Improvisasi Algoritma Enkripsi Caesar Cipher dengan Index sebagai *Dynamic Parameter*

Aliif Arief Maulana (21/479029/SV/19418)
Fikri Yurcel Milano (21/473378/SV/18824)

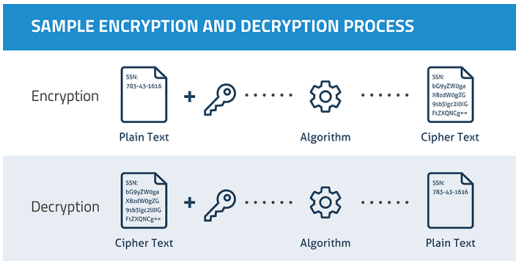
Program Studi Teknik Rekayasa Perangkat Lunak
Sekolah Vokasi, Departemen Teknik Elektro dan Informatika
Universitas Gadjah Mada, Bulaksumur, Caturtunggal, Depok, Sleman, Daerah Istimewa Yogyakarta

I. Apa itu Enkripsi dan Dekripsi

Enkripsi adalah proses untuk mengkonversikan informasi **menjadi kode yang tidak jelas** dengan tujuan untuk merahasiakan isi dari informasi asli agar tidak dapat dipahami dan menjadi rahasia dengan memproses informasi tersebut menggunakan algoritma, umumnya dalam proses enkripsi ada parameter tambahan yaitu key.

Dekripsi adalah proses untuk mengkonversikan kode rahasia **menjadi informasi *readable***, menggunakan sebuah key dan algoritma dekripsi.

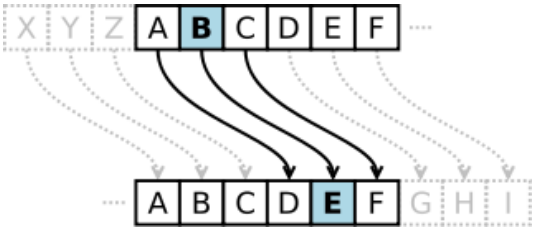
Key yang digunakan untuk enkripsi dan dekripsi **haruslah sama.**



Gambar 1.0 : encryption and decryption (Sumber: *https://mediawiki.middlebury.edu/LIS/Encryption*)

II. Improvisasi Caesar Cipher

Dalam kriptografi, sandi Caesar, atau sandi geser, kode Caesar atau Geseran Caesar adalah salah satu teknik enkripsi paling sederhana dan paling terkenal. Sandi ini termasuk sandi substitusi dimana setiap huruf pada teks terang digantikan oleh huruf lain yang memiliki selisih posisi tertentu dalam alfabet.



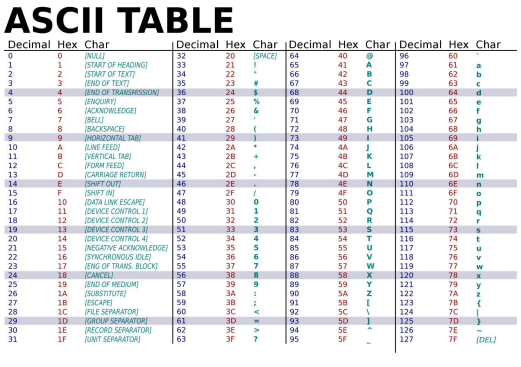
Gambar 2.0 : Ilustrasi Sandi Caesar (Sumber: *https://upload.wikimedia.org/wikipedia/commons/thumb/2/2b/Caesar3.svg/320px-Caesar3.svg.png*)

Algoritma enkripsi Caesar Cipher ini sebenarnya memiliki kelemahan yang sangat mudah untuk dipecahkan karena pola data yang telah dienkripsinya mudah dibaca dan umumnya cara untuk men *decrypt* paksa data

yang di enkripsi dengan algoritma Caesar Cipher adalah dengan menggunakan teknik *Brute Force* dengan mencoba berbagai parameter value key maka cepat atau lambat data asli akan cepat didapatkan.

Karena itu kami mencoba untuk melakukan improvisasi untuk meningkatkan abstraksi dan tingkat kerumitan dari algoritma Caesar Cipher dengan menambahkan satu parameter lagi yaitu urutan tiap karakter dalam data yang dienkripsi sehingga tiap karakter hasil enkripsi akan memiliki karakter yang berbeda dan dapat berubah dengan dinamis tergantung dengan urutan karakter dalam datanya.

Dalam implementasinya Algoritma yang kami terapkan, agar dapat menggunakan fungsi dalam bahasa pemrograman Python kami menggunakan Tabel ASCII sebagai pedoman standarisasi urutan karakter dengan indexnya, sebenarnya ini bisa dikustomisasi sesuai dengan kebutuhan untuk keabstrakan yang lebih rumit dengan membuat tabel karakter beserta indexnya lalu membuat fungsi konversinya sendiri namun agar mudah diilustrasikan kami akan menggunakan tabel ASCII.



Gambar 3.0 : ASCII Table (Sumber: *https://id.wikipedia.org/wiki/Berkas:ASCII-Table-wide.svg*)

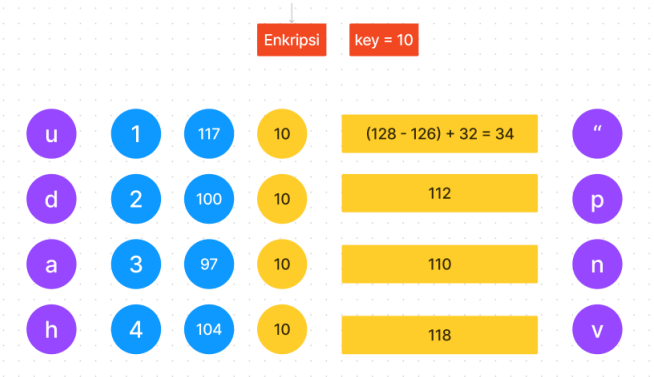
Kami mengambil urutan desimal karakter 32 sebagai batas bawah yaitu (spasi) sampai dengan urutan desimal karakter ke 126 sebagai batas teratas yaitu karakter (~). Untuk index urutan karakter dimulai dari 1 hingga panjang stringnya.

Agar panjang karakter hasil enkripsi bisa dinamis karena itu kami menggunakan struktur data ***Linked List*** untuk menyatukan setiap karakter yang sudah dienkripsi begitu juga ketika melakukan dekripsi kami menggunakan ***Linked List*** untuk menyatukan dan mengkonversi menjadi list.

III. Algoritma Caesar Cipher

- **Enkripsi**

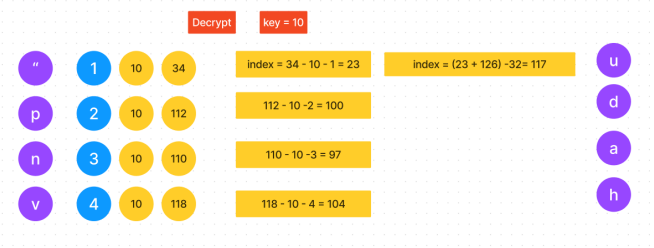
1. Masukkan String dan key
2. Ambil setiap karakter pada String
3. Pada setiap karakter, ubah menjadi bentuk kode ASCII nya
4. Pada setiap karakter, jumlahkan dengan **index + kode ASCII + key**
5. Pada setiap karakter, jika hasil **penjumlahan lebih dari 126**
    a. Maka jalankan operasi **(hasil - 126) + 32**
    b. Lalu dari hasil operasi, ubah menjadi bentuk karakternya
6. Pada setiap karakter, jika hasil **penjumlahan tidak lebih dari 126**
    a. Maka langsung diubah menjadi bentuk karakternya



Gambar 4.0 : enkripsi (Sumber: *Penulis*)

- **Dekripsi**

1. Masukkan String dan key
2. Ambil setiap karakter pada String
3. Pada setiap karakter, ubah menjadi bentuk kode ASCII nya
4. Pada setiap karakter, jumlahkan dengan **index - kode ASCII - key**
5. Pada setiap karakter, jika hasil **penjumlahan kurang dari 32**
    a. Maka jalankan operasi **(hasil + 126) - 32**
    b. Lalu dari hasil operasi, ubah menjadi bentuk karakternya
6. Pada setiap karakter, jika hasil **penjumlahan tidak kurang dari 32**
    a. Maka ubah menjadi bentuk karakternya



Gambar 5.0 : dekripsi (Sumber: *Penulis*)

IV. Penjelasan Implementasi Kode

Kode dibawah ini merupakan fungsi untuk implementasi algoritma improvisasi enkripsi Caesar Cipher dengan menggunakan bahasa Python 3. Cara kerja dari fungsi ini yaitu pertama fungsi menerima parameter string *plaintext* dan *key* nya lalu akan membuat objek *linked list* yang berguna untuk menampung karakter yang sudah dikonversi lewat proses enkripsi, lalu ketika proses enkripsi tiap karakter selesai maka fungsi akan mengembalikan string yang sudah terenkripsi.

```
def caesar_encrypt(string, key):
    linked_list = LinkedList()
    for i in range(1, len(string) + 1):
        char = string[i - 1]
        index = ord(char) + key + i
        if index > 126:
            index = (index - 126) + 32
            linked_list.add_last(chr(index))
        else:
            linked_list.add_last(chr(index))
    result = "".join(linked_list.as_array())
    return
```

Gambar 6.0 : kode enkripsi (Sumber: *Penulis*)

Kode dibawah ini merupakan fungsi untuk implementasi algoritma improvisasi dekripsi Caesar Cipher dengan menggunakan bahasa Python 3. Cara kerja programnya kurang lebih sama seperti enkripsi namun berbeda pada proses konversi tiap karakternya saja dan parameternya.

```
def caesar_decrypt(encoded, key):
    linked_list = LinkedList()
    for i in range(1, len(encoded) + 1):
        char = encoded[i - 1]
        index = ord(char) - key - i
        if index < 32:
            index = (index + 126) - 32
            linked_list.add_last(chr(index))
        else:
            linked_list.add_last(chr(index))
    result = "".join(linked_list.as_array())
    return result
```

Gambar 7.0 : kode dekripsi (Sumber: *Penulis*)

V. Demo, Testing, dan Logging

Untuk membuktikan kevalidan dari algoritma yang kami buat tentunya kami harus mengujinya, cara yang kami gunakan untuk melakukan test kepada algoritma enkripsi yang kami buat yaitu dengan melakukan *automate testing* via Github Actions, untuk lebih jelasnya seperti ini alurnya :

1. Algoritma sudah teruji saat jalan di lokal diuji dengan menggunakan unit test.

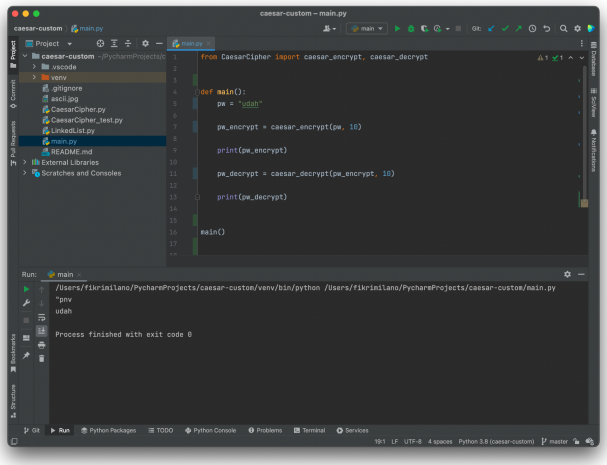
2. Untuk memastikan kevalidan algoritma kami membuat program generator string dan key dengan menggunakan modul random python.

3. Agar testing dapat berjalan terus menerus dan memiliki banyak test case string dan key maka kami memanfaatkan fitur scheduled testing di Github Actions dengan menggunakan *cron job*.

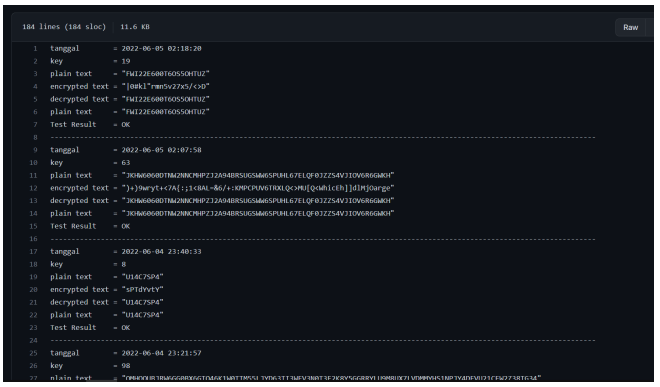
4. Setiap program melakukan *testing* maka hasil testing tersebut akan di *logging* untuk mengetahui hasilnya dalam file log.txt

5. Semua proses akan terjadi setiap 30 menit sekali bisa disesuaikan dengan cron job nya kali ini kami mengaturnya tiap 30 menit sekali, setiap aktifitas push ke repository lewat *master branch* testing juga akan dijalankan.

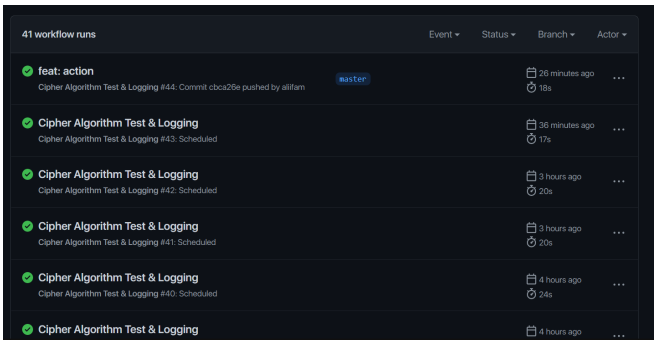
6. Dengan ini semua sudah terotomatisasi dan kita hanya perlu mengecek logging dari proses *automated testing* tersebut.



Gambar 8.0 : Hasil *testing* manual program di komputer lokal (Sumber : *Penulis*)



Gambar 9.0 : *log history* hasil testing di Github Actions (Sumber : *Penulis*)



Gambar 10.0 : *automate testing* dengan Github Actions (Sumber : *Penulis*)

VI. Source code

Berikut link public Github repository tempat program disimpan dan di test otomatis : github.com/aliifam/caesar-custom

Keterangan file :

- CaesarCipher.py : inti dari fungsi *encrypt* dan *decrypt* program.
- CaesarCipher_test.py : Unit testing dan logging.
- logFormatter.py : fungsi untuk memformat history saat logging agar log terbaru selalu berada di paling atas
- log.txt : history hasil testing program.
- .github/workflows/python-test.yml : *script* standar untuk melakukan CI & CD *(continuous integration and continuous delivery)* via Github Actions.

VII. Referensi

[1] Encryption and Decryption Image - Middlebury. Retrieved June 3, 2022, from https://mediawiki.middlebury.edu/LIS/Encryption

[2] Gambar Ilustrasi Sandi Caesar - Wikimedia. Retrieved June 3 2022, from https://upload.wikimedia.org/wikipedia/commons/thumb/2/2b/Caesar3.svg/320px-Caesar3.svg.png

[3] ASCII Table Image - Wikipedia. Retrieved June 3 2022, from https://id.wikipedia.org/wiki/Berkas:ASCII-Table-wide.svg

VIII. Ucapan Terimakasih

Kami mengucapkan terimakasih dan syukur kepada Allah SWT yang atas segala nikmatnya yang tak terhitung sehingga kami dapat menyelesaikan proyek ini dengan baik.

IX. Pernyataan

Dengan ini kami menyatakan bahwa laporan yang kami tulis ini adalah tulisan kami sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Aliif Arief Maulana Fikri Yurcel Milano