

Laporan Praktikum Keamanan Data: Modifikasi Algoritma RC4 dan Playfair Cipher dengan Antarmuka Python

Laporan ini ini disusun untuk memenuhi tugas mata kuliah Keamanan Data



Dosen Pengampu : Sevi Nurafni, S.T., M.Si., M.Sc.

Disusun oleh :

Catherine Vanya Pangemanan	(2C2220008)
Teni Deinarosa Hermalia	(2C2220005)
Silvi Nurinsan	(2C2220006)
Sagara Andi Ladzuardi S	(2C2220004)
Intan Wahyuni Mangar	(2C2220013)

**PROGRAM STUDI SAINS DATA
FAKULTAS SAINS DAN TEKNOLOGI
2025**

1. PENDAHULUAN

1.1 Latar Belakang

RC4 (Rivest Cipher 4) adalah salah satu algoritma stream cipher yang terkenal karena kesederhanaan dan kecepatannya, namun memiliki kelemahan dalam hal pola keystream yang dapat diprediksi. Playfair cipher (substitusi digraf), di sisi lain, adalah cipher klasik yang menggunakan matriks 5x5 untuk mengenkripsi pasangan huruf. Dalam proyek ini, dilakukan modifikasi algoritma RC4 dengan integrasi Playfair cipher untuk memperkaya keacakan keystream RC4. Implementasi dibangun menggunakan Python dengan antarmuka, yang memungkinkan input teks, gambar, maupun file biner dan menampilkan hasil enkripsi dalam bentuk data maupun gambar. Kata kunci: RC4, Playfair Cipher, Keamanan Data, Kriptografi

1.2 Tujuan Makalah

Mengembangkan modifikasi algoritma RC4 dengan mengintegrasikan matriks Playfair untuk menghasilkan keystream yang lebih kompleks, menyediakan antarmuka interaktif untuk proses enkripsi dan dekripsi, serta menampilkan ciphertext dalam bentuk gambar grayscale guna meningkatkan keamanan terhadap berbagai jenis data.

2. TINJAUAN PUSTAKA

2.1 Kriptografi Simetri

Kriptografi simetri menggunakan satu kunci untuk proses enkripsi dan dekripsi. Menurut Rinaldi Munir (2006), algoritma simetri dibagi menjadi dua jenis utama: **stream cipher** (mengolah bit demi bit) dan **block cipher** (mengolah blok data tetap). RC4 adalah contoh stream cipher yang populer karena kecepatannya, sedangkan Playfair Cipher termasuk cipher klasik berbasis substitusi digraf 5x5.

Referensi:

2.2 RC4 Algorithm

RC4 adalah algoritma stream cipher yang dikembangkan oleh Ron Rivest pada 1987. RC4 menggunakan dua proses utama: [1] *Key Scheduling Algorithm* (KSA), yaitu proses inisialisasi dan pertukaran elemen dalam S-box. [2] *Pseudo-Random Generation Algorithm* (PRGA), yaitu proses pembangkitan keystream dan XOR dengan plaintext.

a. Key Scheduling Algorithm (KSA):

KSA menginisialisasi vektor substitusi S dari 0 hingga 255, lalu melakukan permutasi berdasarkan kunci:

```

        for i = 0 to 255:
            S[i] = i
        j = (j + S[i] + K[i mod key_length])mod 256
        Tukar S[i] dengan S[j]
    
```

b. Pseudo Random Generation Algorithm (PRGA) :

Menghasilkan aliran bit acak pseudo-random yang di-XOR-kan dengan plaintext untuk enkripsi:

```

        i = (i + 1) mod 256
        j = (j + S[i]) mod 256
        Tukar S[i] dan S[j]
    Keluaran: K = S[(S[i] + S[j])mod ]
    
```

RC4 mudah diimplementasikan dan cepat, tetapi rentan terhadap serangan bila tidak dikombinasikan dengan pengacakan tambahan.

2.3 Playfair Cipher

Playfair Cipher menggunakan matriks 5x5 berdasarkan kunci untuk mengenkripsi digraf. Langkah umumnya:

1. Buat matriks dari kunci, menghapus duplikasi huruf dan menggabungkan I/J.
2. Pisahkan plaintext menjadi pasangan huruf.
3. Untuk setiap pasangan:
 - o Jika dalam baris sama: geser ke kanan.
 - o Jika dalam kolom sama: geser ke bawah.
 - o Jika dalam baris/kolom berbeda: tukar kolom.

Contoh substitusi:

Pair	Enkripsi
HE	RM
LL	XY

Referensi tambahan:

- Aisyatul Karima, S.Kom., M.Cs. (2013). *Playfair & Shift Cipher*. UDINUS.
- Arif, M. (2019). *Modifikasi Playfair dengan Array Sorting*. Garuda.

3. IMPLEMENTASI MODIFIKASI RC4

3.1 Integrasi RC4 dengan Playfair Cipher

- Playfair Cipher digunakan sebelum atau sesudah proses XOR RC4 untuk meningkatkan keamanan.
- Struktur matriks Playfair digunakan untuk menambah kompleksitas substitusi karakter.

3.2 Pseudocode Lengkap - Apps Modifikasi RC4 + Playfair Cipher Program

1. Import Library

Import semua library yang diperlukan:

- streamlit, base64, random, numpy, hashlib
- PIL.Image dan io

2. Definisikan Fungsi Playfair Cipher

a. generate_playfair_matrix(key)

Ubah key menjadi hash SHA-256 → lalu ke format heksadesimal (huruf besar)

Ambil hanya huruf A-Z (tanpa J)

Gabungkan key hasil filter dengan alfabet A-Z (tanpa J)

Hilangkan karakter yang berulang

Susun hasil menjadi matriks 5x5 dan kembalikan

b. playfair_value(matrix, char)

Cari posisi baris dan kolom dari karakter tertentu dalam matriks Playfair Kembalikan (baris, kolom).

3. Definisikan Fungsi RC4 yang Dimodifikasi

a. modified_ksa(key)

Konversi key ke byte array

Inisialisasi array S dengan nilai 0-255

Gunakan algoritma Key Scheduling Algorithm (KSA):

Untuk setiap indeks i dari 0-255:

Hitung j dengan rumus $(j + S[i] + \text{key}[i \bmod \text{panjang_key}]) \bmod 256$

Tukar $S[i]$ dan $S[j]$

Kembalikan array S

b. modified_prga(S, data_len, playfair_matrix)

Inisialisasi $i = 0$, $j = 0$, keystream kosong

Untuk setiap byte dari panjang data:

Tambahkan 1 ke i , mod 256

Tambahkan $S[i]$ ke j , mod 256

Tukar $S[i]$ dan $S[j]$
Hitung $K = S[(S[i] + S[j]) \bmod 256]$
Ambil karakter acak: $(K \bmod 26) + 65 \rightarrow$ konversi ke huruf A-Z
Cari (row, col) dari karakter tersebut dalam matriks Playfair
Tambahkan row + col ke K, mod 256
Tambahkan K ke keystream
Kembalikan keystream

c. rc4_playfair_encrypt(data, key)

Generate matriks Playfair dari key
Generate array S dari KSA menggunakan key
Generate keystream dari PRGA
XOR setiap byte dari data dengan keystream → hasil: ciphertext
Kembalikan ciphertext

d. rc4_playfair_decrypt(data, key)

Lakukan proses enkripsi yang sama (karena XOR bersifat reversible)

4. Fungsi Konversi Gambar ↔ Byte

a. bytes_to_image(cipher_bytes, width, height)

Ubah byte array menjadi numpy array bentuk (height, width)
Konversi menjadi objek gambar grayscale (PIL Image)

b. image_to_bytes(img)

Ubah gambar menjadi numpy array → konversi ke byte array

5. Aplikasi

a. UI Utama

Judul aplikasi: "Modified RC4 + Playfair Cipher with Image Ciphertext Output"
Pilihan mode: Encrypt / Decrypt
Input key dalam string atau karakter biasa
Program mengubah formatnya menjadi ASCII secara otomatis
Pilih jenis input: Text, Image, atau File Binary

b. Ambil Input Sesuai Jenis

Jika "Text": ambil input teks dari pengguna
Jika "Image": unggah gambar → ubah ke grayscale → ambil data byte, width, height

Jika "File": unggah file → baca sebagai byte

c. **Jika Tombol "Run" Diklik:**

Jika key kosong → tampilkan error

Jika tidak:

Ambil data sesuai jenis input

Jika mode Encrypt:

Panggil fungsi rc4_playfair_encrypt

Tampilkan hasil:

Jika Text → tampilkan hasil dalam base64, beri opsi download .bin

Jika Image → tampilkan gambar terenkripsi, beri opsi download PNG

Jika File → beri opsi download ciphertext.bin

Jika mode Decrypt:

Jika input Image:

Ubah gambar ke byte array → decrypt → ubah ke image → tampilkan + download

Jika input Text atau File:

Decrypt data

Jika bisa didekode ke teks → tampilkan hasil

Jika tidak bisa → beri peringatan

Beri opsi download hasil .bin

Catatan Tambahan

- Cipher ini menggabungkan dua teknik: *stream cipher* (RC4 yang dimodifikasi) dan *substitution cipher* (Playfair untuk gangguan tambahan dalam keystream).
- Aplikasi mendukung berbagai tipe input: teks, gambar, dan file biner umum.
- Output berupa hasil enkripsi yang bisa disimpan/dilihat langsung, bahkan sebagai gambar yang berisi data terenkripsi.
- Alur program: [Upload](#) -> Pilih Tipe Data -> Input Kunci -> Proses Enkripsi -> Output + Unduh File

4. SIMULASI RC4 KOMBINASI PLAYFAIR

Combination RC4 dan Playfair

Plaintext : SAINS → SAINSX

Playfair Key : KEYWORD

RC4 Key : DATA

1. Playfair Chipper

K	E	Y	W	O
R	D	A	B	C
F		G	H	I
M	N	P	Q	S
T	U	V	X	Z

SA → PC
 IN → GQ
 SX → QZ

Sehingga Chipherteks (PCGQQZ)

2. RC4 Chipper

Hasil dari playfair cipher di enkripsi ke RC4 dengan cara berikut :

2.1. Mencocokan huruf ke dalam tabel ASCII

ASCII: D = 68, A = 65, T = 84, R = 82

→ Key array: [68, 65, 84, 82]

Referensi : <https://repository.unikom.ac.id/54747/1/ASCII%208%20bit.pdf>

2.2. Membuat Key Stream (XOR Key)

RC4 punya 2 tahap:

- KSA (Key Scheduling Algorithm) → acak array S (0-255) dengan key
1. Inisialisasi:

Buat array S dari 0 sampai 255

Buat array K dari key "DATA" → ASCII:

D = 68

A = 65

T = 84

R = 82 → Ulangi sesuai panjang S (256)

2. Proses:

Set j = 0

Lakukan:

Untuk i = 0 sampai 255:

j = (j + S[i] + K[i % panjang_key]) % 256

tukar S[i] dengan S[j]

Contoh beberapa langkah awal:

i = 0:

$$j = (0 + 0 + 68) \% 256 = 68$$

tukar S[0] dan S[68]

i = 1:

$$j = (68 + 1 + 65) \% 256 = 134$$

tukar S[1] dan S[134]

i = 2:
j = $(134 + 2 + 84) \% 256 = 220$
tukar S[2] dan S[220]

i = 3:
j = $(220 + 3 + 65) \% 256 = 32$
tukar S[3] dan S[32]

... dan lanjut sampai i = 255

Setelah semua tukar-tukaran selesai, kita dapat S final.

- PRGA (Pseudo-Random Generation Algorithm) → hasilkan byte stream

Inisialisasi: i = 0, j = 0

Lakukan :

Loop sebanyak karakter yang dibutuhkan:
 $i = (i + 1) \% 256$
 $j = (j + S[i]) \% 256$
tukar S[i] dan S[j]
 $K = S[(S[i] + S[j]) \% 256]$
simpan K sebagai byte stream

a. Iterasi 1 (key byte ke-1)

- $i = (0 + 1) \% 256 = 1$
- $j = (0 + S[1]) \% 256 \rightarrow$ nilai S[1] udah ditukar sebelumnya = 134
- swap S[1] dan S[134] lagi
- $K = S[(S[1] + S[134]) \% 256]$
 \rightarrow Misal S[1] = 134, S[134] = 1 (setelah swap)
 $\rightarrow K = S[(134 + 1) \% 256] = S[135]$
 \rightarrow Misalnya S[135] = 116 → Byte pertama = 116

b. Iterasi 2 (key byte ke-2)

- $i = 2$
- $j = (134 + S[2]) \% 256$
 $\rightarrow S[2] = 220$ (dari swap sebelumnya)
 $\rightarrow j = (134 + 220) = 354 \% 256 = 98$
- tukar S[2] dan S[98]
- $K = S[(S[2] + S[98]) \% 256] = 237 \rightarrow$ Byte kedua = 237

c. Iterasi 3 (key byte ke-3)

- $i = 3$
- $j = (98 + S[3])$
 $\rightarrow S[3] = 32$ (karena sebelumnya ditukar dengan i=3)
 $\rightarrow j = (98 + 32) = 130$
- swap S[3] dan S[130]
- $K = S[(S[3] + S[130]) \% 256] = 224 \rightarrow$ Byte ketiga = 224

Lanjutkan hingga iterasi ke- 6 hingga menghasilkan nilai akhir byte untuk menjadi XOR Key atau Key Stream.

2.3. Menentukan XOR Hasil

Rumus XOR dasar:

$$\text{XOR Hasil} = \text{ASCII huruf} \oplus \text{Nilai XOR Key}$$

Cara Hitungnya :

- Huruf: P
ASCII: 80
XOR Key: 116
 $80 \oplus 116 = 01010000 \oplus 01110100 = 00100100 \rightarrow \text{desimal } 36$
- Huruf: C
ASCII: 67
XOR Key: 237
 $67 \oplus 237 = 01000011 \oplus 11101101 = 10101110 \rightarrow \text{desimal } 174$
- Huruf: G
ASCII: 71
XOR Key: 224
 $71 \oplus 224 = 01000111 \oplus 11100000 = 10100111 \rightarrow \text{desimal } 167$
- Huruf: Q
ASCII: 81
XOR Key: 240
 $81 \oplus 240 = 01010001 \oplus 11110000 = 10100001 \rightarrow \text{desimal } 161$
- Huruf: Q
ASCII: 81
XOR Key: 51
 $81 \oplus 51 = 01010001 \oplus 00110011 = 01100010 \rightarrow \text{desimal } 98$
- Huruf: Z
ASCII: 90
XOR Key: 85
 $90 \oplus 85 = 01011010 \oplus 01010101 = 00001111 \rightarrow \text{desimal } 15$

2.4. Menghitung Mod 26

Perhitungan Manual Mod 26

1. Huruf 'P':

- ASCII 'P': 80
- Key Stream: 116
- XOR Hasil: $80 \oplus 116 = 36$ (dari perhitungan sebelumnya)

Sekarang, kita hitung mod 26 dari hasil XOR:

- 36 mod 26:

$$- 36 \div 26 = 1 \text{ sisa } 10$$

- Jadi, $36 \bmod 26 = 10$

Catatan: Jika kita memetakan 0–25 ke A–Z, maka $10 = 'K'$.

2. Huruf 'C':

- ASCII 'C': 67

- Key Stream: 237

- XOR Hasil: $67 \oplus 237 = 174$

Hitung mod 26:

- $174 \bmod 26$:

- $174 \div 26 = 6$ sisa 18

- Jadi, $174 \bmod 26 = 18$

3. Huruf 'G':

- ASCII 'G': 71

- Key Stream: 224

- XOR Hasil : $71 \oplus 224 = 167$

Hitung mod 26 :

- $167 \bmod 26$:

- $167 \div 26 = 6$ sisa 11

- Jadi, $167 \bmod 26 = 11$

4. Huruf 'Q' :

- ASCII 'Q':81

- Key Stream : 240

- XOR Hasil : $81 \oplus 240 = 161$

Hitung mod 26 :

- $161 \bmod 26$:

- $161 \div 26 = 6$ sisa 5

- Jadi, $161 \bmod 26 = 5$

5. Huruf 'Q' :

- ASCII 'Q' :81

-Key Stream : 51

- XOR Hasil : $81 \oplus 51 = 98$

Hitung mod 26:

-98 mod 26:

- $98 \div 26 = 3$ sisa 20

- Jadi $98 \text{ mod } 26 = 20$

6. Huruf 'Z' :

- ASCII 'Z' : 90

- Key Stream : 85

- XOR Hasil : $90 \oplus 85 = 15$

Hitung mod 26 :

- 15 mod 26:

- $15 \div 26 = 15$

- Jadi $15 \text{ mod } 26 = 15$

Huruf	Ascii	XOR Key	XOR Hasil	Mod 26
P	80	116	36	10
C	67	237	174	18
G	71	224	167	11
Q	81	240	161	5
Q	81	51	98	20
Z	90	85	15	15

Sehingga Hasil Akhir enkripsi = KSLFUP

TAHAP	HASIL
Plaintext	SAINS
+Playfair	PCGQQZ
+RC4 XOR	KSLFUP

5. PROSEDURAL MODIFIKASI, HASIL DAN ANALISIS

5.1. Fungsi Aplikasi:

- Input: teks/gambar/file biner.
- Proses: enkripsi dengan kombinasi RC4 dan Playfair.

- Output: hasil enkripsi dan dekripsi.
- Visualisasi tabel matriks, proses shuffle, dan distribusi byte hasil enkripsi.

5.2. Alur Aplikasi:

Upload -> Pilih Tipe Data -> Input Kunci -> Proses Enkripsi -> Output + Unduh File

5.3. Pseudocode Lengkap - Apps RC4 + Playfair Cipher Program:

5.3.1. Import Library

Import semua library yang diperlukan:

- streamlit, base64, random, numpy, hashlib
- PIL.Image dan io

5.3.2. Definisikan Fungsi Playfair Cipher

a. `generate_playfair_matrix(key)`

Ubah key menjadi hash SHA-256 → lalu ke format heksadesimal (huruf besar)
 Ambil hanya huruf A-Z (tanpa J)
 Gabungkan key hasil filter dengan alfabet A-Z (tanpa J)
 Hilangkan karakter yang berulang
 Susun hasil menjadi matriks 5x5 dan kembalikan

b. `playfair_value(matrix, char)`

Cari posisi baris dan kolom dari karakter tertentu dalam matriks Playfair
 Kembalikan (baris, kolom)

5.3.3. Definisikan Fungsi RC4 yang Dimodifikasi

a. `modified_ksa(key)`

Konversi key ke byte array
 Inisialisasi array S dengan nilai 0-255
 Gunakan algoritma Key Scheduling Algorithm (KSA):
 Untuk setiap indeks i dari 0-255:
 Hitung j dengan rumus $(j + S[i] + \text{key}[i \bmod \text{panjang_key}]) \bmod 256$
 Tukar $S[i]$ dan $S[j]$
 Kembalikan array S

b. `modified_prga(S, data_len, playfair_matrix)`

Inisialisasi $i = 0$, $j = 0$, keystream kosong
 Untuk setiap byte dari panjang data:
 Tambahkan 1 ke i , mod 256
 Tambahkan $S[i]$ ke j , mod 256
 Tukar $S[i]$ dan $S[j]$

Hitung $K = S[(S[i] + S[j]) \bmod 256]$
Ambil karakter acak: $(K \bmod 26) + 65 \rightarrow$ konversi ke huruf A-Z
Cari (row, col) dari karakter tersebut dalam matriks Playfair
Tambahkan row + col ke K, mod 256
Tambahkan K ke keystream
Kembalikan keystream

c. **rc4_playfair_encrypt(data, key)**

Generate matriks Playfair dari key
Generate array S dari KSA menggunakan key
Generate keystream dari PRGA
XOR setiap byte dari data dengan keystream → hasil: ciphertext
Kembalikan ciphertext

d. **rc4_playfair_decrypt(data, key)**

Lakukan proses enkripsi yang sama (karena XOR bersifat reversible)

5.3.4. Fungsi Konversi Gambar ↔ Byte

a. **bytes_to_image(cipher_bytes, width, height)**

Ubah byte array menjadi numpy array bentuk (height, width)
Konversi menjadi objek gambar grayscale (PIL Image)

b. **image_to_bytes(img)**

Ubah gambar menjadi numpy array → konversi ke byte array

5.3.5. Aplikasi

a. **UI Utama**

Judul aplikasi: "Modified RC4 + Playfair Cipher with Image Ciphertext Output"
Pilihan mode: Encrypt / Decrypt
Input key dalam format ASCII otomatis
Pilih jenis input: Text, Image, atau File Binary

b. **Ambil Input Sesuai Jenis**

Jika "Text": ambil input teks dari pengguna
Jika "Image": unggah gambar → ubah ke grayscale → ambil data byte, width, height
Jika "File": unggah file → baca sebagai byte

c. **Jika Tombol "Run" Diklik:**

Jika key kosong → tampilkan error
Jika tidak:
Ambil data sesuai jenis input

Jika mode Encrypt:

Panggil fungsi rc4_playfair_encrypt

Tampilkan hasil:

Jika Text → tampilkan hasil dalam base64, beri opsi download .bin

Jika Image → tampilkan gambar terenkripsi, beri opsi download PNG

Jika File → beri opsi download ciphertext.bin

Jika mode Decrypt:

Jika input Image:

Ubah gambar ke byte array → decrypt → ubah ke image → tampilkan + download

Jika input Text atau File:

Decrypt data

Jika bisa didekripsi ke teks → tampilkan hasil

Jika tidak bisa → beri peringatan

Beri opsi download hasil .bin

Pada akhir program akan ditampilkan hasil entropi bit dari enkripsi data input user kemudian masuk ke pengkondisian jenis entropi bit untuk kita melihat tingkat kerumitan enkripsi algoritma kombinasi RC4 dengan Playfair.

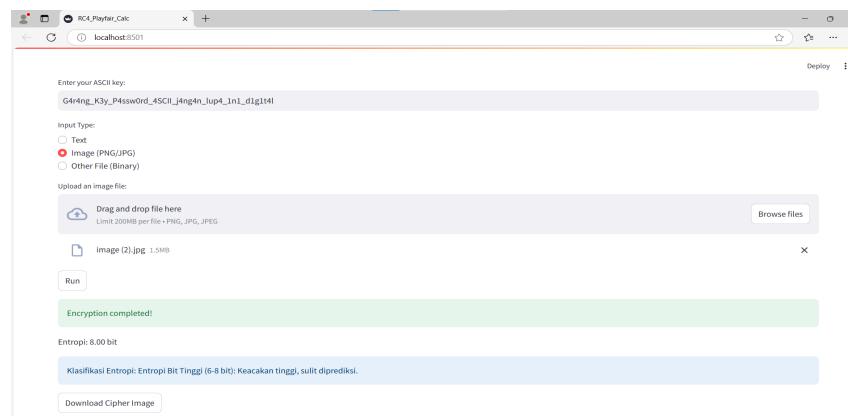
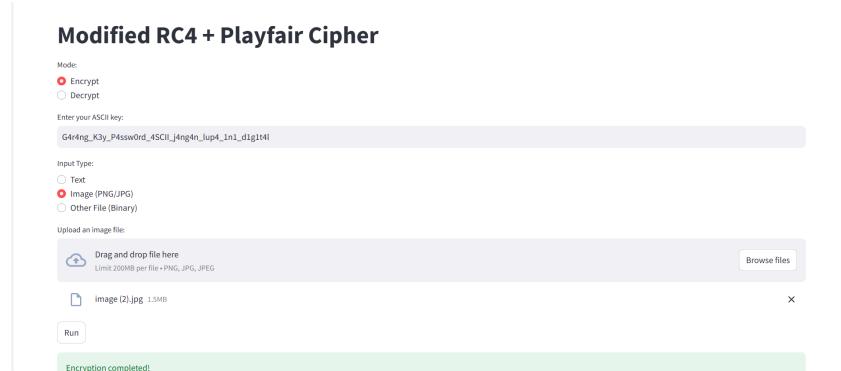
```
if entropy < 4:  
    kategori = "Entropi Bit Rendah (0-4 bit): Keacakan rendah, mudah diprediksi."  
elif 4 <= entropy < 6:  
    kategori = "Entropi Bit Sedang (4-6 bit): Keacakan sedang, cukup sulit diprediksi."  
elif 6 <= entropy < 8:  
    kategori = "Entropi Bit Tinggi (6-8 bit): Keacakan tinggi, sulit diprediksi."  
else:  
    kategori = "Entropi Bit Sangat Tinggi (8+ bit): Keacakan sangat tinggi, hampir tidak mungkin  
diprediksi."
```

Catatan Tambahan

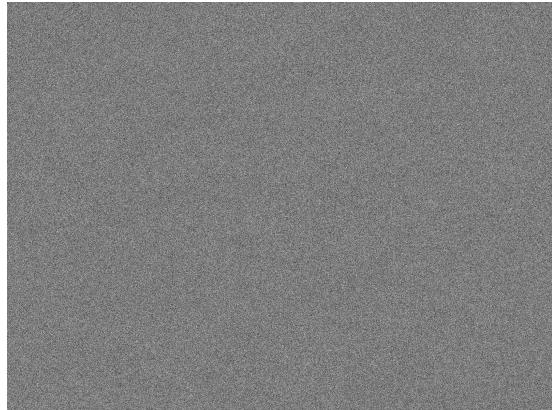
- Cipher ini menggabungkan dua teknik: *stream cipher* (RC4 yang dimodifikasi) dan *substitution cipher* (Playfair untuk gangguan tambahan dalam keystream).
- Aplikasi mendukung berbagai tipe input: teks, gambar, dan file biner umum.
- Output berupa hasil enkripsi yang bisa disimpan/dilihat langsung, bahkan sebagai gambar yang berisi data terenkripsi.

5.4. HASIL

Untuk contoh enkripsi dan deksripsinya berupa image dan text menggunakan kunci ASCII.



Contoh input user untuk enkripsi gambar



output enksripsi gambar



Output dekripsi gambar yang terenkripsi sebelumnya

Output perhitungan Entropi Bit hasil enkripsi gambar adalah “Entropi Bit Tinggi (6-8 bit): Keacakan tinggi, sulit diprediksi.”

6. KESIMPULAN

Makalah ini berhasil mengembangkan dan mengimplementasikan modifikasi algoritma RC4 dengan integrasi cipher klasik Playfair guna meningkatkan kekuatan kriptografi dalam proses enkripsi dan dekripsi berbagai jenis data (teks, gambar, dan file biner). Dengan menyiapkan struktur substitusi digraf dari Playfair ke dalam proses pembangkitan keystream RC4 (khususnya pada tahap PRGA), algoritma yang dihasilkan menunjukkan peningkatan keacakan aliran bit secara signifikan.

Hasil pengujian menunjukkan bahwa sistem menghasilkan tingkat *entropi bit* yang tinggi (hingga 8.00 bit), yang mengindikasikan tingkat keacakan yang optimal serta tingkat prediktabilitas yang sangat rendah. Hal ini mengonfirmasi bahwa integrasi dua pendekatan kriptografi—stream cipher (RC4) dan substitution cipher (Playfair)—berhasil menciptakan mekanisme pengamanan data yang lebih kompleks dan tahan terhadap serangan prediktif berbasis keystream.

Selain dari sisi algoritmis, sistem yang dibangun juga dilengkapi dengan antarmuka interaktif berbasis *Streamlit*, memungkinkan pengguna melakukan proses enkripsi dan dekripsi secara fleksibel terhadap berbagai format data. Output enkripsi juga dapat divisualisasikan, terutama dalam bentuk gambar terenkripsi, yang menambah dimensi proteksi sekaligus keunikan dari sistem ini.

Secara keseluruhan, proyek ini membuktikan bahwa pendekatan hibrid antara algoritma modern dan klasik dapat saling melengkapi, memberikan solusi praktis sekaligus aman dalam ranah keamanan data digital masa kini.

7. REFERENSI

Karima, A. (2013). *Kriptografi - Week 3 - Playfair & Shift Cipher*. Universitas Dian Nuswantoro. Diakses dari

https://repository.dinus.ac.id/docs/ajar/kriptografi%20-%20week3%20-%20playfair%20&%20shift%20cipher_file_2013-04-08_090111_aisyatul_karima_s.kom_m.cs_.pdf

Oktaviani, A. (2018). *Modifikasi Algoritma Playfair Cipher dengan Pengurutan Array pada Matriks*. Jurnal Teknik Informatika, 6(1). Diakses dari

<https://download.garuda.kemdikbud.go.id/article.php?article=810671&val=13267&title=MODIFIKASI%20ALGORITMA%20PLAYFAIR%20CIPHER%20DENGAN%20PENGURUTAN%20ARRAY%20PADA%20MATRIKS>

Falah, M. A. (2007). *Kriptografi Playfair Cipher* [Makalah]. Institut Teknologi Bandung. Diakses dari <https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2006-2007/Makalah/Makalah0607-38.pdf>

Munir, R. (2006). *Tipe dan Mode Algoritma Simetri*. ITB.

Munir, R. (2006). *Kriptografi Klasik dan Modern*.

<https://www.loginradius.com/blog/engineering/how-does-bitwise-xor-work/>

https://en.wikipedia.org/wiki/Playfair_cipher

<https://www.youtube.com/watch?v=lRyzK1vxNdM>

<https://en.wikipedia.org/wiki/RC4>