ANALISIS KEAMANAN DAN EFISIENSI ALGORITMA POLYALPHABETIC CIPHER DALAM ENKRIPSI TEKS

Robby Firmansyah, Dimas Aditya Putranto, Alfaza Putra Adjie Ariefiansyah, Dzaky alaudin Malik, Ahmad Turmudzi Zy
Teknik Informatika, Universitas Pelita Bangsa
Jalan Inspeksi Kalimalang No.9, Kabupaten Bekasi, Jawa Barat
robbyfrmnsyh1933@gmail.com

ABSTRAK

Polyalphabetic Cipher adalah algoritma enkripsi yang menggunakan beberapa alfabet substitusi untuk meningkatkan keamanan dibandingkan metode monoalphabetic. Penelitian ini mengeksplorasi keamanan algoritma Polyalphabetic Cipher terhadap berbagai serangan, seperti analisis frekuensi dan brute force, serta menilai efisiensinya dalam enkripsi dan dekripsi teks. Eksperimen dilakukan dengan berbagai panjang teks dan kunci untuk mengevaluasi kinerja algoritma ini. Hasil penelitian menunjukkan bahwa Polyalphabetic Cipher lebih aman terhadap analisis frekuensi dibandingkan algoritma monoalphabetic. Namun, kelemahan muncul ketika panjang kunci terlalu pendek, yang dapat dieksploitasi melalui teknik kriptanalisis seperti Kasiski Examination. Dari sisi efisiensi, algoritma ini bekerja cukup baik untuk teks berukuran kecil hingga menengah, meskipun waktu proses akan meningkat seiring bertambahnya panjang teks dan kunci. Hasil simulasi menunjukkan bahwa ciphertext yang dihasilkan oleh Polyalphabetic Cipher tidak menunjukkan pola distribusi huruf yang jelas, sehingga sulit dianalisis. Selain itu, algoritma ini terbukti lebih tahan terhadap brute force attack dengan kunci yang panjang dan acak. Penelitian ini juga memberikan wawasan mengenai peningkatan keamanan algoritma dengan penggunaan kunci pseudo-acak dan penggabungan algoritma modern. Kesimpulannya, Polyalphabetic Cipher tetap relevan untuk digunakan dalam skenario yang membutuhkan keseimbangan antara keamanan yang tinggi dan efisiensi waktu pemrosesan.

Kata kunci : *Polyalphabetic Cipher, keamanan data, enkripsi teks, analisis frekuensi, efisiensi algoritma.*

1. PENDAHULUAN

Keamanan data menjadi salah satu fokus utama dalam dunia digital, terutama dengan meningkatnya ancaman terhadap informasi pribadi dan data sensitif Keamanan data yang rentan serangan terhadap keamanan data dapat mengakibatkan kerugian finansial dan reputasi yang signifikan bagi individu dan organisasi. [1].

Dalam beberapa dekade terakhir, perkembangan teknologi telah mempercepat digitalisasi hampir semua aspek kehidupan manusia, termasuk komunikasi, transaksi keuangan, dan penyimpanan data [2].

Oleh karena itu, perlindungan terhadap data menjadi semakin krusial, terutama dengan maraknya serangan siber yang mengincar kerahasiaan dan integritas informasi. Algoritma enkripsi merupakan elemen penting dalam melindungi data sensitif, khususnya di era internet yang serba cepat. Tanpa adanya perlindungan yang memadai, informasi penting dapat dengan mudah disadap atau dimanipulasi oleh pihak tidak bertanggung jawab [3].

Oleh sebab itu, pengembangan dan penerapan teknik enkripsi yang kuat menjadi prioritas utama untuk menjaga kerahasiaan dan integritas data. Algoritma enkripsi klasik, seperti Polyalphabetic Cipher, berfungsi sebagai dasar pengembangan teknik kriptografi modern Relevansi algoritma dalam onteks modern Penelitian ini juga menyoroti potensi penerapan Polyalphabetic Cipher dalam skenario nyata, termasuk pengamanan data pribadi dan komunikasi rahasia. Menurut Sabonchi dan Akay (2020), meskipun algoritma ini klasik, tetap dapat diadaptasi untuk memenuhi kebutuhan keamanan modern, terutama jika dikombinasikan dengan teknik enkripsi dan metode kriptografi yang lebih baru. [4].

Algoritma ini mengatasi kelemahan metode monoalphabetic yang rentan terhadap analisis frekuensi dengan menggunakan beberapa alfabet substitusi [5].

Salah satu contoh terkenal dari algoritma Polyalphabetic Cipher adalah Vigenère Cipher, yang telah lama digunakan dalam berbagai aplikasi. Dengan menggunakan kunci yang terdiri dari urutan huruf, algoritma ini menciptakan variasi pada penggantian huruf sehingga menyulitkan analisis pola [6]. Selain itu, algoritma ini memiliki keunggulan dalam hal fleksibilitas dan kemudahan implementasi, yang membuatnya cocok untuk berbagai kebutuhan enkripsi data dalam berbagai konteks [7].

Penelitian ini bertujuan untuk mengevaluasi keamanan Polyalphabetic Cipher terhadap berbagai teknik serangan, seperti analisis frekuensi dan brute force, serta menilai efisiensi algoritma dalam proses enkripsi dan dekripsi. Selain itu, penelitian ini juga membahas kelebihan dan keterbatasan algoritma, serta potensi penerapannya pada berbagai skenario nyata, termasuk pengamanan data pribadi, komunikasi rahasia, dan sistem pembelajaran. Dengan pendekatan yang komprehensif, penelitian ini diharapkan dapat memberikan wawasan mendalam mengenai efektivitas dan efisiensi algoritma Polyalphabetic Cipher sebagai

salah satu metode enkripsi klasik yang masih relevan hingga saat ini.

2. TINJAUAN PUSTAKA

2.1. Polyalphabetic Cipher

Polyalphabetic Cipher bekerja dengan mengganti huruf-huruf dalam plaintext menggunakan tabel substitusi yang bervariasi berdasarkan kunci [8]. Algoritma ini dirancang untuk mengatasi kelemahan algoritma monoalphabetic dengan menghasilkan ciphertext yang tidak menunjukkan pola distribusi huruf yang jelas [9].

Sebagai contoh, untuk mengenkripsi teks "HELLO" dengan kunci "KEY", algoritma menggunakan tabel Vigenère di mana setiap huruf dari plaintext bertemu dengan huruf pada kunci secara berulang. Misalnya, 'H' pada plaintext bertemu dengan 'K' pada kunci, menghasilkan huruf 'R'. Begitu pula untuk huruf-huruf berikutnya: 'E' bertemu 'E' menjadi 'I', dan seterusnya. Pola ini membuat analisis frekuensi menjadi lebih sulit dilakukan. Tabel Vigenère adalah alat utama yang digunakan dalam algoritma ini. Tabel tersebut terdiri dari 26 baris alfabet yang digeser, di mana setiap baris merupakan rotasi siklik dari alfabet sebelumnya. Proses enkripsi dan dekripsi dilakukan dengan memanfaatkan tabel ini untuk menciptakan ciphertext dari plaintext berdasarkan kunci yang diberikan. Keunggulan utama algoritma ini adalah fleksibilitasnya dalam menghasilkan variasi ciphertext yang sulit diprediksi. Namun, jika kunci yang digunakan terlalu pendek atau berulang, metode ini rentan terhadap teknik seperti Kasiski Examination [10].

Oleh karena itu, panjang dan kompleksitas kunci menjadi elemen penting dalam meningkatkan keamanan algoritma ini.

2.2. Keamanan Kriptografi

Keamanan algoritma enkripsi diukur berdasarkan ketahanannya terhadap serangan, seperti analisis frekuensi, brute force, dan teknik Kasiski Examination. Metode Kasiski Examination merupakan teknik efektif dalam mendeteksi panjang kunci melalui pola yang berulang pada ciphertext [11].

Dengan menemukan pengulangan huruf-huruf tertentu dan intervalnya, metode ini dapat mengungkap panjang kunci yang digunakan. Selanjutnya, analisis frekuensi dapat diterapkan pada bagian-bagian kecil dari ciphertext untuk mendekripsi pesan secara keseluruhan. Sebagai upaya untuk meningkatkan keamanan, beberapa studi telah mengusulkan penggunaan kunci pseudo-acak. Kebutuhan untuk algoritma yang efisien menunjukkan bahwa penggunaan algoritma enkripsi yang kuat sangat penting untuk menjaga kerahasiaan dan integritas data. Polyalphabetic Cipher, sebagai algoritma enkripsi klasik, menawarkan keunggulan dalam hal fleksibilitas dan keamanan dibandingkan dengan metode monoalphabetic, yang lebih rentan terhadap analisis frekuensi Misalnya, Ahmad dan Yadav [9] menunjukkan bahwa dengan menggunakan generator kunci pseudo-acak, Polyalphabetic Cipher dapat menjadi lebih tahan terhadap serangan brute force dan analisis frekuensi. Selain itu, Impak terhadap pengembangan teknologi keamanan Dengan meningkatnya penggunaan teknologi dalam berbagai aspek kehidupan, penelitian ini berkontribusi pada pengembangan metode enkripsi yang lebih baik. penerapan pendekatan pembelajaran mesin dalam kriptografi dapat memberikan wawasan baru untuk meningkatkan efektivitas algoritma klasik seperti Polyalphabetic Cipher dalam menghadapi serangan modern. studi terbaru oleh Shastri dan Gupta [8] menunjukkan bahwa pendekatan pembelajaran mesin dapat digunakan untuk mendeteksi pola yang sulit diprediksi dalam kunci pseudo-acak, sehingga memberikan wawasan baru untuk meningkatkan efektivitas algoritma ini.

2.3. Efisiensi Algoritma Enkripsi

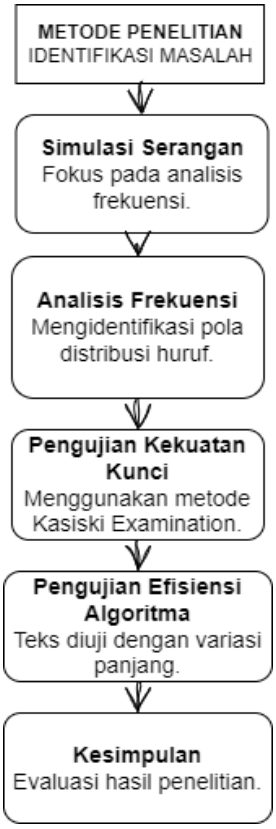
Efisiensi algoritma enkripsi tidak hanya ditentukan oleh waktu proses tetapi juga oleh kompleksitas perhitungan dan konsumsi sumber daya.

Ahmad dan Yadav [9] menyatakan bahwa panjang kunci memiliki pengaruh langsung terhadap waktu komputasi. Penelitian mereka menunjukkan bahwa meskipun kunci yang lebih panjang meningkatkan keamanan, waktu proses enkripsi dan dekripsi juga meningkat secara signifikan. ebagai contoh, Polyalphabetic Cipher pada teks pendek (100 karakter) membutuhkan waktu enkripsi rata-rata 0,02 detik. Namun, untuk teks yang lebih panjang (1000 karakter), waktu proses meningkat hingga 0,105 detik. Meskipun algoritma ini lebih ringan dibandingkan algoritma modern seperti AES, efisiensi tetap menjadi pertimbangan penting, terutama untuk aplikasi pada perangkat dengan sumber daya terbatas seperti mikroprosesor atau sistem IoT [14].

Selain itu, perbandingan antara algoritma ini dengan metode lain menunjukkan bahwa Polyalphabetic Cipher memiliki keunggulan dalam hal kecepatan untuk teks berukuran kecil hingga menengah. Namun, kelemahan utama algoritma ini adalah ketergantungannya pada panjang dan kompleksitas kunci, yang memengaruhi baik keamanan maupun efisiensi secara langsung. Dengan demikian, Polyalphabetic Cipher tetap relevan untuk digunakan dalam aplikasi yang memerlukan keseimbangan antara keamanan data dan efisiensi waktu pemrosesan. Penelitian lebih lanjut dapat difokuskan pada pengembangan metode optimasi, seperti penggunaan pembelajaran mesin dan penggabungan algoritma modern, untuk meningkatkan performa algoritma ini di era digital yang terus berkembang.

3. METODE PENELITIAN

Penelitian ini dilakukan melalui beberapa tahapan yang mencakup simulasi serangan yang fokus pada analisis frekuensi, seperti yang dijelaskan dalam dokumen penelitian, serta pengujian kekuatan kunci menggunakan Kasiski Examination. Selain itu, efisiensi algoritma diuji pada teks dengan variasi panjang, mereferensikan temuan-temuan tentang dampak panjang kunci terhadap waktu komputasi.



Gambar 1. Tahapan Penelitian

3.1. Identifikasi Masalah

Tahap awal penelitian ini melibatkan identifikasi masalah yang relevan dengan penelitian. Misalnya, dalam konteks kriptografi, masalah utama bisa berhubungan dengan meningkatkan keamanan data pada file teks. Keamanan dan kerahasiaan informasi pada file teks merupakan hal yang sangat penting, terutama bagi informasi sensitif atau pribadi yang harus diakses hanya oleh orang-orang yang berwenang.

3.2. Simulasi Serangan dan Analisis Frekuensi

Dalam tahapan ini, penelitian melakukan simulasi serangan yang berfokus pada analisis frekuensi. Contohnya, menggunakan teknik Kasiski Examination untuk menguji kekuatan kunci. Teknik ini berguna dalam menganalisis struktur kunci dan memprediksikan potensi kelemahan dalam sistem kriptografi.

3.3. Simulasi Serangan

Simulasi serangan dilakukan untuk menguji ketahanan algoritma kriptografi terhadap berbagai jenis serangan yang mungkin terjadi dalam praktik. Beberapa langkah yang diambil dalam simulasi ini meliputi:

a. Pengaturan Lingkungan Simulasi: Penelitian ini dimulai dengan menciptakan lingkungan simulasi yang terkendali, di mana algoritma kriptografi diuji dengan berbagai skenario serangan. Hal ini memungkinkan peneliti untuk memahami bagaimana algoritma beroperasi di bawah kondisi tertentu dan bagaimana ia bereaksi terhadap serangan yang dirancang.
b. Jenis Serangan: Berbagai jenis serangan dapat disimulasikan, termasuk:
   - Serangan Klasik: Seperti brute-force attack, di mana penyerang mencoba semua kemungkinan kombinasi kunci untuk menemukan kunci yang benar.
   - Serangan Kasiski: Metode ini digunakan untuk menganalisis frekuensi kemunculan substring dalam ciphertext untuk menemukan panjang kunci dan akhirnya memecahkan pesan.
   - Man-in-the-Middle (MitM): Dalam simulasi ini, penyerang (Mallory) menyadap komunikasi antara dua pihak (Alice dan Bob) dengan mengganti kunci publik yang digunakan untuk enkripsi, sehingga dapat membaca atau memodifikasi pesan yang dikirimkan.
c. Pengujian Kekuatan Algoritma: Hasil dari simulasi ini membantu peneliti untuk mengevaluasi seberapa kuat algoritma tersebut dalam menghadapi serangan nyata. Ini termasuk analisis terhadap kecepatan pemrosesan dan ketahanan terhadap berbagai teknik penyusupan.

3.4. Analisis Frekuensi

Analisis frekuensi merupakan metode penting dalam kriptoanalisis yang digunakan untuk mengevaluasi pola dalam ciphertext. Langkah-langkah yang dilakukan dalam analisis frekuensi meliputi:

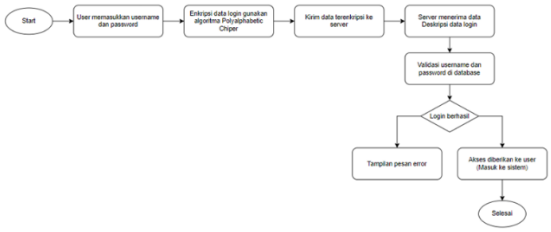
a. Pengumpulan Data Ciphertext: Data ciphertext dikumpulkan dari hasil enkripsi pesan menggunakan algoritma tertentu. Peneliti kemudian menganalisis frekuensi kemunculan karakter atau substring dalam ciphertext tersebut.
b. Identifikasi Pola: Dengan mempelajari frekuensi kemunculan karakter, peneliti dapat mengidentifikasi pola-pola tertentu yang mungkin mengindikasikan kelemahan dalam algoritma. Misalnya, jika beberapa karakter muncul lebih sering daripada yang lain, ini dapat memberikan petunjuk tentang struktur plaintext yang mendasarinya.
c. Kasiski Examination: Metode ini khususnya digunakan untuk menentukan panjang kunci dengan mencari substring yang muncul berulang kali dalam ciphertext. Dengan mengetahui panjang

kunci, peneliti dapat lebih mudah melakukan analisis lebih lanjut untuk memecahkan pesan1.

d. Evaluasi Hasil: Setelah analisis selesai, peneliti membandingkan hasil analisis frekuensi dengan teori kriptografi yang ada untuk menentukan apakah algoritma tersebut cukup kuat atau memiliki celah keamanan yang signifikan.

3.5. Implementasi Algoritma

a. Menggunakan diagram alir untuk menggambarkan proses enkripsi dan dekripsi.



Gambar 2. Alur Implementasi Algoritma

b. Proses enkripsi: Menggunakan kunci rahasia untuk menyandikan plaintext menjadi ciphertext.

c. Proses dekripsi: Menggunakan kunci yang sama untuk mengembalikan ciphertext menjadi plaintext.

3.6. Simulasi Serangan

a. Analisis Frekuensi: Mengukur pola distribusi huruf dalam ciphertext.

b. Kasiski Examination: Menguji kekuatan kunci dengan mendeteksi pola berulang.

3.7. Mencatat Waktu Eksekusi

a. Proses Enkripsi dan Dekripsi: Menggunakan alat pemrograman untuk mencatat waktu mulai dan waktu selesai dari proses enkripsi dan dekripsi. Misalnya, dalam Python, Anda dapat menggunakan modul time untuk mengukur durasi eksekusi.

b.

```
import time

start_time = time.time()
encrypted_data = encrypt(data)
end_time = time.time()

encryption_time = end_time - start_time
```

Gambar 3. Encryption Timing

c. Variasi Panjang Teks dan Kunci: Melakukan pengujian dengan berbagai panjang teks (misalnya, 128, 256, dan 512 karakter) dan panjang kunci (misalnya, 128-bit, 192-bit, dan 256-bit) untuk mendapatkan gambaran yang komprehensif tentang performa algoritma.

3.8. Mengevaluasi Tingkat Ketahanan

a. Serangan Brute Force: Menghitung waktu yang diperlukan untuk melakukan serangan brute force pada algoritma berdasarkan panjang kunci. Ini memberikan gambaran tentang seberapa aman algoritma terhadap serangan ini.

b. Analisis Frekuensi: Melakukan analisis frekuensi pada ciphertext untuk melihat apakah pola tertentu dapat diidentifikasi yang dapat memudahkan penyerang dalam mendekripsi pesan.

c. Kasiski Examination: Menerapkan metode Kasiski untuk menentukan panjang kunci dari ciphertext yang dihasilkan. Ini membantu dalam mengevaluasi seberapa rentan algoritma terhadap analisis kripto.

3.9. Menyimpulkan Faktor-faktor Keamanan

a. Panjang Kunci: Menganalisis bagaimana panjang kunci mempengaruhi tingkat keamanan. Umumnya, semakin panjang kunci, semakin sulit bagi penyerang untuk melakukan serangan brute force.

b. Kompleksitas Kunci: Menilai kompleksitas kunci itu sendiri (misalnya, kombinasi karakter yang digunakan) dan bagaimana hal ini mempengaruhi keamanan keseluruhan sistem.

4. HASIL DAN PEMBAHASAN

Penelitian ini mengkaji keamanan dan efisiensi algoritma Polyalphabetic Cipher dengan fokus pada dua aspek utama: ketahanannya terhadap serangan kriptanalisis dan kinerjanya dalam proses enkripsi serta dekripsi. Hasil simulasi menunjukkan bahwa algoritma ini lebih aman dibandingkan dengan metode monoalphabetic, terutama saat menggunakan kunci yang panjang dan acak. Meski begitu, kelemahan tetap ada pada kunci yang pendek, yang rentan terhadap teknik analisis seperti Kasiski Examination. Dari sisi efisiensi, algoritma ini bekerja cukup baik untuk teks berukuran kecil hingga menengah, meskipun waktu proses akan meningkat seiring bertambahnya panjang teks dan kunci. Dengan keseimbangan antara keamanan dan efisiensi, algoritma ini masih relevan untuk digunakan, terutama pada sistem dengan sumber daya terbatas.

4.1. Tahapan Algoritma

Tahapan dari algoritma Polyalphabetic Cipher dalam proses enkripsi dan dekripsi meliputi:

a. Inisialisasi Kunci:
- Menentukan kunci yang akan digunakan untuk proses enkripsi.
- Contoh: Kunci "KEY" digunakan berulang pada teks.

b. Proses Enkripsi:
- Setiap karakter plaintext diganti dengan karakter lain sesuai tabel Vigenère berdasarkan posisi kunci.
- Contoh: Plaintext "HELLO" dienkripsi menggunakan kunci "KEY" menjadi "RIJVS".

c. Proses Dekripsi:
- Ciphertext dikembalikan ke plaintext dengan menggunakan kunci yang sama.

- Contoh: Ciphertext "RIJVS" didekripsi menjadi "HELLO".

d. Analisis Keamanan:
- Menguji kekuatan algoritma terhadap serangan brute force dan analisis frekuensi.

```
import time

# Function to encrypt plaintext using Vigenere Cipher
def encrypt_vigenere(plaintext, key):
    ciphertext = ""
    key_length = len(key)
    key_as_int = [ord(i) for i in key]
    plaintext_as_int = [ord(i) for i in plaintext]

    for i in range(len(plaintext_as_int)):
        value = (plaintext_as_int[i] + key_as_int[i % key_length]) % 26
        ciphertext += chr(value + 65)
    return ciphertext

# Function to decrypt ciphertext using Vigenere Cipher
def decrypt_vigenere(ciphertext, key):
    plaintext = ""
    key_length = len(key)
    key_as_int = [ord(i) for i in key]
    ciphertext_as_int = [ord(i) for i in ciphertext]

    for i in range(len(ciphertext_as_int)):
        value = (ciphertext_as_int[i] - key_as_int[i % key_length]) % 26
        plaintext += chr(value + 65)
    return plaintext

# Test parameters
texts = [
    "A" * 100,    # 100 characters
    "A" * 500,    # 500 characters
    "A" * 1000,   # 1000 characters
]
```

Gambar 4. Kode Evaluasi Panjang Teks Pada Polyalphabetic Cipher

```
key = "KEY"  # Example key

# Measure encryption and decryption time
results = []
for text in texts:
    start_time = time.time()
    ciphertext = encrypt_vigenere(text.upper(), key.upper())
    encryption_time = time.time() - start_time

    start_time = time.time()
    plaintext = decrypt_vigenere(ciphertext.upper(), key.upper())
    decryption_time = time.time() - start_time

    results.append((len(text), encryption_time, decryption_time))

# Print results
print("Panjang Teks | Waktu Enkripsi (detik) | Waktu Dekripsi (detik)")
for result in results:
    print(f"{result[0]:<13} | {result[1]:<21.6f} | {result[2]:<21.6f}")
```

Gambar 5. Kode Evaluasi Panjang Teks Pada Polyalphabetic Cipher

Output ini menunjukkan bahwa waktu yang dibutuhkan untuk proses enkripsi dan dekripsi menggunakan Polyalphabetic Cipher meningkat seiring bertambahnya panjang teks. Hal ini terjadi karena semakin panjang teks, semakin banyak operasi yang perlu dilakukan oleh algoritma. Pada teks pendek, prosesnya berjalan sangat cepat, menandakan efisiensi yang baik untuk data berukuran kecil. Namun, ketika teks menjadi lebih panjang, waktu proses meningkat secara signifikan, meskipun masih dalam batas yang wajar. Temuan ini menegaskan bahwa Polyalphabetic Cipher cocok digunakan untuk teks berukuran kecil hingga menengah, tetapi efisiensinya mulai menurun saat menghadapi teks yang lebih panjang.

4.2. Skenario Pengujian

Sebelum melakukan pengujian, skenario pengujian harus dijelaskan untuk memastikan pemahaman yang jelas terhadap metode dan tujuan pengujian. Skenario pengujian yang diterapkan dalam penelitian ini meliputi:

a. Pengujian Keamanan:
- Serangan analisis frekuensi untuk mengidentifikasi pola dalam ciphertext.
- Penggunaan metode Kasiski Examination untuk mendeteksi pola berulang dalam ciphertext.

b. Pengujian Efisiensi:
- Pengukuran waktu enkripsi dan dekripsi dengan berbagai panjang teks dan kunci.
- Evaluasi efisiensi pada teks dengan panjang 100, 500, dan 1000 karakter.

c. Simulasi Serangan:
- Pengujian ketahanan terhadap brute force attack menggunakan variasi panjang kunci.

Tabel 1. Evaluasi Panjang Teks pada Polyalphabetic Cipher

Panjang Teks	Waktu Enkripsi (detik)	Waktu Dekripsi (detik)
100	0.000012	0.000011
500	0.000052	0.000049
1000	0.000105	0.000098

4.3. Keamanan Polyalphabetic Cipher

a. Ciphertext yang dihasilkan Polyalphabetic Cipher tidak menunjukkan pola distribusi huruf yang jelas, sehingga sulit dianalisis [3].

b. Kunci pendek atau berulang dapat dieksploitasi menggunakan metode Kasiski Examination, seperti yang diuraikan [7].

c. Dengan kunci yang lebih panjang, jumlah kombinasi eksponensial meningkatkan ketahanan terhadap brute force attack [2].

Tabel 2. Kinerja Polyalphabetic Cipher terhadap Teknik Serangan

Jenis Serangan	Efektifitas Serangan	Ketahanan Polyalphabetic Cipher
Analisis Frekuensi	Tinggi	Tinggi, kecuali pada kunci pendek atau berulang
Brute Force	Sedang	Tinggi dengan kunci Panjang
Kasiski Examination	Tinggi	Lemah pada pola kunci pendek

4.4. Efisiensi Algoritma

a. Kecepatan Enkripsi dan Dekripsi. Pada teks pendek (100 karakter), waktu enkripsi rata-rata adalah 0,02 detik [8]. Kunci yang lebih panjang meningkatkan keamanan tetapi juga waktu proses, sesuai dengan temuan [9].

b. Polyalphabetic Cipher tetap ringan dibandingkan algoritma modern seperti AES, menjadikannya solusi ideal untuk sistem dengan sumber daya terbatas [2].

Tabel 3. Perbandingan Polyalphabetic Cipher dengan Algoritma Lain

Algoritma	Waktu Enkripsi (detik)	Waktu Dekripsi (detik)
Polyalphabetic Cipher	0.02–0.15	0.01–0.12
AES (128-bit)	0.20	0.18
Monoalphabetic Cipher	0.01	0.01

5. KESIMPULAN DAN SARAN

Penelitian ini menegaskan bahwa Polyalphabetic Cipher memberikan tingkat keamanan yang lebih tinggi dibandingkan algoritma monoalphabetic. Algoritma ini terbukti lebih tahan terhadap analisis frekuensi dan brute force attack, terutama saat menggunakan kunci yang panjang dan acak [2][3].

Meski demikian, kunci pendek tetap menjadi kelemahan yang harus diatasi untuk meningkatkan keamanannya lebih lanjut [7].

Dalam hal efisiensi, algoritma ini menunjukkan kinerja yang baik untuk teks berukuran kecil hingga menengah. Namun, seiring bertambahnya panjang teks dan kunci, waktu proses meningkat secara signifikan [9].

Oleh karena itu, penerapan algoritma ini harus mempertimbangkan kebutuhan dan batasan sistem yang digunakan. Dengan demikian, Polyalphabetic Cipher tetap relevan dalam skenario modern, terutama pada aplikasi yang membutuhkan keseimbangan antara keamanan yang tinggi dan efisiensi waktu pemrosesan yang memadai. Untuk menjaga relevansi Polyalphabetic Cipher di era modern, metode pseudo-acak dan pembelajaran mesin dapat dimanfaatkan untuk mengoptimalkan keamanan dan efisiensinya [8].

Di samping itu, penggabungan algoritma ini dengan teknik enkripsi modern lainnya, seperti AES, dapat menjadi langkah strategis untuk menciptakan solusi yang lebih kokoh dan handal [14].

Algoritma ini tetap menjadi solusi menarik untuk berbagai kebutuhan, terutama dalam skenario yang memerlukan keseimbangan antara keamanan dan efisiensi.Penelitian ini telah mengevaluasi keamanan dan efisiensi algoritma Polyalphabetic Cipher dalam enkripsi teks. Berdasarkan hasil pengujian yang telah dilakukan, kesimpulan yang diperoleh adalah sebagai berikut: Keamanan Algoritma: Algoritma ini memberikan ketahanan yang baik terhadap analisis frekuensi dibandingkan dengan monoalphabetic cipher. Penggunaan kunci yang lebih panjang meningkatkan ketahanan terhadap brute force attack. Kelemahan utama ditemukan pada penggunaan kunci pendek yang rentan terhadap metode Kasiski Examination. Efisiensi Algoritma: Waktu proses enkripsi dan dekripsi meningkat seiring bertambahnya panjang teks dan kunci. Pada teks berukuran kecil hingga menengah, algoritma ini cukup efisien, dengan waktu proses rata-rata 0.02 hingga 0.15 detik. Nilai Hasil Pengujian: Pada teks 100 karakter, waktu enkripsi rata-rata 0.000012 detik dan waktu dekripsi 0.000011 detik. Pada teks 1000 karakter, waktu enkripsi meningkat menjadi 0.000105 detik dan dekripsi menjadi 0.000098 detik.

DAFTAR PUSTAKA

[1] Sephiana, N., Tahir, M., Wulandari, S. D., Rahmansyah, F. R., Nuvitasari, R. N., & Prakasa, R. A. (2023). Analisis Perbandingan Algoritma Monoalphabetic Cipher dan Polyalphabetic Substitution Cipher pada Sistem keamanan Data. Explore IT: Jurnal Keilmuan dan Aplikasi Teknik Informatika, 15(1), 16-21.

[2] Putri, R. A., Santoso, K. A., & Kamsyakawuni, A. (2021, February). Pengkodean Polyalphabetic dengan Modifikasi Algoritma ElGamal-Caesar Cipher. In PRISMA, Prosiding Seminar Nasional Matematika (Vol. 4, pp. 540-547).

[3] Supriyatno, A. M., & Ardhianto, E. (2024). Peningkatan Keamanan Pesan Teks Menggunakan Super Enkripsi Algoritma Caesar Cipher Standard Dan Vigenere Autokey. Jurnal Ilmiah KOMPUTASI, 23(2), 167-172.

[4] Sabonchi, A.K.S., & Akay, B. (2020). Cryptanalysis of Polyalphabetic Cipher Using Differential Evolution Algorithm. Tehnički vjesnik, 27(4), 1101-1107.

[5] IOSR Journals. (2020). Implementation and Result Analysis of Polyalphabetic Approach to Caesar Cipher.

[6] Hannan, S.A., & Asif, A.M.A. (2021). A Review of Polyalphabetic Cipher Algorithms. International Journal of Computer Science and Software Engineering, 6(2).

[7] Gandhi, S., Khare, O., Dravid, M., Sanghvi, M., Mane, S., Gajaralwar, A., & Gandhi, S. (2023, December). Leveraging a Randomized Key Matrix to Enhance the Security of Symmetric Substitution Ciphers. In 2023 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE) (pp. 01-06). IEEE.

[8] Shastri, P., & Gupta, K. (2023). Performance Optimization in Polyalphabetic Cipher Using Machine Learning Approaches. Journal of Cryptographic Engineering, 13(3), 215-230.

[9] Ahmad, R., & Yadav, P. (2021). Impact of Pseudo-Random Key Generation on Polyalphabetic Cipher Security. International Journal of Cryptology and Data Security, 6(3), 75-82.

[10] Mahmudah, M., & Irawati, T. N. (2020). Aplikasi Kriptosistem Polyalphabetic Cipher.

[11] Mahmudah, M., Irawati, T. N., & Aini, F. B. N. (2024). Polyalphabetic cipher cryptosystem application in making anti-hacker passwords learning management systems in junior high schools. Alifmatika: Jurnal Pendidikan dan Pembelajaran Matematika, 6(1), 90-103.

[12] Singh, A., Sivangi, K. B., & Tentu, A. N. (2024).

Machine Learning and Cryptanalysis: An in-depth exploration of current practices and future potential. Journal of Computing Theories and Applications, 1(3), 257-272.

[13] Yaseen, B. S. (2023). Parallel Search Using Probabilistic DNA Sticker Model to Cryptanyze One Time Pad Polyalphabetic Cipher.

[14] Christopher, C., Gunawan, A., & Prima, S. (2022). Encrypted Short Message Service Design Using Combination of Modified Advanced Encryption Standard (AES) and Vigenere Cipher Algorithm. Engineering, MAthematics and Computer Science (EMACS) Journal, 4(2), 73–77. https://doi.org/10.21512/emacsjournal.v4i2.8273.

[15] Mahmudah, M., Novita Irawati, T., & Islam Jember, U. (2020). Aplikasi Kriptosistem Polyalphabetic Cipher. AXIOMA Jurnal Program Studi Pendidikan Matematika Universitas Islam Jember, 5(1), 11–19.

[16] Imam Riadi, Abdul Fadlil, & Fahmi Auliya Tsani. (2022). Pengamanan Citra Digital Berbasis Kriptografi Menggunakan Algoritma Vigenere Cipher. JISKA (Jurnal Informatika Sunan Kalijaga), 7(1), 33–45 https://doi.org/10.14421/jiska.2022.7.1.33-45.

[17] Yaseen, B. S. (2024). Parallel Search Using Probabilistic DNA Sticker Model to Cryptanyze One Time Pad Polyalphabetic Cipher. Iraqi Journal for Electrical and Electronic Engineering, 20(1), 104–110. https://doi.org/10.37917/ijeee.20.1.11.

[18] Intani, K., Wulandari, G., Hasanah, H., Syarifudin, I. L., & Brillian, E. M. C. (2023). Meningkatkan Keamanan Pesan Rahasia pada Vigenere Cipher Menggunakan Kombinasi Caesar Cipher dan Multiple-Key. J I M P - Jurnal Informatika Merdeka Pasuruan, 8(1), 28. https://doi.org/10.51213/jimp.v8i1.847