Implementasi Pembangkit Bilangan Acak Semu dalam Pembangkitan Tulisan *Leet Speak*

Alifia Rahmah - 13520122[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13520122@mahasiswa.itb.ac.id

***Abstract*—*Leet Speak* adalah gaya penulisan dengan mengganti tiap huruf di tiap kata dalam kalimat menjadi kombinasi huruf, simbol, dan angka. *Leet speak* untuk satu kata dapat memiliki banyak kemungkinan. Untuk membangkitkan tulisan *leet speak*, dapat digunakan pembangkit bilangan acak semu (*pseudo-random number generator*) dengan menyimpan kumpulan ekuivalensi angka atau simbol dari setiap huruf dalam alfabet dalam sebuah *hash map* dan memilih simbol dan angka yang ekuivalen dengan tiap huruf secara acak.**

***Keywords*—*leet speak*, acak, *hash map*.**

I. PENDAHULUAN

Leet speak adalah gaya penulisan yang mengubah suatu kata dalam kalimat yang awalnya terdiri dari huruf alfabet menjadi kombinasi antara huruf, angka, dan simbol dengan bentuk yang mirip dengan huruf tersebut. Satu huruf dalam alfabet dapat punya lebih dari satu pengganti dalam gaya penulisan *leet speak*. Gaya penulisan *leet speak* sempat populer di kalangan pengguna internet, dan kini masih banyak digunakan oleh pengguna internet di komunitas *hacking* dan *gaming*. Dalam penulisan *leet speak*, satu kata atau kalimat dapat memiliki kombinasi penulisan yang unik.

Dalam makalah ini, akan dirancang program yang mengubah tulisan yang awalnya ditulis dalam huruf alfabet dengan baik dan benar, menjadi tulisan *leet speak* yang terdiri dari huruf, angka, dan simbol, dengan pemilihan karakter yang ekuivalen dengan huruf pada tiap kata dengan menggunakan pembangkit bilangan acak semu.

II. TEORI DASAR

A. Teori Bilangan

Bilangan bulat adalah bilangan yang tidak mempunyai pecahan desimal. Lawan dari bilangan bulat adalah bilangan riil, yang mempunyai nilai pecahan. Sebuah bilangan bulat a habis membagi bilangan bulat lain b jika terdapat suatu bilangan c sedemikian rupa sehingga $b = ac$.

Teorema Euclidean menyatakan ketika m dan n adalah bilangan bulat dengan $n > 0$, jika m dibagi dengan n maka hasil pembagian dua bilangan tersebut adalah q (*quotient*), dengan sisa hasil bagi r (*remainder*), sedemikian sehingga $m = nq + r$, dengan $0 \leq r < n$.

Sisa hasil bagi dari pembagian dua buah bilangan bulat disebut modulus atau modulo. Operasi a mod m memberikan sisa jika a dibagi dengan m. Hasil operasi modulo dengan bilangan bulat m yaitu x selalu terletak dalam interval $0 \leq x < m$. Ilmu yang mempelajari tentang sifat operasi modulo disebut aritmatika modulo.

B. Pembangkit Bilangan Acak Semu

Pembangkit bilangan acak (*random number generator*) adalah aplikasi dari teori bilangan sebagai metode untuk membangkitkan bilangan yang tidak dapat ditentukan secara pasti. Implementasi dari pembangkit bilangan acak banyak terdapat dalam ilmu komputasi. Sejatinya, tidak ada persamaan matematika yang pasti menghasilkan bilangan acak yang benar-benar tidak dapat ditentukan secara pasti. Semua bilangan acak yang dihasilkan oleh dari persamaan matematika dalam operasi komputer bersifat acak semu, sehingga pembangkit bilangan acak yang dihasilkan dari prosedur komputasi disebut juga sebagai pembangkit bilangan acak semu (*pseudo-random number generator*).

Pembangkit bilangan acak semu memproses bilangan dari suatu umpan. Umpan ini dapat menentukan bilangan apa yang akan dihasilkan dari pembangkit bilangan acak semu. Bilangan acak dari pembangkitan dapat ditentukan dengan mudah apabila umpan untuk pembangkit tersebut diketahui. Terdapat beberapa metode untuk membangkitkan bilangan acak semu, salah satunya *Linear Congruential Generator* (LCG).

Linear Congruential Generator (LCG), atau pembangkit bilangan acak kongruen lanjar memiliki bentuk

$$X_n = (aX_{n-1} + b) \bmod m$$

Dengan a, b, dan m adalah suatu bilangan, dan X_0 sebagai umpan awal. Jika $b = 0$, pembangkit ini juga disebut sebagai *Multiplicative Congruential Generator* (MCG), atau Lehmer RNG.

Karena hasil LCG merupakan hasil modulo dengan m, bilangan acak yang dibangkitkan dari LCG tidak akan lebih besar daripada m. LCG unggul karena persamaan yang sederhana dan waktu komputasi yang cepat, namun tidak dapat digunakan pada kriptografi karena urutan bilangan acak yang mudah ditentukan. Mesipun demikian, LCG sangat efisien untuk digunakan untuk aplikasi non-kriptografi.

C. Map dan Hash Table

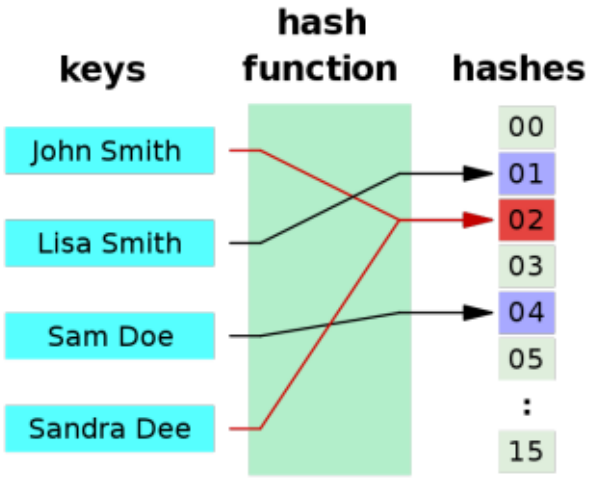
Abstract data type (ADT) Map adalah tipe data yang berupa kumpulan pasangan/*binding* dari *key* dan *value*, dengan nilai *key* bersifat unik di dalam kumpulan tersebut. Tipe data Map disebut juga *associative array*, *symbol table*, atau *dictionary*. Terdapat beberapa operasi terhadap Map, yaitu penambahan pasangan baru, penghapusan suatu pasangan, modifikasi nilai *value* dari pasangan, dan pencarian nilai *value* dari *key* tertentu.

Jika jumlah elemen sedikit, implementasi dari Map dapat dilakukan dengan menggunakan association list dalam bentuk *array* dengan elemen berupa pasangan *key* dan *value*. Namun jika terdapat banyak pasangan, digunakan pemetaan *key* ke indeks dengan isi elemen *value*, yang umum diimplementasikan dengan memanfaatkan *hash table*.

Hash table atau *hash map* memanfaatkan fungsi *hash* untuk menentukan di mana data disimpan dan bagaimana menemukan data kembali, dengan mengubah *key* menjadi suatu *hash* yang dapat digunakan sebagai indeks. Fungsi *hash* adalah fungsi yang dapat digunakan untuk memetakan sebuah data (*key*) menjadi sebuah nilai berukuran tetap. Dalam fungsi *hash*, terdapat istilah *key* yaitu data yang menjadi masukan, dan *hash* atau *digest* yang berupa nilai hasil perhitungan fungsi *hash*. Fungsi hash yang baik yaitu fungsi hash dengan nilai komputasi yang cepat dan meminimalisir terdapat *collision*. Dengan fungsi *hash* ini, *key* dan *value* dapat disimpan pada lokasi indeks hasil *hash* tersebut.

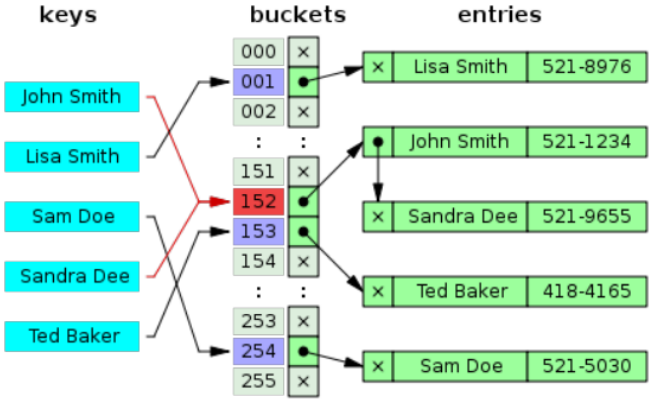
Dalam metode penyimpanan *key* dan *value* ke dalam *array*, untuk melakukan pencarian perlu dilakukan iterasi pencarian indeks dengan *key* yang cocok dengan yang diinginkan, lalu mengambil *value* yang berpasangan dengan *key* tersebut. Dalam kasus terburuk, dengan pasangan *key* dan *value* yang terletak di akhir *array*, sehingga kompleksitas pencarian tersebut adalah O(N). Sedangkan pada tipe data Map dengan menggunakan *hash table*, pencarian elemen dilakukan dengan menggunakan fungsi *hash* pada *key* untuk langsung menuju indeks tempat *value* berada. Sehingga operasi pencarian dengan *hash table* dilakukan dengan kompleksitas O(1), jauh lebih efisien dibanding menggunakan *array*.

Terkadang, dapat terjadi masalah saat menyimpan data, yaitu hasil *hash* suatu key dengan key yang lain bernilai sama. Masalah ini disebut *collision*/bentrokan. Jika terjadi *collision*, suatu *key* yang seharusnya digunakan untuk mengakses suatu *value* akan mengakses value yang seharusnya diakses oleh *key* lain karena hasil *hash* dari dua key tersebut bernilai sama.

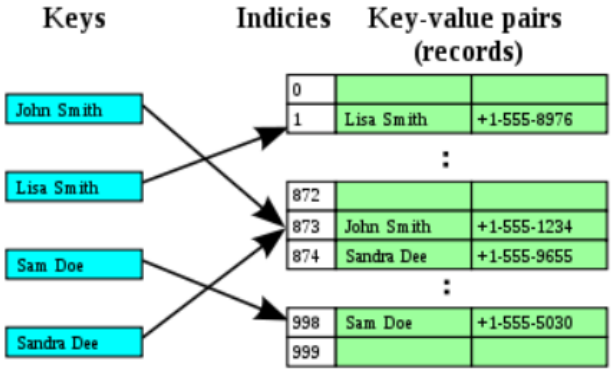


Gambar 1 Ilustrasi collision pada fungsi hash
(sumber: Bahan Kuliah IF2110/IF2111)

Collision dapat diselesaikan dengan beberapa cara, salah satunya *hash chaining*, yaitu dengan mengikat elemen *value* yang memiliki hasil *hash key* sama dalam tipe data lain. Pengikatan ini dapat dilakukan dengan menggunakan ADT *linked list* atau *array* dinamis. Selain itu, *collision* juga dapat diselesaikan dengan metode *open addresing*, yaitu mencari lokasi indeks alternatif hingga pasangan yang diinginkan ditemukan, atau ditemukan lokasi indeks yang tidak terpakai.



Gambar 2 Metode *hash chaining* untuk menangani *collision* pada *hash table*, dengan menggunakan *linked list* untuk pengikatan elemen-elemen dengan *hash key* yang sama
(sumber: Bahan Kuliah IF2110/IF21)



Gambar 3 Metode *open addresing* untuk menangani *collision* pada *hash table* (sumber: Bahan Kuliah IF2110/IF21)

D. Leet Speak

Leet speak, atau disebut juga *1337 speak*, adalah gaya penulisan di internet yang mengubah mengubah satu kata dalam kalimat yang awalnya terdiri dari huruf alfabet, menjadi kombinasi antara huruf, angka, dan simbol. *Leet* berasal dari kata *elite*, yang memiliki arti orang-orang terbaik/khusus, yang memiliki arti pemakai *leet speak* merasa dirinya termasuk orang-orang khusus atau spesial dalam sebuah kelompok dibandingkan orang lain. *Leet speak* banyak digunakan oleh peselancar internet, terutama komunitas *hacking* dan *gaming*.

Gambar 4 Penggunaan *leet speak* untuk penulisan nama samar pelaku dalam website yang terkena serangan *deface* (sumber: cyberkendra.com)

Leet speak mulai muncul sekitar tahun 1980-an dan populer sekitar 1990-an, yaitu pada masa-masa internet mulai populer di masyarakat. Awal mula *leet speak* berasal dari usaha anggota komunitas internet untuk melewati *filter* dari *Bulletin Board System* (BBS) dalam forum. Sistem BBS dalam forum menyaring kata-kata yang tidak pantas dibicarakan agar tidak muncul dalam forum tersebut. Dengan *leet speak*, pengguna dapat menulis kata-kata yang termasuk topik yang tidak pantas dibicarakan tanpa terkena konsekuensi dari sistem *filter* di dalam forum. Akhir-akhir ini, penggunaan *leet speak* sudah mulai bergeser untuk membercandakan orang lain di internet secara sarkas, atau digunakan komunitas *hacker* untuk memberi nama samar bagi dirinya sendiri saat melakukan aksi kejahatan.

Cara kerja *leet speak* adalah mengganti tiap huruf dalam suatu kata menjadi simbol atau angka, dapat pula berupa kombinasi keduanya, dengan bentuk yang sekilas mirip dengan bentuk huruf tersebut.

A	4 , @ , /\ , /-\ , ? , ^ , α , λ
B	8 , |3 , ß , I³ , 13 , I3 , J3
C	(, [, < , © , ¢
D	|) , |] , Ð , đ , 1)
E	3 , € , & , £ , ε
F	|= , PH , |*|-| , |" , *f* , I²
G	6 , & , 9
H	# , 4 , |-| , }{ ,]-[, /-/ ,)-(, 1-1
I	! , 1 , | ,][, i
J	_| , ¿
K	|< , |{ , |(, X
L	1 , |_ , £ , | ,][_
M	/\/\ , /v\ , |V| ,]V[, |\/| , AA , []V[] , |11 , /|\ , ^^ , (V) , |Y| , !\/!
N	|| , /\/ , /V , |V , /\\/ , |1 , 2 , ? , (\) , 11 , r , !\!
O	0 , 9 , () , [] , * , ° , <> , ø , {[]}
P	9 , |° , p , |> , |* , []D ,][D , |² , |? , |D
Q	0_ , 0,
R	2 , |2 , 1² , ® , ? , я , 12 , .-
S	5 , $, § , ? , ŝ , ş
T	7 , + , † , '][' , |
U	|_| , µ , [_] , v
V	\/ , |/ , | , \'
W	\/\/ , VV , \A/ , \\' , uu , \^/ , \\|/ , uJ
X	>< ,)(, }{ , % , ? , × ,][
Y	`/ , °/ , ¥
Z	z , 2 , "/_

Tabel 1 Penggantian huruf menjadi kumpulan angka dan simbol untuk untuk penulisan *leet speak* (sumber: interestingengineering.com)

Selain huruf yang berubah menjadi kombinasi huruf dan simbol, terkadang ada kata-kata yang penulisannya berubah menjadi lebih singkat, seperti *you* menjadi *u* dan *right now* menjadi *rn*, serta tata kalimat yang lebih singkat.

Di Indonesia, istilah *leet speak* tidak terlalu populer. Ekuivalensi dari leet speak dalam tren internet Indonesia adalah bahasa *gaul* yang sempat populer tahun 2010-an, dan sering digunakan di media sosial oleh anak-anak remaja. Pada saat itu, orang-orang lebih sering menyebut cara mengetik demikian sebagai bahasa *alay*.

Diperkirakan awal kemunculan tulisan alay berasal dari masa-masa marak penggunaan *Short Message Service* (SMS) sebagai media komunikasi. Pengiriman satu pesan SMS memiliki batas karakter, dengan penambahan biaya pulsa yang harus dikeluarkan saat mengirim pesan. Penulisan pesan secara singkat ini seringkali digunakan untuk menghemat biaya saat mengirim pesan dengan tulisan yang banyak. Namun seiring

berjalannya waktu, tulisan *alay* ini juga marak digunakan baik di media cetak maupun di media sosial pada masa itu.

Gambar 5 Contoh tulisan *alay* dalam media sosial. (sumber: ferboes.com)

III. IMPLEMENTASI

A. Rancangan Awal

Garis besar dari pembangkitan tulisan *leet speak* adalah dengan memilih simbol atau angka pengganti dari tiap huruf dalam kata yang ingin diubah menjadi tulisan *leet speak*. Dengan adanya kemungkinan lebih dari satu, maka dipilih salah satu dari simbol secara acak. Program ini menerima masukan string kata yang akan diubah menjadi tulisan *leet speak*, mencari kombinasi karakter yang ekuivalen dengan tiap huruf dengan indeks secara acak yang akan dibangkitkan dengan pembangkit bilangan acak semu, dan mengeluarkan keluaran kombinasi karakter untuk masing-masing huruf.

Daftar pemetaan huruf alfabet ke dalam *leet speak* disusun dalam tipe data *map* yang memetakan tiap huruf ke *array* berisi kumpulan kombinasi karakter yang menjadi ekuivalensi dari setiap huruf tersebut. Dari setiap huruf dalam kombinasi karakter, akan dipilih satu indeks *array* sebagai hasil konversi dari tiap huruf menjadi kombinasi karakter, melalui pembangkitan indeks secara acak semu.

Untuk melakukan pembangkitan indeks secara acak semu, digunakan *Linear Congruential Generator* (LCG) yang dimodifikasi, dengan pertimbangan pembangkitan indeks ini tidak memerlukan algoritma pembangkit bilangan acak semu yang aman secara kriptografi, dan merupakan algoritma pembangkitan bilangan acak yang paling sederhana dan efisien. Hasil dari pembangkitan harus berupa indeks dari huruf tersebut, sehingga dari rumus LCG

$$X_n = (aX_{n-1} + b) \bmod m,$$

dapat dipilih bilangan *a*, *b*, dan *m* sedemikian dengan nilai yang berbeda setiap dilakukan operasi, sehingga hasil pembangkitan dapat bersifat acak semu. Agar hasil pembangkitan berupa bilangan yang selalu berada di rentang indeks *list*, dipilih nilai *m* berupa panjang *array value* hasil pemetaan suatu huruf. Dengan demikian, nilai *a* dan *b* dapat dipilih secara acak, sehingga dapat dibangkitkan menggunakan patokan bilangan yang selalu berubah, seperti waktu atau tanggal. Dalam program ini, dipilih nilai *a* detik saat ini dan nilai *b* berupa mikrodetik saat ini agar hasil pembangkitan dapat teracak sedemikian rupa.

B. Implementasi Program

Dari hasil rancangan awal, pembangkit tulisan *leet speak* diimplementasikan dengan bahasa Python dengan pertimbangan sintaks Python yang sederhana dan terdapat berbagai modul untuk memudahkan implementasi program.

Implementasi pemetaan karakter dilakukan dengan bantuan modul `defaultdic` dalam library `collections`, yang memiliki dapaet melakukan pemetaan karakter sebagai *key* dan tipe data *list* sebagai *value*. Pemetaan karakter dimasukkan dalam file terpisah agar modularitas program dapat terjaga. File berisi pemetaan setiap karakter alfabet ini dinamakan `leetdict.py`, dengan isi pemetaan sesuai pemetaan pada Tabel 1, ke dalam variabel `d`.

```
from collections import defaultdict


d = defaultdict(list)

d['a'] = ['4', '@', "/\\", '?', '^', 'α',
'λ']

d['b'] = ['8', '|3', 'ß', '13', 'I3',
'J3']

d['c'] = ['(', '[', '<', '©', '¢']

d['d'] = ['|)', '|]', 'Ð', 'đ', '1)']

d['e'] = ['3', '€', '&', '£', 'ε']

...

d['w'] = ["\\/\\/", 'VV', "\\A/",
"\\\\'", 'uu', "\\^/", "\\|/", 'uJ']

d['x'] = ['><', ')(', '}{', '%', '?',
'×', '][']

d['y'] = ['`/', '°/', '¥']

d['z'] = ['z', '2', '"/_']
```

Untuk pembangkitan bilangan acak semu menggunakan LCG, diimplementasikan dengan membuat fungsi yang dipisah dalam file bernama `prng.py`. Fungsi ini menerima parameter

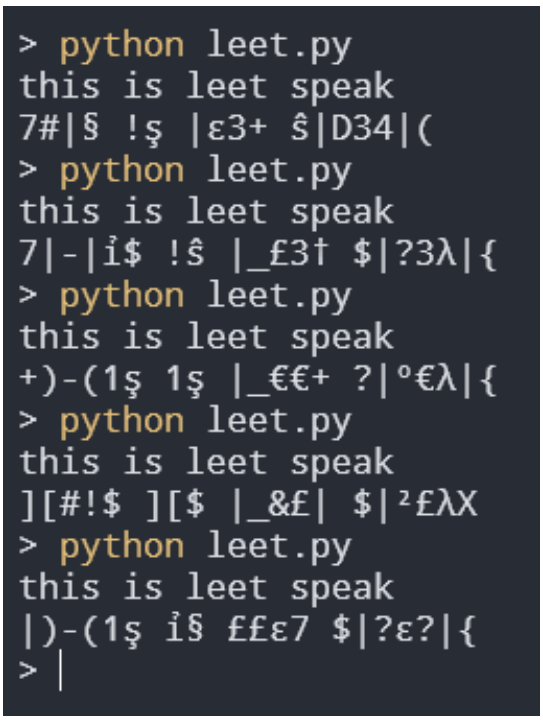
umpan (`xo`) dan pembagi `m`, lalu mengembalikan hasil perhitungan sesuai persamaan LCG. Untuk mengambil detik dan mikrodetik, digunakan dari modul `datetime` yang mendukung pengambilan komponen dalam waktu dan tanggal., dengan perintah `datetime.now().second` untuk mengambil detik dan `datetime.now().microsecond` untuk mengambil mikrodetik.

```
from datetime import datetime


def LCG(xo, m):
    a = datetime.now().second
    b = datetime.now().microsecond
    x = (a * xo + b) % m
    return x
```

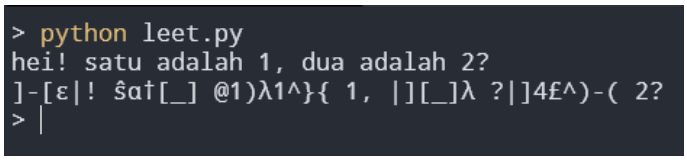
Pada program utama, dilakukan pemanggilan variabel `d` dari file `leetdict.py` dan fungsi LCG dari file `prng.py`. Program utama dari pembangkitan tulisan *leet speak* ini menerima masukan string, lalu dilakukan iterasi tiap karakter ke string. Pada tiap iterasi, dilakukan terlebih dahulu pengecekan apakah karakter merupakan alfabet atau bukan dengan mengecek kode ASCII dari huruf tersebut. Jika karakter merupakan alfabet, kombinasi karakter *leet speak* yang ekuivalen dengan karakter tersebut digabung ke string `result`, dengan indeks dari elemen list *value* yang dipilih berupa hasil pemanggilan fungsi LCG, dengan umpan berupa indeks iterasi saat itu dan nilai *m* banyak elemen dalam list value dari key karakter tersebut. Jika karakter bukan merupakan alfabet, karakter akan langsung digabung ke dalam string `result`. Setelah iterasi selesai, ditampilkan keluaran hasil string `result`. Program utama ini kemudian diberi nama `leet.py`.

```
from leetdict import d
from prng import LCG


def isAlphabet(c):
    return ((ord(c) >= 65) & (ord(c) <= 90)) | ((ord(c) >= 97) & (ord(c) <= 122))


s = input()
idx = []
result = ""

for i in range(len(s)):
    if isAlphabet(s[i].lower()):
        idx = LCG(i, len(d[s[i].lower()]))
        result += d[s[i].lower()][idx]
    else:
        result += s[i]

print(result)
```

C. Hasil Eksekusi



Gambar 6 Hasil eksekusi program pembangkit tulisan leet speak selama lima kali
(sumber: dokumentasi penulis)



Gambar 7 Hasil eksekusi program pembangkit tulisan leet speak dengan adanya huruf dan simbol lain di dalam masukan
(sumber: dokumentasi penulis)

Dari hasil implementasi, secara keseluruhan, program ini berjalan dengan lancar sesuai rancangan program, dan keluaran

sesuai dengan penulisan *leet speak*. Pada saat pengujian dengan adanya karakter non-alfabet lain dalam tulisan, karakter non-alfabet tersebut juga muncul di keluaran tanpa ada proses lain dalam karakter non-alfabet tersebut.

IV. KESIMPULAN

Matematika diskrit dapat diimplementasikan pada berbagai hal dalam berbagai bidang, termasuk bidang sosial budaya. Gaya penulisan *leet speak* dapat dibangkitkan dengan program yang mengimplementasikan pembangkit bilangan acak semu *Linear Congruential Generator* (LCG) dalam pemilihan kombinasi karakter yang sesuai dengan *leet speak* dari alfabet tersebut secara acak semu, sehingga tulisan yang awalnya menggunakan huruf-huruf alfabet dengan baik dan benar, dapat diubah menjadi kumpulan angka dan simbol dengan bentuk yang mirip dengan huruf-huruf tersebut, atau disebut juga *leet speak*.

V. SARAN

Dengan makalah pembangkitan tulisan *leet speak* ini, diharapkan proses perancangan pembangkitan *leet speak* ini dapat dimanfaatkan dengan baik. Pada saat ini, penulis membuat program pembangkit tulisan *leet speak* dengan mengubah masing-masing huruf dengan kumpulan angka dan simbol yang ekuivalen dengan masing-masing huruf tersebut. Program pembangkit tulisan *leet speak* dapat dikembangkan lebih lanjut dengan turut serta mengubah gaya penulisan kata dan tata kalimat, atau menambah fitur menerjemahkan tulisan *leet speak* menjadi tulisan dengan kaidah penulisan yang baik dan benar. Selain itu, program ini juga dapat dimodifikasi lebih lanjut menjadi program pembangkitan *password* dengan tingkat keamanan tinggi.

VI. UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa karena atas berkat rahmat dan karunia-Nya makalah yang berjudul "Implementasi Pembangkit Bilangan Acak Semu dalam Pembangkitan Tulisan *Leet Speak*" dapat diselesaikan dengan baik. Penulis juga berterima kasih kepada Ibu Nur Ulfa Maulidevi selaku dosen pengampu mata kuliah IF2120 Matematika Diskrit Semester I 2021/2022 Kelas K3.

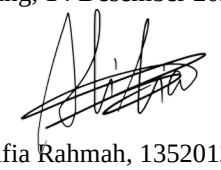
REFERENSI

[1] Munir, Rinaldi. 2015. *Teori Bilangan*. Bahan kuliah IF2120 Matematika Diskrit Semester I Tahun 2021/2022, Institut Teknologi Bandung.
[2] Munir, Rinaldi. 2015. *Pembangkit Bilangan Acak*. Bahan kuliah IF3058 Kriptografi Semester I Tahun 2021/2022, Institut Teknologi Bandung.
[3] Tim Pengajar IF2110/IF2111 Algoritma dan Struktur Data. 2021. *ADT Map/Associative Array*. Bahan kuliah IF2110/2111 Algoritma dan Struktur Data Semester I Tahun 2021/2022, Institut Teknologi Bandung.
[4] Wengrow, Jay. 2020. *A Common-Sense Guide to Data Structures and Algorithms, second edition: Level Up Your Core Programming Skills*. Pragmatic Bookshelf.
[5] McFadden, Christopher. "Leetspeak 101: What Exactly Is It?". https://interestingengineering.com/leetspeak-101-what-exactly-is-it (diakses pada tanggal 10 Desember 2021).
[6] Collections - Container datatypes, Python 3.10.1 documentation. https://docs.python.org/3/library/collections.html (diakses pada tanggal 11 Desember 2021)

PERNYATAAN

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

Bandung, 14 Desember 2021



Alifia Rahmah, 13520122