KEAMANAN SISTEM INFORMASI

TUGAS PERTEMUAN 5 TRANSPOSITION CIPHERS, SUBSTITUTION CIPHERS, DAN HACKING VIGÈNERE CIPHERS

Dosen Pengampu: Nano Yulian Pratama, S.S.T., M.T.

Disusun oleh :

Muhammad Zabbar Falihin 222112225

PROGRAM STUDI D-IV KOMPUTASI STATISTIK

POLITEKNIK STATISTIKA STIS

2024

DAFTAR ISI

A. Transposition Ciphers

1. Algoritma

Transposition Cipher melibatkan pengaturan ulang huruf-huruf dalam teks asli berdasarkan suatu skema atau pola tertentu. Salah satu contoh sederhana adalah Rail Fence Cipher, di mana teks dipecah menjadi dua baris berdasarkan urutan genap dan ganjil karakter. Contoh lain adalah Columnar Transposition, di mana teks ditulis dalam baris pada sebuah grid sesuai dengan kata kunci, dan kolom-kolom grid tersebut diatur ulang berdasarkan urutan alfabetis dari huruf-huruf dalam kata kunci.

Algoritma umum untuk Transposition Cipher adalah sebagai berikut.

- Enkripsi
    1. Tentukan kunci dan permutasi kolom
    2. Tulis teks asli dalam baris pada matriks dengan jumlah kolom sesuai kunci
    3. Mengisi ruang kosong jika pesan tidak sempurna memenuhi matriks
    4. Baca dan catat teks dalam matriks kolom demi kolom mengikuti urutan permutasi yang ditentukan oleh kata kunci, dan gabungkan karakter-karakter untuk mendapatkan ciphertext
- Dekripsi
    1. Hitung jumlah baris berdasarkan panjang ciphertext dan jumlah kolom
    2. Tulis ulang ciphertext ke dalam matriks
    3. Mengatur ulang kolom
    4. Baca matriks baris demi baris untuk mendapatkan plaintext

2. Kompleksitas Waktu

Worst-case complexity time dari algoritma enkripsi dan dekripsi dalam Transposition Cipher untuk metode sederhana seperti Rail Fence Cipher, kompleksitasnya adalah O(n), karena setiap karakter hanya perlu dituliskan dan dibaca sesuai urutan transposisi. Kompleksitas waktu untuk Columnar Transposition Cipher adalah O(n), di mana n adalah panjang dari teks. Ini karena setiap karakter dalam teks harus ditulis ke dalam matriks dan kemudian dibaca kembali. Proses pembuatan matriks dan pembacaan kolom dapat dilakukan dalam satu iterasi melalui teks, membuat operasi ini sangat efisien.

Enkripsi dan Dekripsi

- Operasi Utama: Menulis teks ke dalam matriks, mengisi ruang kosong jika perlu, dan membaca teks dari matriks baik secara kolom demi kolom (enkripsi) maupun baris demi baris (dekripsi).
- Enkripsi: O(n), di mana n adalah panjang dari teks asli. Ini karena setiap karakter harus ditulis ke dalam matriks dan kemudian dibaca lagi untuk membentuk ciphertext.
- Dekripsi: O(n), sama seperti enkripsi karena prosesnya adalah kebalikan dari enkripsi, dengan setiap karakter di matriks dibaca untuk membentuk plaintext.

Jadi, *worst-case complexity time* untuk enkripsi dan dekripsi Transposition Cipher adalah O(n), di mana n adalah panjang teks. Proses penulisan dan pembacaan matriks melibatkan akses sekali ke setiap karakter.

3. Program *Encrypt* dan *Decrypt*

```
def encrypt(text, num_cols):
    # Hilangkan spasi untuk kesederhanaan
    text = text.replace(" ", "").upper()
    # Inisialisasi ciphertext
    ciphertext = ""
    # Iterasi melalui setiap kolom
    for col in range(num_cols):
        idx = col
        while idx < len(text):
            ciphertext += text[idx]
            idx += num_cols
    return ciphertext

def decrypt(ciphertext, num_cols):
    # Menghitung jumlah baris yang diperlukan
    num_rows = len(ciphertext) // num_cols
    # Menangani kasus di mana ciphertext tidak memenuhi matriks secara
sempurna
    if len(ciphertext) % num_cols != 0:
        num_rows += 1
```

```
    # Membuat matriks kosong untuk dekripsi
    matrix = [''] * num_cols

    # Menghitung jumlah karakter ekstra di baris terakhir
    extra_chars = len(ciphertext) % num_cols

    # Mengisi matriks untuk dekripsi
    idx = 0
    for col in range(num_cols):
        for row in range(num_rows):
            if row == num_rows - 1 and col >= extra_chars and
extra_chars != 0:
                # Jika baris terakhir dan kolom melebihi jumlah
karakter ekstra, lanjutkan
                matrix[col] += ' '
            else:
                matrix[col] += ciphertext[idx]
                idx += 1

    # Menggabungkan karakter dari matriks untuk mendapatkan plaintext
    plaintext = ''
    for row in range(num_rows):
        for col in range(num_cols):
            if matrix[col][row] != ' ':
                plaintext += matrix[col][row]

    return plaintext

# Contoh penggunaan
text = "Muhammad Zabbar Falihin"
num_cols = 4

encrypted_text = encrypt(text, num_cols)
print("Encrypted Text:", encrypted_text)

decrypted_text = decrypt(encrypted_text, num_cols)
print("Decrypted Text:", decrypted_text)
```

- Fungsi encrypt yaitu teks asli diubah dengan menghilangkan spasi dan mengonversi semua karakter menjadi huruf besar. Karakter-karakter ini kemudian disusun dalam matriks imajiner berdasarkan jumlah kolom yang ditentukan oleh kunci (dalam hal ini, num_cols). Ciphertext dihasilkan dengan membaca matriks kolom demi kolom.
- Fungsi decrypt yaitu ciphertext dipecah menjadi matriks berdasarkan jumlah kolom yang sama dengan proses enkripsi. Matriks ini diisi kolom demi kolom dengan karakter dari ciphertext. Plaintext asli diperoleh dengan membaca matriks baris demi baris, mengabaikan karakter pengisi jika ada.
- Kompleksitas waktu dari kedua fungsi ini adalah O(n) atau menunjukkan kompleksitas waktu linier, menjadikannya efisien untuk enkripsi dan dekripsi teks dengan panjang apa pun

B. Substitution Ciphers

1. Algoritma

Substitution Cipher adalah teknik kriptografi dasar di mana setiap elemen (biasanya huruf) dalam teks asli (plaintext) diganti dengan elemen lain berdasarkan suatu sistem substitusi. Ini mencakup berbagai jenis cipher, salah satu contohnya adalah Caesar Cipher. Di Caesar Cipher, substitusi dilakukan dengan menggeser huruf-huruf dalam alfabet sejumlah langkah tertentu.

Algoritma umum untuk Substitution Cipher adalah sebagai berikut:

- Tentukan kunci substitusi. Untuk Caesar Cipher, kunci adalah jumlah pergeseran.
- Untuk enkripsi, geser setiap huruf dari teks asli sesuai jumlah pergeseran yang ditentukan oleh kunci.
- Untuk dekripsi, lakukan proses yang berlawanan dengan enkripsi, yaitu menggeser kembali huruf-huruf yang telah dienkripsi sejumlah langkah yang sama, tapi ke arah berlawanan.

2. Kompleksitas Waktu

Worst-case complexity time dari algoritma enkripsi dan dekripsi dalam Substitution Cipher, termasuk Caesar Cipher, adalah O(n), di mana n adalah panjang dari teks yang dienkripsi atau didekripsi. Ini karena setiap huruf dalam teks harus diperiksa dan mungkin digeser satu per satu. Tidak ada operasi yang membutuhkan lebih dari waktu konstan untuk

setiap huruf, sehingga total waktu yang diperlukan berkaitan langsung dengan jumlah huruf dalam teks.

Enkripsi dan Dekripsi

- Operasi Utama: Menggeser setiap huruf dari teks asli sesuai dengan jumlah pergeseran yang ditentukan oleh kunci untuk enkripsi, dan melakukan proses sebaliknya untuk dekripsi.
- Enkripsi dan Dekripsi: O(n), dengan n adalah panjang dari teks. Setiap huruf dalam teks diakses dan diubah (digeser) satu kali selama enkripsi atau dekripsi.

Jadi, *worst-case complexity time* untuk enkripsi dan dekripsi Transposition Cipher adalah O(n), di mana n adalah panjang teks. Setiap karakter hanya diakses dan diubah sekali.

3. Program *Encrypt* dan *Decrypt*

- **Source Code substitution.py**

```
def encrypt(text, shift):
    result = ""
    for i in range(len(text)):
        char = text[i]
        if char.isalpha():  # Cek apakah karakter adalah huruf
            # Proses untuk huruf kapital
            if char.isupper():
                result += chr((ord(char) + shift - 65) % 26 + 65)
            # Proses untuk huruf kecil
            else:
                result += chr((ord(char) + shift - 97) % 26 + 97)
        else:
            # Tambahkan karakter asli jika bukan huruf
            result += char
    return result

def decrypt(encrypted_text, shift):
    return encrypt(encrypted_text, -shift)

# Contoh penggunaan
text = "Muhammad Zabbar Falihin!"
```

```
shift = 4

encrypted_text = encrypt(text, shift)
print("Encrypted Text:", encrypted_text)

decrypted_text = decrypt(encrypted_text, shift)
print("Decrypted Text:", decrypted_text)
```

- Fungsi encrypt mengambil teks asli dan jumlah pergeseran sebagai argumen, lalu mengembalikan teks yang telah dienkripsi.
- Fungsi decrypt mengambil teks terenkripsi dan jumlah pergeseran sebagai argumen, menggunakan proses yang sama dengan enkripsi tapi dengan pergeseran ke arah berlawanan untuk mengembalikan teks ke bentuk aslinya.
- Kompleksitas waktu dari kedua fungsi ini adalah O(n) menunjukkan kompleksitas waktu linier, menjadikannya efisien untuk enkripsi dan dekripsi teks dengan panjang apa pun.

C. *Hacking* Vigènere Ciphers

1. Algoritma

Vigenère Cipher memperluas konsep dari Substitution Cipher dengan menggunakan kunci yang terdiri dari beberapa huruf, di mana setiap huruf menentukan pergeseran untuk huruf-huruf dalam teks asli. Ini menciptakan pola enkripsi yang lebih kompleks dan sulit ditebak dibandingkan dengan Caesar Cipher. Untuk meng-*hack* Vigenère Cipher tanpa mengetahui kunci, salah satu metode yang digunakan adalah analisis frekuensi dan Kasiski Examination.

a) Identifikasi Sekuens yang Berulang

Cari sekuens huruf yang berulang dalam teks terenkripsi. Jarak antara sekuens-sekuens yang sama ini bisa memberikan petunjuk tentang panjang kunci. Sekuens berulang merujuk pada pola huruf atau grup huruf yang muncul lebih dari satu kali dalam teks terenkripsi. Pencarian sekuens berulang ini merupakan bagian penting dari Kasiski Examination, sebuah metode untuk membantu menentukan panjang kunci dalam Vigenère Cipher.

b) Kasiski Examination

Kasiski Examination adalah metode yang lebih spesifik untuk meng-hack Vigenère Cipher. Kasiski Examination menggunakan jarak antar sekuens berulang untuk menentukan panjang kunci yang mungkin. Ide dasarnya adalah bahwa jarak tersebut seringkali merupakan kelipatan dari panjang kunci. Kasiski Examination membantu menentukan panjang kunci, yang merupakan informasi kritikal untuk memecahkan Vigenère Cipher.

c) Analisis Frekuensi

Setelah panjang kunci diduga, teks terenkripsi dapat dibagi menjadi beberapa bagian berdasarkan panjang kunci, dan analisis frekuensi dapat diterapkan pada setiap bagian untuk menebak huruf kunci.

d) Percobaan dan Kesalahan

Dengan menggunakan huruf-huruf kunci yang ditebak, dilakukan percobaan untuk mendekripsi teks. Ulangi proses dengan menyesuaikan tebakan sampai teks terdekripsi dengan benar terbaca.

D. Link Github Program Transposition Cipher, Substitution Cipher, dan *Hacking* Vigènere Ciphers serta Keseluruhan File

- **Source Code *Hacking* Vigènere Ciphers dan keseluruhan File dilampirkan di link Github sebagai berikut :**

https://github.com/zabbarfalih/ksi-pertemuan5