IMPLEMENTASI RC4 (RIVEST CIPHER 4)

DAN VISUALISASI KEYSTREAM



**Anggota Kelompok :**

* + - 1. Lio Kusnata (G1A023013)
      2. Rahman Firdaus Ilahi (G1A023055)
      3. Abim Bintang Audio (G1A023073)
      4. Ajis Saputra Hidayah (G1A023083)

**Dosen Pengampu :**

1. Ir. Kurnia Anggraini, S.T., M.T.,Ph.D

SEMESTER 4

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS BENGKULU**

**2025**

BAB I

**PENDAHULUAN**

## Latar Belakang Masalah

Perkembangan teknologi informasi yang sangat pesat telah mengubah cara manusia menyimpan, mengakses, dan mendistribusikan informasi. Berbagai data penting mulai dari dokumen pribadi, institusional, hingga nasional kini dikemas dalam bentuk digital. Namun, di balik kemudahan tersebut, muncul pula ancaman serius terhadap keamanan informasi: manipulasi, penyadapan, hingga pencurian data menjadi tantangan utama, terutama bagi dokumen digital seperti file teks, dokumen penggajian, maupun soal ujian (Febriyani & Arfriandi, 2021).

Sebagai solusi, kriptografi hadir untuk menjamin kerahasiaan, integritas, otentikasi, dan non-repudiation informasi. Salah satu algoritma kunci simetris yang paling dikenal dalam kategori stream cipher adalah RC4 (*Rivest Cipher 4*). RC4 menggunakan proses *Key-Scheduling Algorithm* (KSA) untuk menginisialisasi state-array, diikuti oleh *Pseudo-Random Generation Algorithm* (PRGA) yang menghasilkan deretan keystream pseudo-acak berukuran byte per byte, lalu di XOR kan dengan plaintext atau ciphertext (Saragi et al., 2020). Keunggulan RC4 terletak pada kecepatan prosesnya sekitar sepuluh kali lebih cepat dibanding DES serta kesederhanaan implementasi, membuatnya cocok untuk aplikasi enkripsi file teks dan sistem real-time .

Berbagai penelitian telah membuktikan efektivitas RC4 dalam konteks yang beragam. (Purba et al., 2020) menunjukkan keberhasilan pengamanan file teks melalui aplikasi berbasis web menggunakan RC4, sedangkan (Setiawan & Fatimah, 2021) mengimplementasikannya untuk mengamankan database penggajian karyawan berbasis PHP/MySQL dengan hasil yang memadai. Selain itu, pengujian pada beberapa sampel ciphertext oleh (Saragi et al., 2020) membuktikan bahwa, dengan implementasi yang tepat, RC4 tahan terhadap crackStation dan brute-force attack pada tingkat praktis.

Meskipun sederhana dan cepat, RC4 memiliki kelemahan mendasar, bias dalam byte keystream awal dan kemungkinan pengulangan pola dalam state-array yang dapat dieksploitasi melalui *known-plaintext* atau statistical attacks. Berbagai mitigasi telah diusulkan, seperti penggunaan *Initialization Vector* (IV) berbeda setiap sesi, pembusaan state, atau pembuangan sejumlah byte awal keystream sebelum digunakan(Purba et al., 2020). Visualisasi keystream RC4 menjadi alat penting untuk menganalisis distribusi nilai byte yang dihasilkan. Dengan memetakan frekuensi byte keystream dalam bentuk grafik atau heatmap, pengguna dapat mengidentifikasi bias atau ketidaksempurnaan algoritma, serta mengevaluasi efektivitas dan effisiensi nya.

Melalui latar belakang inilah, maka laporan ini disusun untuk mensimulasikan algoritma RC4 dan memvisualisasikan keystream yang dihasilkan. Penelitian ini bertujuan untuk memahami cara kerja algoritma RC4, serta mengetahui bagaimana visualisasi keystreamnya agar dapat menganalisis pola keystream. Dengan pendekatan ini, diharapkan laporan ini dapat memberikan kontribusi dalam memperkaya pemahaman tentang kriptografi klasik, serta memberikan gambaran terhadap tantangan dan solusi dalam sistem keamanan informasi.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, maka rumusan masalah dalam proyek ini dapat dirumuskan sebagai berikut:

1. Bagaimana prinsip kerja algoritma RC4 dalam menghasilkan keystream untuk proses enkripsi dan dekripsi data secara efisien?
2. Apa kelebihan dan kekurangan penggunaan keystream berbasis RC4 dibandingkan dengan algoritma stream cipher XOR sederhana?
3. Bagaimana efektivitas implementasi algoritma enkripsi ringan berbasis keystream (seperti RC4 dan XOR) dalam menjaga keamanan data digital, terutama pada sistem dengan keterbatasan sumber daya?

## 1.3 Tujuan Proyek

Tujuan dari proyek ini adalah:

1. Menganalisis dan memahami cara kerja algoritma RC4, khususnya dalam hal pembangkitan keystream dan penerapannya dalam proses enkripsi menggunakan operasi XOR.
2. Mengembangkan dan mengimplementasikan algoritma stream cipher ringan berbasis XOR yang dapat dibandingkan secara performa dan efisiensi terhadap pendekatan RC4.
3. Memberikan solusi alternatif enkripsi sederhana yang cocok untuk penggunaan pada data teks atau citra digital, terutama di lingkungan dengan keterbatasan komputasi seperti aplikasi QR Code, pengamanan pesan singkat, atau sistem validasi digital.
4. Melakukan evaluasi terhadap kehandalan algoritma dari sisi kecepatan, ukuran output, serta ketahanan dasar terhadap serangan brute force atau kelelahan kunci (key exhaustion).

## 1.4 Ruang Lingkup Proyek

Agar pengembangan aplikasi enkripsi RC4 ini fokus dan terarah, ruang lingkup proyek dibatasi pada hal-hal berikut:

1. Algoritma Kriptografi yang Digunakan

Proyek ini hanya mengimplementasikan algoritma RC4 (Rivest Cipher 4) sebagai metode enkripsi dan dekripsi berbasis stream cipher dengan operasi logika XOR. Algoritma RC4 digunakan untuk menghasilkan keystream yang dikombinasikan dengan plaintext/ciphertext.

1. Jenis Data yang Didukung

Aplikasi ini terbatas untuk mengenkripsi dan mendekripsi data dalam bentuk teks. Data citra, file biner, atau media lainnya tidak termasuk dalam cakupan pengujian dan implementasi.

1. Antarmuka Pengguna (GUI)

Aplikasi dibangun menggunakan pustaka Tkinter sebagai GUI utama. Antarmuka terdiri dari dua mode:

* Enkripsi: Input berupa plaintext dan kunci, menghasilkan ciphertext dalam format heksadesimal.
* Dekripsi: Input berupa ciphertext heksadesimal dan kunci, menghasilkan plaintext yang dapat dibaca.

1. Fungsi Visualisasi Keystream

Aplikasi menyediakan fitur tambahan berupa visualisasi keystream menggunakan pustaka Matplotlib dan Seaborn, dalam bentuk:

* Bar Chart
* Line Chart
* Heatmap
* Histogram
* Scatter Plot

Visualisasi hanya dilakukan setelah proses enkripsi atau dekripsi, dan ditampilkan dalam jendela terpisah.

1. Platform dan Teknologi

Aplikasi ini:

* Dibangun menggunakan bahasa pemrograman Python 3.
* Memanfaatkan pustaka bawaan dan eksternal: tkinter, matplotlib, seaborn.
* Dirancang untuk berjalan di platform desktop (Windows/Linux/macOS), tidak untuk web atau mobile.

1. Ruang Pengujian dan Evaluasi

Pengujian terbatas pada performa aplikasi dalam mengenkripsi dan mendekripsi teks berukuran pendek hingga sedang, serta distribusi keystream secara statistik melalui visualisasi.

1. Keterbatasan Sistem

* Tidak mencakup sistem keamanan tingkat lanjut seperti proteksi brute force, enkripsi asimetris, atau otentikasi.
* Tidak ada penyimpanan data secara lokal atau integrasi dengan basis data atau cloud.
* Validasi terbatas pada input teks dan format hex yang sesuai.

# BAB II

# TINJAUAN PUSTAKA

## 2.1 Kriptografi

Kriptografi berasal dari bahasa Yunani “*kryptos*” yang berarti tersembunyi, dan “*graphein*” yang berarti tulisan. Dalam konteks modern, kriptografi adalah ilmu yang mempelajari teknik matematika untuk menjaga keamanan informasi dalam sistem komputer. Tujuan utama dari kriptografi adalah menjaga kerahasiaan (*confidentiality*), integritas data (*data integrity*), autentikasi (*authentication*), dan ketiadaan penyangkalan (*non-repudiation*).

Kriptografi modern menggunakan algoritma enkripsi dan dekripsi dengan melibatkan kunci (key). Informasi asli yang disebut plaintext akan dienkripsi menjadi ciphertext menggunakan suatu algoritma dan kunci tertentu, dan sebaliknya, ciphertext dapat dikembalikan ke plaintext melalui proses dekripsi.

## 2.2 Algoritma RC4

RC4 adalah algoritma kriptografi yang dikembangkan oleh Ron Rivest pada tahun 1987 dan termasuk dalam kategori stream cipher simetris. RC4 memproses data byte per byte menggunakan keystream yang dihasilkan dari proses *Key Scheduling Algorithm (KSA)* dan *Pseudo-Random Generation Algorithm (PRGA)*. Proses enkripsi dilakukan dengan operasi XOR antara plaintext dan keystream, sehingga efisien dalam hal waktu dan tidak memerlukan padding seperti pada algoritma block cipher.

Kelebihan RC4 antara lain:

* Proses enkripsi sangat cepat (sekitar 10x lebih cepat dari DES).
* Dapat digunakan untuk data dengan panjang variatif.
* Implementasi sederhana dan ringan secara komputasi.

Namun, RC4 juga memiliki kelemahan, yaitu keystream awal cenderung menampilkan pola yang dapat dimanfaatkan dalam serangan seperti known-plaintext attack. Oleh karena itu, penggunaannya disarankan dengan kunci unik untuk setiap proses enkripsi dan disertai inisialisasi acak (IV) untuk meningkatkan keamanannya.

## 2.3 Keysteam dan Operator XOR

Keystream adalah urutan byte acak semu yang dihasilkan dari algoritma RC4 dan digunakan untuk mengenkripsi atau mendekripsi data. Dalam RC4, keystream dihasilkan oleh array yang telah diacak menggunakan kunci, dan prosesnya melibatkan operasi pertukaran (swap) elemen array untuk menghasilkan hasil yang tidak terprediksi. Enkripsi dilakukan dengan menggunakan operasi XOR antara keystream dan plaintext, sementara dekripsi dilakukan dengan XOR antara keystream dan ciphertext.

## 2.4 Penelitian Terkait

Penelitian oleh Febriyani & Arfriandi (2021) menunjukkan bahwa RC4 berhasil digunakan dalam pengamanan dokumen soal ujian berbasis web, dengan pengujian yang menunjukkan hasil enkripsi tidak dapat dipecahkan menggunakan CrackStation. Penelitian oleh Setiawan & Fatimah (2021) menerapkan RC4 dalam sistem penggajian karyawan dan menyimpulkan bahwa algoritma ini cukup efektif untuk keamanan database (sams). Purba et al. (2020) juga menegaskan pentingnya penggunaan RC4 untuk pengamanan file teks, terutama dalam menghadapi serangan brute force, dengan penekanan pada penggunaan kunci yang unik dan panjang. Saragi et al. (2020) menggarisbawahi bahwa RC4 adalah salah satu metode stream cipher yang dapat diimplementasikan dengan efisien untuk menjaga kerahasiaan dokumen berbasis Word.

# BAB III

# DESAIN DAN METODOLOGI

Aplikasi ini dirancang untuk melakukan proses enkripsi dan dekripsi teks menggunakan algoritma stream cipher RC4, serta menyajikan visualisasi keystream untuk keperluan analisis dan pembelajaran. Antarmuka aplikasi dibangun berbasis Graphical User Interface (GUI) dengan menggunakan pustaka Tkinter, yang memungkinkan pengguna melakukan interaksi secara intuitif.

Terdapat dua fitur utama dalam aplikasi ini, yaitu:

1. Mode Enkripsi: Pengguna memasukkan plaintext dan kunci, kemudian sistem menghasilkan ciphertext dalam format heksadesimal.
2. Mode Dekripsi: Pengguna memasukkan ciphertext heksadesimal dan kunci, kemudian sistem mengembalikan teks asli (plaintext).

Selain itu, aplikasi juga menyediakan fitur **visualisasi keystream** menggunakan berbagai jenis grafik, seperti bar chart, line chart, heatmap, histogram, dan scatter plot. Hal ini bertujuan untuk menganalisis pola keystream yang dihasilkan oleh RC4.

Metodologi implementasi sistem ini mencakup beberapa tahapan berikut:

Input: Teks asli (plaintext) dan kunci (key) dalam bentuk string.

Langkah-langkah:

* 1. Kunci dikonversi ke dalam bentuk byte array (key.encode()).
  2. Dilakukan proses KSA (Key Scheduling Algorithm) untuk menghasilkan array S.
  3. Array S digunakan dalam PRGA (Pseudo-Random Generation Algorithm) untuk menghasilkan keystream sepanjang plaintext.
  4. Proses XOR dilakukan antara plaintext byte dan keystream untuk menghasilkan ciphertext dalam bentuk byte dan dapat dikonversi menjadi heksadesimal.
  5. Ciphertext dikonversi ke format heksadesimal untuk ditampilkan kepada pengguna.

Proses Dekripsi

Input: Ciphertext dalam bentuk heksadesimal dan kunci.

Langkah-langkah:

* 1. Ciphertext heksadesimal dikonversi ke byte array.
  2. Kunci diproses melalui KSA dan PRGA untuk menghasilkan keystream sepanjang ciphertext.
  3. Ciphertext di-XOR dengan keystream untuk menghasilkan kembali plaintext.
  4. Hasil plaintext ditampilkan dalam bentuk string teks.

Visualisasi Keystream

Setelah proses enkripsi atau dekripsi, keystream disimpan dalam memori.

Pengguna dapat memilih jenis visualisasi:

* Bar Chart: Menampilkan nilai byte keystream secara vertikal.
* Line Chart: Menampilkan tren nilai keystream sebagai garis berurutan.
* Heatmap: Menampilkan distribusi keystream dalam bentuk warna.
* Histogram: Menampilkan frekuensi kemunculan nilai byte.
* Scatter Plot: Menampilkan sebaran nilai byte terhadap indeks.
* Statistik: Menampilkan Statistik untuk hasil

Visualisasi dibuat menggunakan pustaka matplotlib dan seaborn, yang di-render ke dalam jendela Tkinter dengan menggunakan FigureCanvasTkAgg.

Arsitektur Antarmuka Pengguna (GUI)

Antarmuka pengguna dibagi menjadi tiga bagian utama:

* Tab Enkripsi: Berisi input plaintext dan kunci, serta output ciphertext.
* Tab Dekripsi: Berisi input ciphertext heksadesimal dan kunci, serta output plaintext.
* Frame Visualisasi: Berisi tombol-tombol untuk menampilkan berbagai bentuk grafik keystream.

|  |  |
| --- | --- |
| Komponen | Detail |
| Bahasa Pemograman | Python 3 |
| GUI Library | Tkinter |
| Visualisasi | Matplotlib |
| IDE/Editor | Visual sStudio Code/PyCharm |
| Sistem Operasi | Windows/Linux/MacOS |
| Library Tambahan | Matplotlib.backends.backend\_tkagg untuk integrasi plot |

Data Uji atau Simulasi yang dilakukan :

Pada table ini merupakan hasil perhitungan waktu pada aplikasi berdasarkan berapa banyak waktu yang diperlukan untuk setiap katanya:

|  |  |  |  |
| --- | --- | --- | --- |
| ***NO.*** | ***KATA PAINTEXT*** | ***WAKTU*** | ***OUTPUT*** |
| 1 | <100 | 0,1 s | Chipertext (Berhasil) |
| 2 | 100-10.000 | 0,1 s | Chipertext (Berhasil) |
| 3 | >10.000 | 0,1 s | Pesan ERROR |

Perbandingan data output untuk bagian Enkripsi dan Dekripsi pada aplikasi yang telah kami lakukan dengan mengunakan key yang sama.

Enkripsi:

|  |  |  |  |
| --- | --- | --- | --- |
| ***NO.*** | ***PAINTEXT*** | ***KEY*** | ***CHIPERTEXT*** |
| 1 | INFORMATIKA | KEY | dc2eba1be4c205ac7fcfba |
| 2 | PRODI INFORMATIKA | KEY | c532b310ffaf0db670cba9cbb10b9a6637 |
| 3 | UNIVERSITAS BENGKULU | KEY | c02eb502f3dd17b162c5a8a6b21a9d6a3d603462 |
| 4 | FAKULTAS TEKNIK | KEY | d321b701fadb05ab16d0becdbe1698 |
| 5 | ANGKATAN 2023 | KEY | d42ebb1ff7db05b616b6cbb4c3 |
| 6 | SEMESTER GENAP | KEY | c625b111e5db01aa16c3bec8b10f |
| 7 | MATAKULIAH KERIPTOGRAFI | KEY | d821a815fdda08b177ccdbcdb50d9a7d227a3f651df5dc |
| 8 | TAHUN 2025 BULAN JUNI | KEY | c121b401f8af76c804b1dbc4a5139263567f2d7915 |

Dekripsi:

|  |  |  |  |
| --- | --- | --- | --- |
| ***NO.*** | ***CHIPERTEXT*** | ***KEY*** | ***HASIL DEKRIPSI*** |
| 1 | dc2eba1be4c205ac7fcfba | KEY | INFORMATIKA |
| 2 | c532b310ffaf0db670cba9cbb10b9a6637 | KEY | PRODI INFORMATIKA |
| 3 | c02eb502f3dd17b162c5a8a6b  21a9d6a3d603462 | KEY | UNIVERSITAS BENGKULU |
| 4 | d321b701fadb05ab16d0becdbe1698 | KEY | FAKULTAS TEKNIK |
| 5 | d42ebb1ff7db05b616b6cbb4c3 | KEY | ANGKATAN 2023 |
| 6 | c625b111e5db01aa16c3bec8b10f | KEY | SEMESTER GENAP |
| 7 | d821a815fdda08b177ccdbcdb50d9a7d  227a3f651df5dc | KEY | MATAKULIAH KERIPTOGRAFI |
| 8 | c121b401f8af76c804b1dbc4a51392  63567f2d7915 | KEY | TAHUN 2025 BULAN JUNI |

Data pengujian terhadap penggunaan KEY pada aplikasi ini

Enkripsi:

|  |  |  |  |
| --- | --- | --- | --- |
| ***NO.*** | ***PAINTEXT*** | ***KEY*** | ***CHIPERTEXT*** |
| 1 | INFORMATIKA | KEY | dc2eba1be4c205ac7fcfba |
| 2 | INFORMATIKA | 12KEY | a5e1d5c75bd6a589fb92d3 |
| 3 | INFORMATIKA | KE23 | 5c69080b6b975f31e0c27d |
| 4 | INFORMATIKA | KE Y | 110129add7dfc1f21c7a15 |
| 5 | INFORMATIKA | 123456 | 49b63828767e54c8367c63 |
| 6 | INFORMATIKA |  | error |

Deskripsi:

|  |  |  |  |
| --- | --- | --- | --- |
| ***NO.*** | ***CHIPERTEXT*** | ***KEY*** | ***HASIL DEKRIPSI*** |
| 1 | dc2eba1be4c205ac7fcfba | KEY | INFORMATIKA |
| 2 | a5e1d5c75bd6a589fb92d3 | 12KEY | INFORMATIKA |
| 3 | 5c69080b6b975f31e0c27d | KE23 | INFORMATIKA |
| 4 | 110129add7dfc1f21c7a15 | KE Y | INFORMATIKA |
| 5 | 49b63828767e54c8367c63 | 12345 | INFORMATIKA |
| 1 | dc2eba1be4c205ac7fcfba |  | error |

Data perbandingan Keystream dalam enkripsi paintext yang sama menggunkan key yang berbeda dengan menghapus satu karakter huruf didalmnya.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| NO. | Paintext | key | waktu | Chipertext | Keystream |
|  |  |  |  |  |  |
| 1 | Informatika adalah bidang yang dinamis dan terus berkembang, berfokus pada studi komputasi dan informasi | kunci | 0,1 s | caa82eec3e24a00248302  789923ef213c0b  8b99e1add02e0  195a447e5a  9b9ef038abdf05e  8f8292d9c57  6c4a4f29a5ecb  e382504b8ac3de  9a9c6cfee910  00df5d95a016112253008f0  49d9ebb22ab86  e5acf243a8733  8bc0106b176de  b1cc5d377eaa97bcd307b63 | [131, 198, 72, 131, 76, 73, 193, 118, 33, 91, 70, 169, 243, 90, 147, 127, 161, 208, 153, 252, 115, 185, 99, 142, 126, 122, 61, 31, 52, 252, 190, 148, 81, 197, 190, 104, 129, 139, 9, 73, 253, 57, 76, 62, 42, 91, 208, 159, 158, 90, 64, 118, 211, 201, 80, 139, 200, 168, 168, 194, 177, 98, 104, 135, 191, 53, 106, 20, 97, 5, 64, 105, 148, 40, 249, 152, 198, 95, 220, 7, 122, 164, 75, 87, 247, 70, 255, 161, 99, 2, 55, 9, 138, 114, 229, 186, 25, 140, 198, 9, 160, 81, 8, 10] |
| 2 | Informatika adalah bidang yang dinamis dan terus berkembang, berfokus pada studi komputasi dan informasi | kunci | 0,1 s | a7b4daf871dd670c6b8  b59759046e  1900bf4ca89b1acde25e  7358320c8c9eda3933  4456718166af49943e82b  f4dd3b2dcf6bd092  e7d7c11420526  421381f6  4ae28b1c509aa2  f8a23278905aa08350  5352a3716b7f4a  35614fd10374  2e0a52ad6  2089c3b4d72ba797 | [238, 218, 188, 151, 3, 176, 6, 120, 2, 224, 56, 85, 241, 34, 128, 252, 106, 156, 234, 235, 216, 200, 191, 75, 128, 21, 250, 65, 166, 174, 205, 199, 250, 90, 36, 10, 113, 101, 74, 144, 248, 45, 200, 95, 145, 175, 78, 94, 239, 9, 181, 224, 140, 178, 172, 118, 65, 60, 3, 13, 24, 125, 1, 220, 78, 222, 174, 124, 217, 15, 250, 66, 67, 232, 37, 217, 124, 64, 97, 92, 10, 92, 121, 218, 132, 214, 34, 117, 142, 121, 23, 38, 129, 203, 10, 191, 78, 239, 172, 198, 186, 74, 212, 254] |

# BAB IV

# IMPLEMENTASI

1. Penjabaran Kode Program atau Hasil Konfigurasi

Kode program ini merupakan aplikasi desktop yang menggunakan antarmuka grafis (GUI) dan memakai bahasa pemrograman *python.* Program ini dirancang agar dapat melakukan simulasi menggunakan algoritma RC4 dan memvisualisasikan keystream yang dihasilkan.

1. Fungsi Inti RC4

Pada bagian ini terdapat dua fase utama dalam algoritma RC4 yang akan digunakan untuk melakukan proses enkripsi dan dekripsi.

* *Key-Scheduling Algoritma* (KSA) 🡪 Fase ini berfungsi untuk menghasilkan state array S yang berisi 256 byte yaitu nilai dari 0-255. Lalu array ini akan diacak berdasarkan kunci yang dimasukkan oleh pengguna.
* *Pseudo-Random Generation Algorithm* (PRGA) 🡪 Setelah fase pertama, PRGA ini akan digunakan untuk menghasilkan sebuah keystream, yaitu urutan byte yang terlihat acak. Dimana panjang keystream yang dihasilkan akan sama dengan panjang plaintext yang dimasukkan oleh pengguna.
* Enkripsi dan Dekripsi 🡪 Terdapat fungsi enkripsi dan dekripsi dalam program ini yang memiliki tugas untuk menyandikan dan mengembalikan pesan. Proses enkripsi ini akan menghasilkan ciphertext dalam bentuk heksadesimal agar lebih ringkas dan mudah untuk dibaca. Proses ini menggunakan operasi XOR sederhana antara data dengan keystream.
* Enkripsi 🡪 Ciphertexti = Plaintexti ⊕ Keystreami
* Dekripsi 🡪 Plaintext = Ciphertexti ⊕ Keystreami

1. Fungsi Analisis dan Visualisasi Keystream

Fungsi analisis ini berfungsi untuk menghitung statistik dari keystream yang dihasilkan, seperti nilai rata-rata, standar deviasi, nilai minimum/maksimum, dan jumlah nilai unik. Statistik ini sangat membantu dalam menganalisis keytream yang dihasilkan karena keamanan RC4 bergantung pada keacakan dari keystream nya.

Selain itu pengguna dapat memvisualisasikan keystream dalam beberapa bentuk grafik untuk mengidentifikasi pola, anomali, ataupun kelemahan dalam keystream secara visual. Jenis grafik yang tersedia yaitu :

* + Bar, Line, dan Scatter Plot 🡪 Untuk melihat sebaran dan urutan nilai byte dalam keystream
  + Heatmap 🡪 Memberikan representassi visual ringkas nilai-nilai byte secara keseluruhan
  + Histogram 🡪 Menampilkan distribusi frekuensi dari setiap nilai byte.

1. Antarmuka Penggunaan (GUI)

GUI dibuat dengan menggunakan Tkinter, yang memungkinkan pengguna dapat

* Beralih mode enkripsi atau dekripsi
* Memasukkan kunci dan plaintext atau ciphertext
* Melakukan enkripsi dan dekripsi
* Melihat visualisasi keystream
* Melihat statistik dari keystream
* Membuat kunci random

1. Contoh Input-Output

* Enkripsi

Key 🡪 Kripto123

Plaintext 🡪 Informatika Jaya

Ciphertext 🡪 db5df2a98d5e5bc54f5fdd41e28d0aec

Keystream 🡪 [146, 51, 148, 198, 255, 51, 58, 177, 38, 52, 188, 97, 168, 236, 115, 141]

* Dekripsi

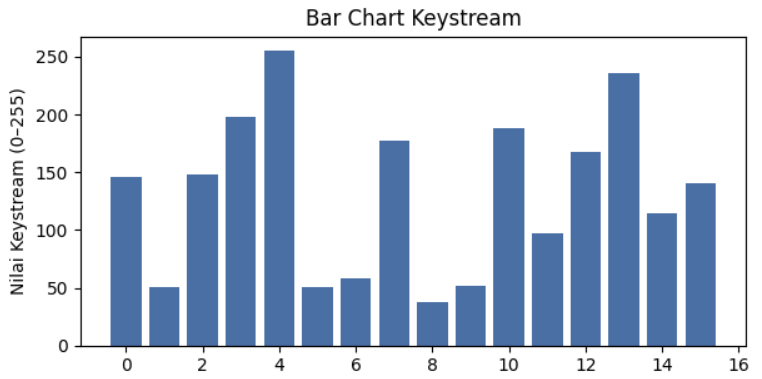
Key 🡪 Kripto123

Ciphertext 🡪 db5df2a98d5e5bc54f5fdd41e28d0aec

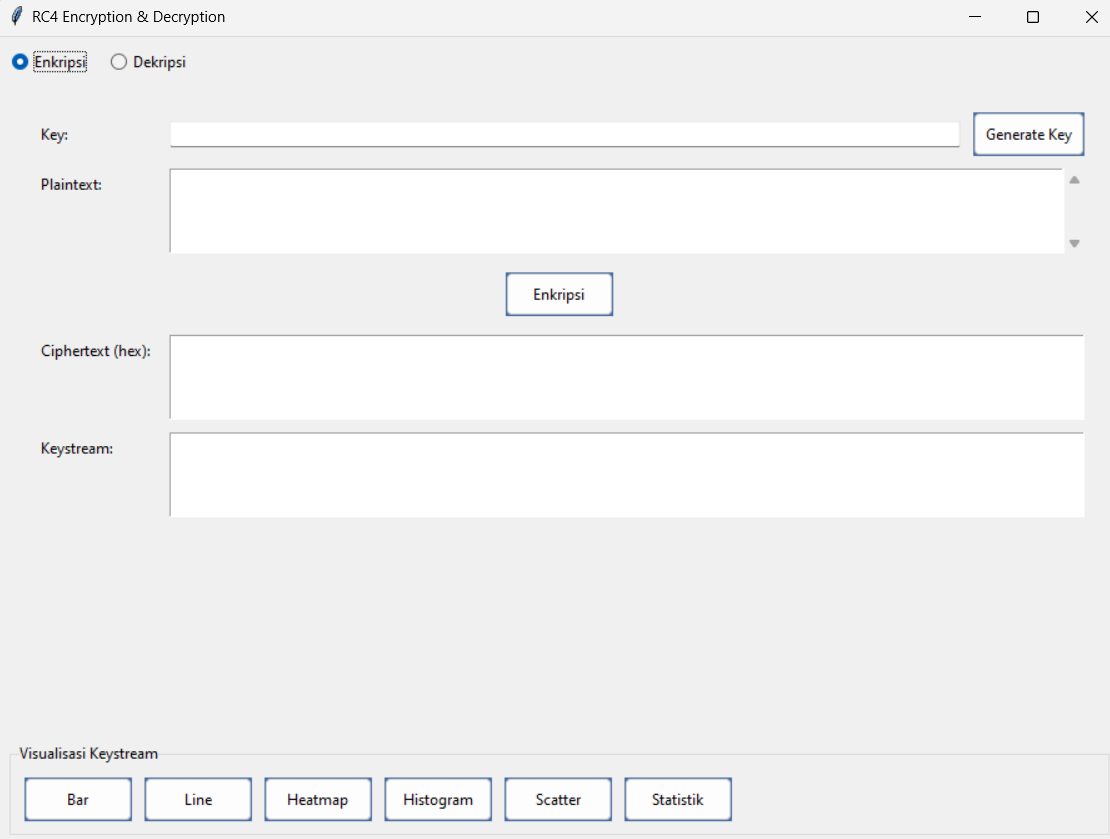
Plaintext 🡪 Informatika Jaya

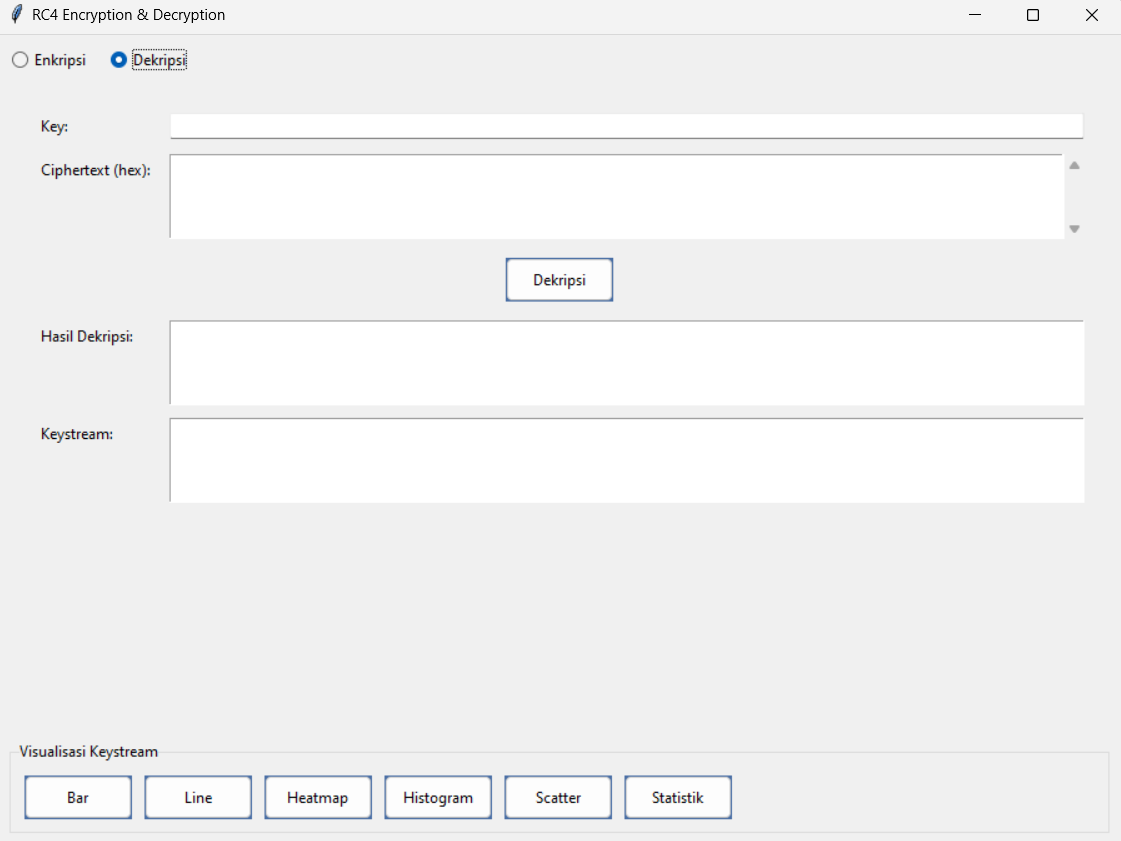
Keystream 🡪 [146, 51, 148, 198, 255, 51, 58, 177, 38, 52, 188, 97, 168, 236, 115, 141]

* Visualisasi Keystream



1. Tampilan Antarmuka





# BAB V

# PENGUJIAN DAN ANALISIS

Aplikasi RC4 yang diimplementasikan menyediakan fungsi utama, yaitu enkripsi dan dekripsi serta visualisasi keystream, dengan keystream yang dihasilkan dari proses KSA dan PRGA. Berikut hasil pengujian dari aspek performansi, akurasi, dan waktu eksekusi:

1. Performansi

* Aplikasi berjalan cukup responsif untuk teks pendek hingga menengah (1–10.000 karakter).
* Waktu eksekusi untuk proses enkripsi dan dekripsi berlangsung hampir kurang dari 0.1 detik pada setiap percobaan.
* Untuk teks panjang (lebih dari 10.000 karakter), pada penggunaan aplikassi akan mengeluarkan sebuah pesan notifikaasi error.

1. Akurasi

* Proses enkripsi menghasilkan ciphertext dalam bentuk heksadesimal yang konsisten dengan RC4.
* Proses dekripsi berhasil mengembalikan plaintext asli tanpa kehilangan atau kesalahan data.
* Validasi dilakukan dengan mencocokkan hasil enkripsi → dekripsi dan hasilnya selalu identik dengan input awal.

1. Waktu Eksekusi

Pengujian dilakukan dengan plaintext yang memiliki panjang berbeda. Hasil menunjukkan bahwa algoritma RC4 dalam aplikasi ini bekerja sangat cepat karena sifat dasar algoritma stream cipher yang ringan. Waktu eksekusi untuk proses enkripsi dan dekripsi terhadap plaintext sepanjang 1-10.000 karakter hanya membutuhkan waktu di kisaran 1 milidetik yang menjadikannya sangat responsif dan cocok untuk teks pendek maupun sedang. Namun dalam pengunjian yang telah dilakukan pada aplikasi ini memiliki kekurangan dalam menyelsaikan plaintext yang memiliki panjang >10.000 yang akan menampilkan sebuah notifikasi ERROR.

RC4 secara historis digunakan secara luas (misal: WEP, SSL), namun saat ini dianggap tidak aman untuk banyak aplikasi kriptografi modern. RC4 memiliki beberapa kelemahan seperti :

* Bias awal dalam keystream: byte awal dari keystream tidak sepenuhnya acak.
* Berpotensi untuk serangan key recovery jika digunakan dalam kondisi yang salah, seperti penggunaan ulang key untuk pesan berbeda.
* Tidak ada autentikasi: Ciphertext dapat dimodifikasi tanpa terdeteksi.

Kemudian aplikasi ini juga memiliki kelemahan pada sistemnya diantaranya :

* Tanpa Salt atau IV (Initialization Vector):Aplikasi tidak menggunakan salt atau IV, sehingga jika key dan plaintext sama, ciphertext juga akan sama. Ini membuka peluang pattern attack.
* Tidak Ada Validasi Key atau Panjang Keystream:Pengguna bisa memasukkan key terlalu pendek atau terlalu panjang tanpa batasan, yang bisa berdampak pada kestabilan atau kekuatan enkripsi.
* Tidak Aman untuk Data Sensitif:Karena tidak ada perlindungan terhadap sniffing, injection, atau logging, sistem ini tidak aman untuk mengenkripsi data penting seperti password, dokumen rahasia, dan lainnya.
* Tidak Terenkripsi Saat Disimpan atau Ditampilkan:Hasil ciphertext dan keystream langsung ditampilkan ke pengguna tanpa perlindungan lebih lanjut (misalnya masking, checksum, dll).

Terdapat Evektivitas dan efisiensi pada aplikasi ini yaitu :

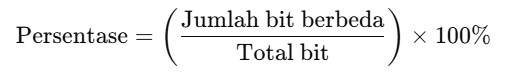
1. Evektivitas

* Aplikasi berhasil melakukan fungsi enkripsi dan dekripsi dengan hasil yang valid dan akurat.
* Antarmuka pengguna terbagi dua (enkripsi & dekripsi), memudahkan penggunaan.
* Tersedia fitur visualisasi keystream dengan berbagai grafik (bar, line, scatter, heatmap, histogram) yang sangat berguna untuk analisis keacakan dan distribusi keystream.

1. Efisiensi

* Konsumsi sumber daya aplikasi rendah, tidak memerlukan koneksi internet atau dependensi berat.
* Respons aplikasi cepat dan efisien, cocok digunakan untuk pembelajaran maupun demonstrasi kriptografi.
* Struktur modular dan pemisahan fungsi membuat kode mudah dipelihara dan dikembangkan lebih lanjut.

Pada data kami juga menguji untuk bagian *avalanche effect.* Dengan mengunakan konsep dasar untuk menghitung perbedaan keystream secara bitwise (*bitwise difference*) adalah dengan menggunakan Hamming Distance, yaitu jumlah bit yang berbeda antara dua keystream dengan panjang yang sama. Dengan rumus berikut:



Pada hasil dari dua percobaan RC4 dengan perbedaan kunci kecil (“Kunci” dan “Kunc”), diperoleh perbedaan keystream sekitar 54,33% secara bitwise. Ini menunjukkan bahwa RC4 memiliki efek avalanche yang cukup kuat, di mana perubahan kecil pada kunci menghasilkan output yang sangat berbeda. Hal ini mencerminkan karakteristik kriptografi yang baik, meskipun RC4 secara umum sudah tidak direkomendasikan untuk sistem keamanan modern seperti data diatas.

Perbandingan hasil enkripsi RC4 dengan AES secara kuantitatif dari berapa jurnal yang telah kami rangkum seabagi berikut ini:

|  |  |  |
| --- | --- | --- |
| **Parameter** | **AES (Jurnal)** | **RC4** |
| **Avalanche Effect (%)** | 46% - 50% (misal: 49.6% rata-rata) | 53% (biasanya lebih rendah dari jika secara data umum AES) |
| **Waktu Enkripsi** | 3-4 ms (desktop) / ~0.8 ms (IoT) | Lebih cepat, bisa di bawah 1 ms |
| **Waktu Dekripsi** | Mirip enkripsi, beberapa ms | Sama dengan enkripsi, sangat cepat |
| **Keamanan** | Sangat kuat, tahan kriptoanalisis | Sudah rentan, tidak disarankan untuk data sensitif |
| **Ukuran kunci** | 128,192,256 bit | Variabel, biasanya 40-256 bit |
| **Penggunaan** | Standar industri & pemerintah | Legacy, protokol lama (WEP), sekarang jarang dipakai |

* Dalam perbandingan ini RC4 menawarkan kecepatan enkripsi yang lebih tinggi (bisa di bawah 1 ms di desktop), namun memiliki kelemahan signifikan dalam hal keamanan. Algoritma ini sudah rentan dan tidak disarankan untuk data sensitif. Meskipun ukuran kuncinya variabel (biasanya 40-256 bit), kerentanannya menjadikannya pilihan yang buruk untuk aplikasi modern. Penggunaan RC4 saat ini bersifat *legacy* dan sudah jarang dipakai, terutama karena kelemahannya yang dieksploitasi dalam protokol lama seperti WEP. Efek *avalanche* RC4, meskipun 53%, tapi dalam berapa percobaan dalam sebuah jurnal seringkali lebih rendah dari yang diharapkan untuk keamanan yang kuat.

# BAB VI

# KESIMPULAN DAN SARAN

## KESIMPULAN

Program aplikasi simulasi RC4 dan visualisasi keystream ini berhasil mengimplementasikan algoritma RC4. Dimana program ini dapat melakukan proses enkripsi dan dekripsi secara akurat, memberikan visualisasi keystream melalui berbagai grafik seperti diagram batang, Visualisasi ini membantu pengguna memahami pola dan distribusi nilai acak pada keystream yang dihasilkan RC4, sehingga dapat menjadi media pembelajaran kriptografi yang efektif. Selain itu statistik keystream dapat membantu dalam menganalisis keystream. Hal ini dapat menjadikan program aplikasi sebagai alat untuk menganalisis kualitas keacakan keystream dalam kriptografi. Aplikasi yang dibangun dengan Python dan antarmuka Tkinter memberikan hasil yang akurat, dan responsif, serta dapat digunakan sebagai media pembelajaran kriptografi klasik yang interaktif dan informatif.

Melalui simulasi yang telah dilakukan, keamanan algoritma RC4 bergantung pada kualitas statistik dari keystream yang dihasilkan. Algoritma ini menggunakan kunci rahasia untuk menghasilkan byte *pseudo* acak, yang kemudian di XOR kan dengan data. Jika keystream ini memiliki pola yang dapat dideteksi, maka enkripsi dapat dipecahkan. Oleh karena itu, statistik keystream yang ada di dalam program secara langsung dapat menunjukkan kelemahan dari algoritma RC4.

## SARAN

Meskipun aplikasi telah berjalan dengan baik dan memiliki fitur dasar yang lengkap, masih terdapat beberapa aspek yang dapat ditingkatkan untuk pengembangan ke depan. Pertama, dari segi keamanan, algoritma RC4 sudah tidak lagi direkomendasikan untuk penggunaan di sistem keamanan tingkat lanjut karena kelemahan pada struktur keystream awal. Oleh karena itu, disarankan untuk menambahkan algoritma stream cipher modern seperti ChaCha20 atau varian RC4 yang telah dimodifikasi agar lebih aman. Kedua, validasi input perlu diperketat, misalnya dengan membatasi panjang kunci, mendeteksi karakter yang tidak valid, serta memastikan format hexadecimal yang benar. Ketiga, fitur ekspor dan impor data juga perlu ditambahkan agar hasil enkripsi dapat disimpan atau digunakan kembali tanpa perlu proses manual. Keempat, tampilan antarmuka dapat disesuaikan agar lebih responsif dan dapat digunakan pada berbagai ukuran layar, termasuk untuk versi web atau perangkat mobile.

# DAFTAR PUSTAKA

Febriyani, F. S., & Arfriandi, A. (2021). Implementasi Algoritma RC4 pada Sistem Pengamanan Dokumen Digital Soal Ujian. *JISKA (Jurnal Informatika Sunan Kalijaga)*, *6*(3), 171–177. https://doi.org/10.14421/jiska.2021.6.3.171-177

Purba, B., Apustriani Gulo, F., Indah Utami, N., & Annisa Sihotang, Y. (2020). Seminar Nasional Teknologi Komputer & Sains (SAINTEKS) Pengamanan File Teks Menggunakan Algoritma RC4. *Seminar Nasional Teknologi Komputer & Sains (SAINTEKS)*, *February*, 420–425.

Saragi, D. R., Gultom, J. M., Tampubolon, J. A., & Gunawan, I. (2020). Pengamanan Data File Teks (Word) Menggunakan Algoritma RC4. *Jurnal Sistem Komputer Dan Informatika (JSON)*, *1*(2), 114. https://doi.org/10.30865/json.v1i2.1745

Setiawan, A., & Fatimah, T. (2021). Implementasi Algoritma Kriptografi Rc4 Untuk Keamanan Database Aplikasi Penggajian Karyawan Berbasis Web Pada Pt. Trans Intra Asia. *Skanika*, *4*(1), 66–71. https://doi.org/10.36080/skanika.v4i1.2044

# LAMPIRAN

Listing Program :

import tkinter as tkfrom tkinter import ttk, messagebox, filedialogimport matplotlib.pyplot as pltimport seaborn as snsfrom matplotlib.backends.backend\_tkagg import FigureCanvasTkAggimport mathimport randomimport string# === RC4 Core ===def ksa(key): key\_length = len(key) S = list(range(256)) j = 0 for i in range(256): j = (j + S[i] + key[i % key\_length]) % 256 S[i], S[j] = S[j], S[i] return Sdef prga(S, n): i = j = 0 keystream = [] for \_ in range(n): i = (i + 1) % 256 j = (j + S[i]) % 256 S[i], S[j] = S[j], S[i] K = S[(S[i] + S[j]) % 256] keystream.append(K) return keystreamdef rc4(key, n): S = ksa(key) return prga(S, n)def encrypt\_rc4(plaintext, keystream): plaintext\_bytes = plaintext.encode('utf-8') return [plaintext\_bytes[i] ^ keystream[i] for i in range(len(plaintext\_bytes))]def decrypt\_rc4(cipher\_bytes, keystream): return bytes([cipher\_bytes[i] ^ keystream[i] for i in range(len(cipher\_bytes))]).decode('utf-8', errors='replace')def bytes\_to\_hex(cipher\_bytes): return ''.join(f'{b:02x}' for b in cipher\_bytes)def hex\_to\_bytes(hex\_string): try: return bytes.fromhex(hex\_string) except ValueError: return None# === GUI App ===class RC4App: def \_\_init\_\_(self, root): self.root = root self.root.title("RC4 Encryption & Decryption") self.root.geometry("900x650") self.root.resizable(True, True) # Set theme colors self.bg\_color = "#f0f0f0" self.primary\_color = "#4a6fa5" self.success\_color = "#d4edda" self.error\_color = "#f8d7da" self.warning\_color = "#fff3cd" style = ttk.Style() style.configure("TButton", padding=6, relief="flat", background=self.primary\_color) style.configure("TFrame", background=self.bg\_color) style.configure("TLabel", background=self.bg\_color, padding=3) style.configure("TLabelframe", background=self.bg\_color) style.configure("TLabelframe.Label", background=self.bg\_color) # Mode: 0 = Enkripsi, 1 = Dekripsi self.mode\_var = tk.IntVar(value=0) # Frame Utama main\_frame = ttk.Frame(self.root, padding=10) main\_frame.pack(fill="both", expand=True) # Top: Toggle Mode mode\_frame = ttk.Frame(main\_frame) mode\_frame.pack(fill="x", pady=(0, 10)) ttk.Radiobutton(mode\_frame, text="Enkripsi", variable=self.mode\_var, value=0, command=self.switch\_mode).pack(side="left", padx=(0, 15)) ttk.Radiobutton(mode\_frame, text="Dekripsi", variable=self.mode\_var, value=1, command=self.switch\_mode).pack(side="left") # Content Frame (untuk form dinamis) self.content\_frame = ttk.Frame(main\_frame) self.content\_frame.pack(fill="both", expand=True, padx=5, pady=5) # Di bawahnya: Visualisasi Keystream vis\_frame = ttk.LabelFrame(main\_frame, text="Visualisasi Keystream", padding=5) vis\_frame.pack(fill="x", pady=(10, 0)) # Create tooltips for visualization buttons self.tooltips = {} vis\_buttons = [ ("Bar", self.show\_bar, "Tampilkan diagram batang nilai keystream"), ("Line", self.show\_line, "Tampilkan grafik perubahan nilai keystream"), ("Heatmap", self.show\_heatmap, "Tampilkan peta panas distribusi keystream"), ("Histogram", self.show\_histogram, "Tampilkan distribusi frekuensi nilai keystream"), ("Scatter", self.show\_scatter, "Tampilkan sebaran nilai keystream"), ("Statistik", self.show\_stats, "Tampilkan analisis statistik keystream") ] for i, (label, cmd, tip) in enumerate(vis\_buttons): btn = ttk.Button(vis\_frame, text=label, command=cmd) btn.grid(row=0, column=i, padx=5, pady=5) self.tooltips[btn] = tip btn.bind("<Enter>", self.show\_tooltip) btn.bind("<Leave>", self.hide\_tooltip) # Tooltip label self.tooltip\_label = ttk.Label(vis\_frame, text="", background="#ffffe0", relief="solid", borderwidth=1, wraplength=280) self.tooltip\_label.place\_forget() # Inisialisasi form pertama kali sebagai Enkripsi self.build\_encrypt\_form() def show\_tooltip(self, event): widget = event.widget tip = self.tooltips.get(widget, "") if tip: x = widget.winfo\_x() y = widget.winfo\_y() + widget.winfo\_height() + 5 self.tooltip\_label.config(text=tip) self.tooltip\_label.place(x=x, y=y, width=300) def hide\_tooltip(self, event): self.tooltip\_label.place\_forget() def clear\_content(self): for widget in self.content\_frame.winfo\_children(): widget.destroy() def switch\_mode(self): self.clear\_content() if self.mode\_var.get() == 0: self.build\_encrypt\_form() else: self.build\_decrypt\_form() def build\_encrypt\_form(self): frame = ttk.Frame(self.content\_frame, padding=10) frame.pack(fill="both", expand=True) # Konfigurasi grid frame.columnconfigure(1, weight=1) frame.columnconfigure(2, weight=0) # Row 0: Key ttk.Label(frame, text="Key:").grid(row=0, column=0, sticky="w", padx=5, pady=5) self.enc\_key\_entry = ttk.Entry(frame) self.enc\_key\_entry.grid(row=0, column=1, sticky="ew", padx=5, pady=5) # Generate Key button ttk.Button(frame, text="Generate Key", command=self.generate\_key).grid(row=0, column=2, padx=5, pady=5) # Row 1: Plaintext (multiline dengan scrollbar) ttk.Label(frame, text="Plaintext:").grid(row=1, column=0, sticky="nw", padx=5, pady=5) # Frame untuk text dan scrollbar text\_frame = ttk.Frame(frame) text\_frame.grid(row=1, column=1, columnspan=2, sticky="nsew", padx=5, pady=5) # Text widget dengan scrollbar self.enc\_plain\_text = tk.Text(text\_frame, height=4, wrap="word") self.enc\_plain\_scroll = ttk.Scrollbar(text\_frame, orient="vertical", command=self.enc\_plain\_text.yview) self.enc\_plain\_text.configure(yscrollcommand=self.enc\_plain\_scroll.set) self.enc\_plain\_text.pack(side="left", fill="both", expand=True) self.enc\_plain\_scroll.pack(side="right", fill="y") # Row 2: Tombol Enkripsi ttk.Button(frame, text="Enkripsi", command=self.encrypt).grid(row=2, column=0, columnspan=3, pady=10) # Row 3: Output Ciphertext ttk.Label(frame, text="Ciphertext (hex):").grid(row=3, column=0, sticky="nw", padx=5, pady=5) self.enc\_output = tk.Text(frame, height=4, wrap="word") self.enc\_output.grid(row=3, column=1, columnspan=2, sticky="ew", padx=5, pady=5) # Row 4: Output Keystream ttk.Label(frame, text="Keystream:").grid(row=4, column=0, sticky="nw", padx=5, pady=5) self.enc\_keystream\_output = tk.Text(frame, height=4, wrap="word") self.enc\_keystream\_output.grid(row=4, column=1, columnspan=2, sticky="ew", padx=5, pady=5) def build\_decrypt\_form(self): frame = ttk.Frame(self.content\_frame, padding=10) frame.pack(fill="both", expand=True) # Konfigurasi grid frame.columnconfigure(1, weight=1) # Row 0: Key ttk.Label(frame, text="Key:").grid(row=0, column=0, sticky="w", padx=5, pady=5) self.dec\_key\_entry = ttk.Entry(frame) self.dec\_key\_entry.grid(row=0, column=1, sticky="ew", padx=5, pady=5) # Row 1: Ciphertext (multiline dengan scrollbar) ttk.Label(frame, text="Ciphertext (hex):").grid(row=1, column=0, sticky="nw", padx=5, pady=5) # Frame untuk text dan scrollbar text\_frame = ttk.Frame(frame) text\_frame.grid(row=1, column=1, sticky="nsew", padx=5, pady=5) # Text widget dengan scrollbar self.dec\_cipher\_text = tk.Text(text\_frame, height=4, wrap="word") self.dec\_cipher\_scroll = ttk.Scrollbar(text\_frame, orient="vertical", command=self.dec\_cipher\_text.yview) self.dec\_cipher\_text.configure(yscrollcommand=self.dec\_cipher\_scroll.set) self.dec\_cipher\_text.pack(side="left", fill="both", expand=True) self.dec\_cipher\_scroll.pack(side="right", fill="y") # Row 2: Tombol Dekripsi ttk.Button(frame, text="Dekripsi", command=self.decrypt).grid(row=2, column=0, columnspan=2, pady=10) # Row 3: Output Plaintext ttk.Label(frame, text="Hasil Dekripsi:").grid(row=3, column=0, sticky="nw", padx=5, pady=5) self.dec\_output = tk.Text(frame, height=4, wrap="word") self.dec\_output.grid(row=3, column=1, sticky="ew", padx=5, pady=5) # Row 4: Output Keystream ttk.Label(frame, text="Keystream:").grid(row=4, column=0, sticky="nw", padx=5, pady=5) self.dec\_keystream\_output = tk.Text(frame, height=4, wrap="word") self.dec\_keystream\_output.grid(row=4, column=1, sticky="ew", padx=5, pady=5) def generate\_key(self): """Generate a random key dan masukkan ke entry key enkripsi saja""" if self.mode\_var.get() == 0: # Encryption mode characters = string.ascii\_letters + string.digits + string.punctuation key\_length = random.randint(12, 24) key = ''.join(random.SystemRandom().choice(characters) for \_ in range(key\_length)) self.enc\_key\_entry.delete(0, tk.END) self.enc\_key\_entry.insert(0, key) def encrypt(self): try: key\_str = self.enc\_key\_entry.get() plaintext = self.enc\_plain\_text.get("1.0", tk.END).strip() # Ambil dari Text widget if not key\_str.strip() or not plaintext: messagebox.showerror("Error", "Key dan plaintext harus diisi.") return # Konversi key ke bytes key\_bytes = [ord(c) for c in key\_str] self.keystream = rc4(key\_bytes, len(plaintext)) cipher\_bytes = encrypt\_rc4(plaintext, self.keystream) hex\_cipher = bytes\_to\_hex(cipher\_bytes) self.enc\_output.delete("1.0", tk.END) self.enc\_output.insert(tk.END, hex\_cipher) self.enc\_output.config(bg=self.success\_color) self.enc\_keystream\_output.delete("1.0", tk.END) self.enc\_keystream\_output.insert(tk.END, str(self.keystream)) except Exception as e: self.enc\_output.config(bg=self.error\_color) messagebox.showerror("Error", str(e)) def decrypt(self): try: key\_str = self.dec\_key\_entry.get() hex\_cipher = self.dec\_cipher\_text.get("1.0", tk.END).strip() # Ambil dari Text widget if not key\_str.strip() or not hex\_cipher: messagebox.showerror("Error", "Key dan ciphertext harus diisi.") return cipher\_bytes = hex\_to\_bytes(hex\_cipher) if cipher\_bytes is None: messagebox.showerror("Error", "Format ciphertext tidak valid. Harus hex.") return key\_bytes = [ord(c) for c in key\_str] self.keystream = rc4(key\_bytes, len(cipher\_bytes)) plaintext = decrypt\_rc4(cipher\_bytes, self.keystream) self.dec\_output.delete("1.0", tk.END) self.dec\_output.insert(tk.END, plaintext) self.dec\_output.config(bg=self.success\_color) self.dec\_keystream\_output.delete("1.0", tk.END) self.dec\_keystream\_output.insert(tk.END, str(self.keystream)) except Exception as e: self.dec\_output.config(bg=self.error\_color) messagebox.showerror("Error", str(e)) def plot\_figure(self, fig): window = tk.Toplevel(self.root) window.title("Visualisasi Keystream") window.geometry("800x400") canvas = FigureCanvasTkAgg(fig, master=window) canvas.draw() canvas.get\_tk\_widget().pack(fill=tk.BOTH, expand=True) # Add export button export\_btn = ttk.Button(window, text="Simpan Gambar", command=lambda: self.save\_plot(fig)) export\_btn.pack(pady=5) def save\_plot(self, fig): filename = tk.filedialog.asksaveasfilename( defaultextension=".png", filetypes=[("PNG files", ".png"), ("JPEG files", ".jpg"), ("All files", ".")] ) if filename: fig.savefig(filename) def show\_bar(self): if not hasattr(self, 'keystream'): messagebox.showinfo("Info", "Silakan enkripsi atau dekripsi terlebih dahulu.") return # Downsample if too large if len(self.keystream) > 1000: keystream\_sample = random.sample(self.keystream, 1000) else: keystream\_sample = self.keystream fig = plt.Figure(figsize=(8, 3)) ax = fig.add\_subplot(111) ax.bar(range(len(keystream\_sample)), keystream\_sample, color=self.primary\_color) ax.set\_title("Bar Chart Keystream") ax.set\_xlabel("Index Byte") ax.set\_ylabel("Nilai Keystream (0–255)") self.plot\_figure(fig) def show\_line(self): if not hasattr(self, 'keystream'): messagebox.showinfo("Info", "Silakan enkripsi atau dekripsi terlebih dahulu.") return # Downsample if too large if len(self.keystream) > 1000: keystream\_sample = random.sample(self.keystream, 1000) else: keystream\_sample = self.keystream fig = plt.Figure(figsize=(8, 3)) ax = fig.add\_subplot(111) ax.plot(keystream\_sample, color=self.primary\_color) ax.set\_title("Line Chart Keystream") ax.set\_xlabel("Index Byte") ax.set\_ylabel("Nilai Keystream (0–255)") self.plot\_figure(fig) def show\_heatmap(self): if not hasattr(self, 'keystream'): messagebox.showinfo("Info", "Silakan enkripsi atau dekripsi terlebih dahulu.") return fig = plt.Figure(figsize=(8, 1.5)) ax = fig.add\_subplot(111) sns.heatmap([self.keystream], ax=ax, cbar=True, cmap="viridis") ax.set\_title("Heatmap Keystream") ax.set\_yticks([]) ax.set\_xlabel("Index Byte") self.plot\_figure(fig) def show\_histogram(self): if not hasattr(self, 'keystream'): messagebox.showinfo("Info", "Silakan enkripsi atau dekripsi terlebih dahulu.") return fig = plt.Figure(figsize=(8, 3)) ax = fig.add\_subplot(111) ax.hist(self.keystream, bins=256, range=(0, 256), edgecolor='black', color=self.primary\_color) ax.set\_title("Histogram Keystream") ax.set\_xlabel("Nilai Keystream") ax.set\_ylabel("Frekuensi") self.plot\_figure(fig) def show\_scatter(self): if not hasattr(self, 'keystream'): messagebox.showinfo("Info", "Silakan enkripsi atau dekripsi terlebih dahulu.") return # Downsample if too large if len(self.keystream) > 1000: keystream\_sample = random.sample(self.keystream, 1000) indices = random.sample(range(len(self.keystream)), 1000) else: keystream\_sample = self.keystream indices = range(len(self.keystream)) fig = plt.Figure(figsize=(8, 3)) ax = fig.add\_subplot(111) ax.scatter(indices, keystream\_sample, s=10, color=self.primary\_color) ax.set\_title("Scatter Plot Keystream") ax.set\_xlabel("Index Byte") ax.set\_ylabel("Nilai Keystream (0–255)") self.plot\_figure(fig) def show\_stats(self): if not hasattr(self, 'keystream'): messagebox.showinfo("Info", "Silakan enkripsi atau dekripsi terlebih dahulu.") return # Create stats window stats\_win = tk.Toplevel(self.root) stats\_win.title("Statistik Keystream") stats\_win.geometry("400x300") # Create main frame frame = ttk.Frame(stats\_win, padding=10) frame.pack(fill="both", expand=True) # Calculate statistics length = len(self.keystream) unique = len(set(self.keystream)) min\_val = min(self.keystream) max\_val = max(self.keystream) avg = sum(self.keystream) / length # Calculate standard deviation variance = sum((x - avg) \*\* 2 for x in self.keystream) / length std\_dev = math.sqrt(variance) # Create stats display with Treeview ttk.Label(frame, text="Statistik Keystream", font=("Arial", 12, "bold")).grid(row=0, column=0, columnspan=2, pady=(0, 10)) # Create Treeview for tabular display columns = ("metric", "value") tree = ttk.Treeview(frame, columns=columns, show="headings", height=6) # Configure columns tree.heading("metric", text="Metrik") tree.heading("value", text="Nilai") tree.column("metric", width=150, anchor="w") tree.column("value", width=150, anchor="w") # Add data stats = [ ("Panjang Keystream", f"{length}"), ("Nilai Minimum", f"{min\_val}"), ("Nilai Maksimum", f"{max\_val}"), ("Rata-rata", f"{avg:.4f}"), ("Standar Deviasi", f"{std\_dev:.4f}"), ("Nilai Unik", f"{unique} ({unique/256\*100:.1f}%)") ] for stat in stats: tree.insert("", "end", values=stat) tree.grid(row=1, column=0, columnspan=2, sticky="nsew", pady=(0, 10)) # Add scrollbar scrollbar = ttk.Scrollbar(frame, orient="vertical", command=tree.yview) tree.configure(yscrollcommand=scrollbar.set) scrollbar.grid(row=1, column=2, sticky="ns") # Configure grid weights frame.rowconfigure(1, weight=1) frame.columnconfigure(0, weight=1) # Add export button ttk.Button(frame, text="Export ke CSV", command=self.export\_stats).grid(row=2, column=0, columnspan=2, pady=(5, 0)) def export\_stats(self): filename = tk.filedialog.asksaveasfilename( defaultextension=".csv", filetypes=[("CSV files", ".csv"), ("All files", ".\*")] ) if filename: try: with open(filename, 'w') as f: f.write("Statistik,Nilai\n") f.write(f"Panjang Keystream,{len(self.keystream)}\n") f.write(f"Nilai Minimum,{min(self.keystream)}\n") f.write(f"Nilai Maksimum,{max(self.keystream)}\n") avg = sum(self.keystream) / len(self.keystream) f.write(f"Rata-rata,{avg:.4f}\n") variance = sum((x - avg) \*\* 2 for x in self.keystream) / len(self.keystream) std\_dev = math.sqrt(variance) f.write(f"Standar Deviasi,{std\_dev:.4f}\n") unique = len(set(self.keystream)) f.write(f"Nilai Unik,{unique}\n") f.write(f"Persentase Unik,{unique/256\*100:.2f}%\n") except Exception as e: messagebox.showerror("Error", f"Gagal menyimpan file: {str(e)}")# === Jalankan Aplikasi ===if \_\_name\_\_ == "\_\_main\_\_": root = tk.Tk() app = RC4App(root) root.mainloop()

Hasil Implementasi :

