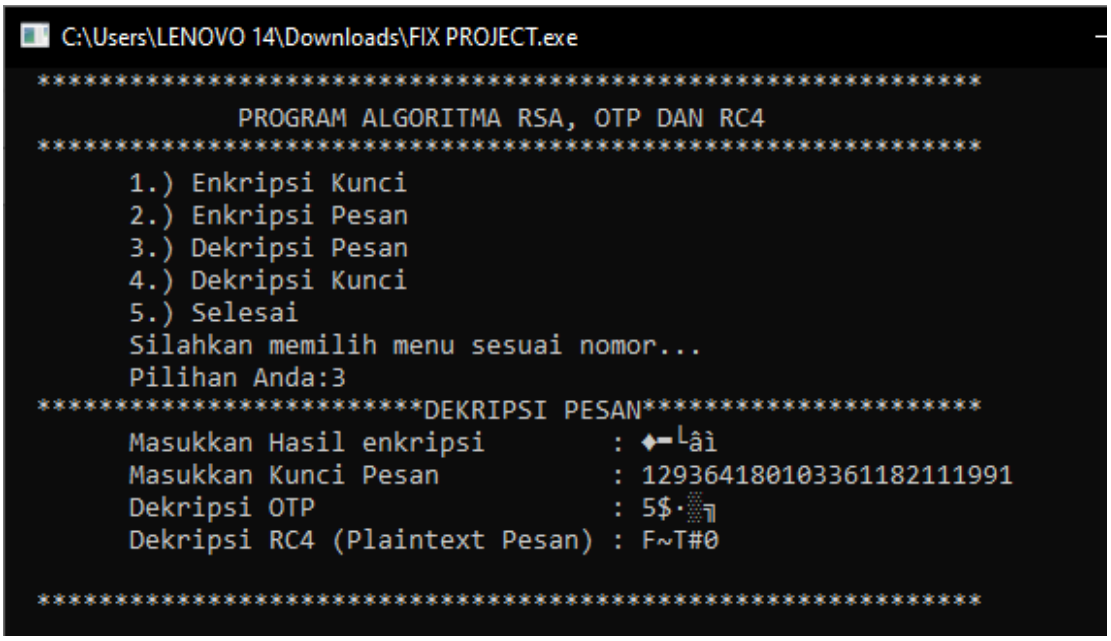
8. Selanjutnya, untuk melihat hasil program enkripsi pesan dengan memasukkan plaintext pesan dan memasukkan kunci yang didapatkan dari Gambar 1. Plaintext dan kunci yang diinputkan akan dienkripsi menggunakan algoritma RC4 dan OTP sehingga didapatkan hasil enkripsi seperti pada Gambar 2.



Gambar 2. Hasil Program Enkripsi Kunci

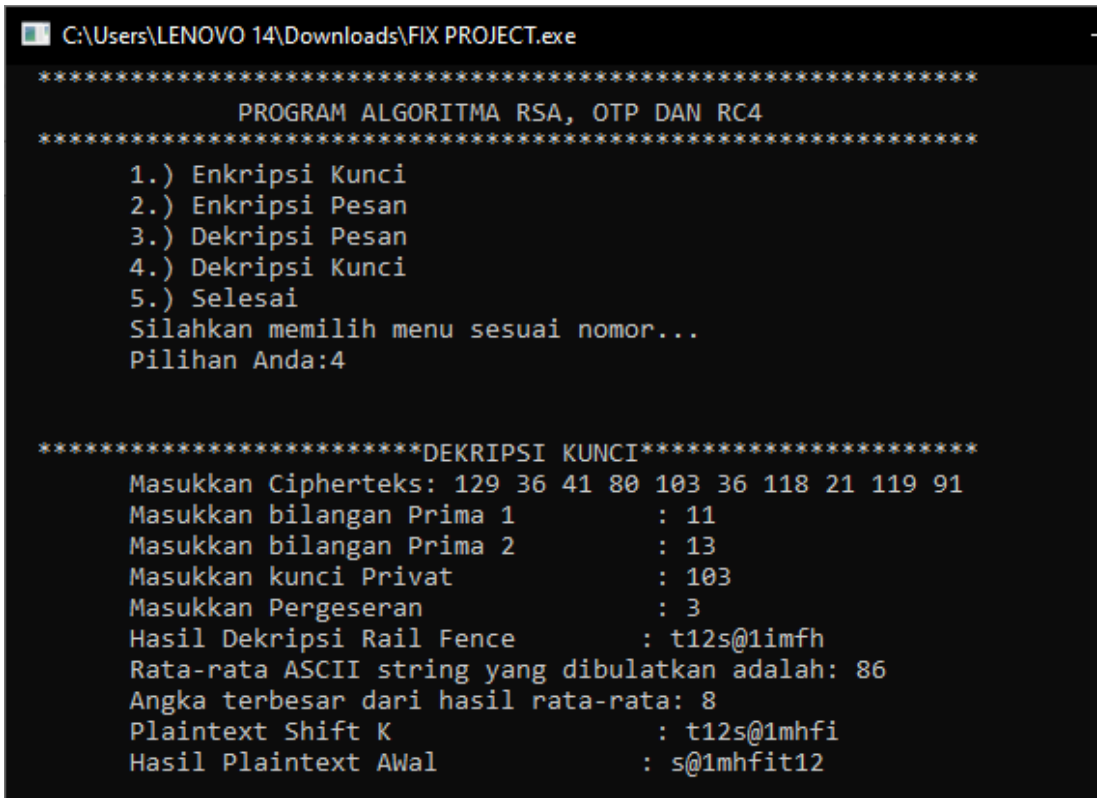
9. Selanjutnya, untuk mendekripsi pesan. Pertama, harus menutup program terlebih dahulu, lalu merunning kembali program dan memilih dekripsi pesan. Selanjutnya, untuk melihat hasil program dekripsi pesan dengan memasukkan hasil enkripsi pesan dapat dilihat pada Gambar 2 dan memasukkan kunci yang didapatkan dari Gambar 1.

Hasil enkripsi dan kunci yang diinputkan akan didekripsi menggunakan algoritma OTP dan RC4 sehingga didapatkan hasil dekripsi seperti pada Gambar 3.



Gambar 3. Hasil Program Dekripsi Pesan

10. Selanjutnya, untuk mendekripsi kunci. Pertama, dengan memasukkan hasil enkripsi kunci, nilai pergeseran k, dan bilangan prima, serta kunci privat yang dapat dilihat pada Gambar 1. Hasil enkripsi kunci yang diinputkan akan didekripsi menggunakan algoritma RFC dan RSA sehingga didapatkan hasil dekripsi kunci seperti Gambar 4.



Gambar 4. Hasil Program Dekripsi Kunci

BAB IV
KESIMPULAN

Program yang dibentuk untuk mengamankan pesan pada pembahasan kali ini adalah program yang mengkombinasikan algoritma RFC, RSA, RC4 dan OTP sehingga pesan yang akan diamankan kerahasiaan pesan terjaga. Program ini dibuat dengan tujuan untuk meningkatkan keamanan pesan. Pada proses enkripsi kunci digunakan dengan algoritma RFC dan Algoritma RSA. Kemudian, hasil enkripsi tersebut akan dijadikan kunci pada enkripsi pesan dengan menggunakan algoritma RC4 dan OTP. Sedangkan, jika proses dekripsi akan dilakukan penutupan program terlebih dahulu. Setelah itu, dilakukan dekripsi pesan dan dekripsi pesan sehingga mendapatkan hasil plaintext awal. Dari pembahasan ini, dapat disimpulkan bahwa 4 algpritma yang digunakan dapat meningkatkan keamanan pesan yang dienkripsikan agar tidak mudah ditebak oleh pihak manapun.

Link Youtube : https://youtu.be/_1FxTxt8rTw

DAFTAR PUSTAKA

Amrulloh, A., Ujianto, E.I.H., 2019. Kriptografi Simetris Menggunakan Algoritma Vigenere Cipher. J. CoreIT 5, 71–77.

Arief, A., Saputra, R., 2016. Implementasi Kriptografi Kunci Publik dengan Algoritma RSA-CRT pada Aplikasi Instant Messaging. Sci. J. Informatics 3, 46–54. https://doi.org/10.15294/sji.v3i1.6115

Arya, D., Virgian, D., Yudha, S., 2022. Implementasi Algoritma Kriptografi Rivest Code 4 Berbasis Web Pada Crypthography Algorithm At Pt . Putri Maharani Medikal. 4, 175–181.

Eko Tinikar, A., Ineke Pakereng, M.A., Alz Danny Wowor, Mk., 2014. Modifikasi Kriptografi One Time Pad (OTP) Menggunakan Padding Dinamis dalam Pengamanan Data File 1–15.

Febriani, D., Purba, D.F., Puspasari, R., 2020. Penerapan Algoritma Rail Fence Untuk Penghasil Pesan Rahasia Berbasis Android. E-Journal.Potensi-Utama.Ac.Id 745–756.

Ginting, A., Isnanto, R.R., Windasari, I.P., 2015. Implementasi Algoritma Kriptografi RSA untuk Enkripsi dan Dekripsi Email. J. Teknol. dan Sist. Komput. 3, 253–258.

Harahap, M.K., Khairina, N., 2018. Analisis Algoritma One Time Pad Dengan Algoritma Cipher Transposisi Sebagai Pengamanan Pesan Teks. J. Penelit. Tek. Inform. 1, 58–62.

Irawan, M.D., 2017. Implementasi Kriptografi Vigenere Cipher Dengan Php. J. Teknol. Inf. 1, 11. https://doi.org/10.36294/jurti.v1i1.21

Jayusman, Y., Apriyadi, D., Rahman, A., Bandung, S., 2017. Sistem Keamanan Short Message Service (Sms) Menggunakan Algoritma Kriptografi Rsa Pada Platform Android. J. Teknol. Inf. dan Komun. 6, 5–10.

Maulana, R., Simanjorang, R.M., 2021. Implementasi Kriptografi Pengamanan Data Pribadi Siswa SMA Swasta Jaya Krama Beringin Dengan Algoritma RC4. J. Nas. Komputasi dan Teknol. Inf. 4, 377–383. https://doi.org/10.32672/jnkti.v4i6.3533

Prabowo, R., Pramusinto, W., 2018. Implementasi Kriptografi dengan Algoritma Vigenere Cipher, AES 128 dan RC 4 untuk Aplikasi Pesan Instan Berbasis Android. J. Skanika 1, 931–937.

Purba, B., Gulo, F.A., Utami, N.I., Sihotang, Y.A., 2020. Pengamanan File Teks Menggunakan Algoritma RC4 420–425.

Ratna, D., 2018. Implementasi Algoritma Rail Fence Chiper dalam Keamanan Data Gambar 2 Dimensi. Pelita Inform. Inf. 7, 38–42.

Sutoyo, A., Nurhayati, Gultom, I., 2019. Implementasi Super Enkripsi Algrotma One Time Pad (OTP) dan Beaufort Cipher untuk Mengamankan Data. J. Sist. Inf. Kaputama 3, 1–5.

Taronisokhi, Z., Ndruru, E., 2017. Pengamanan Citra Digital Berdasarkan Modifikasi

Algoritma RC4 4, 275–282. https://doi.org/10.25126/jtiik.201744474

Zulfikar, M.I., Abdillah, G., Komarudin, A., 2019. Kriptografi untuk Keamanan Pengiriman Email Menggunakan Blowfish dan Rivest Shamir Adleman (RSA). Semin. Nas. Apl. Teknol. Inf. 19–26.