

# Analisis Penerapan Kombinatorial dalam Menentukan Kekuatan Suatu Password

Vanson Kurnialim - 13522049<sup>1</sup>  
Program Studi Teknik Informatika  
Sekolah Teknik Elektro dan Informatika  
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia  
<sup>1</sup>13522049@mahasiswa.itb.ac.id

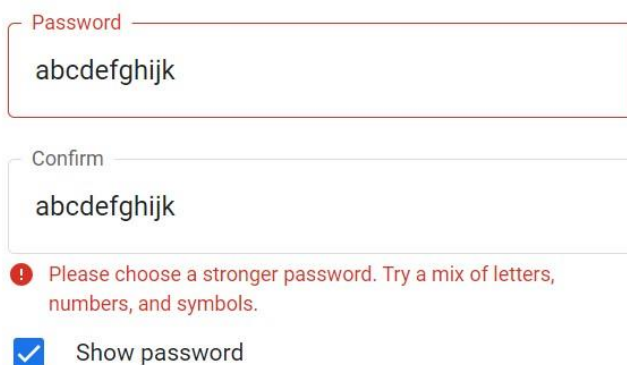
**Abstract**—Password entropy salah satu cara untuk menghitung tingkat kompleksitas suatu password dengan memanfaatkan konsep kombinatorial untuk menghitung banyak kemungkinan password. Bits dari password entropy akan digunakan untuk menentukan kekuatan dari suatu password dan juga waktu yang diperlukan untuk menebak password tersebut.

**Keywords**—Password Entropy, Kompleksitas, Bits Entropy, Kekuatan Password.

## I. LATAR BELAKANG

Password atau kata sandi adalah salah satu metode keamanan yang digunakan untuk memverifikasi validitas identitas pengguna pada suatu sistem keamanan, biasanya dalam suatu jaringan atau sistem operasi. Dalam zaman modern ini, password merupakan hal yang tidak asing lagi dan sudah menjadi bagian dalam kehidupan sehari-hari. Sebuah password dikatakan baik jika password tersebut tidak mudah ditebak atau ditemukan oleh orang lain. Kesulitan penebakan suatu kata sandi dapat ditemukan berdasarkan kompleksitas password yang digunakan.

Kompleksitas password ditentukan dari berbagai faktor seperti panjang password, kombinasi susunan karakter yang membentuk password, dan lainnya. Pada penerapannya, website seringkali menolak password yang ingin kita daftarkan dengan alasan password kurang kuat dan memberikan pesan untuk memasukkan password yang lebih kuat. Hal ini berarti sandi yang kita masukkan kurang kompleks atau bisa dibilang terlalu sederhana.



Password  
abcdefghijk

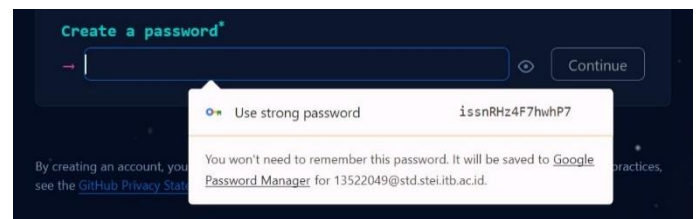
Confirm  
abcdefghijk

⚠ Please choose a stronger password. Try a mix of letters, numbers, and symbols.

Show password

**Gambar 1.** Password kurang kuat saat membuat akun  
Sumber : Dokumentasi Pribadi

Mengambil contoh pembuatan akun google, biasanya akan muncul opsi untuk menggunakan password yang disarankan oleh google. Google akan secara otomatis men-*generate* sebuah password yang memiliki kompleksitas yang masuk ke dalam kategori *strong* password.



**Gambar 2.** Suggested Password saat membuat akun  
Sumber : Dokumentasi Pribadi

## II. LANDASAN TEORI

### A. Kombinatorial

Kombinatorial adalah salah satu cabang ilmu matematika yang membahas terkait data diskrit yang dapat dihitung. Kombinatorial sangat berperan dalam kehidupan kita dan juga dimanfaatkan di berbagai bidang ilmu lain seperti IT, fisika, dan lainnya. Pada dasarnya, kombinatorial membantu kita dalam menghitung berapa banyak kejadian yang mungkin untuk suatu permasalahan tanpa perlu mengenumerasikan semua kemungkinannya.

Terdapat 2 aturan dasar atau kaidah menghitung hasil dari kombinatorial.

#### 1) Kaidah perkalian (*rule of product*)

Jika suatu permasalahan dapat diselesaikan dalam  $n$  dan  $m$  cara, maka banyaknya cara untuk menyelesaikan permasalahan tersebut ada sebanyak  $(n \times m)$  cara.

#### 2) Kaidah penjumlahan (*rule of sum*)

Jika suatu permasalahan dapat diselesaikan dalam  $n$  atau  $m$  cara, maka banyaknya cara untuk menyelesaikan permasalahan tersebut ada sebanyak  $(n + m)$  cara.

Sebagai contoh, ada berapa banyak kemungkinan pembentukan password 1 digit yang terdiri dari angka 0..9? Tentu saja passwordnya hanya dapat terdiri dari 0 atau 1 atau 2 sampai 9. Maka total kemungkinan adalah  $1+1+1+1+1+1+1+1+1+1 = 10$  kemungkinan.

Selain kaidah dasar yang dijelaskan di atas, kombinatorial juga menggunakan beberapa prinsip perhitungan yang lain :

1) Inklusi-Eksklusi

Melihat kembali aturan kaidah penjumlahan, jika terdapat lebih dari satu cara untuk mencari total cara menyelesaikan permasalahan, kaidah perkalian digunakan dengan asumsi tidak ada kesamaan di dalam cara-cara yang diperhitungkan. Karena jika terdapat kesamaan, kesamaan tersebut akan terhitung dua kali sehingga hasil akan salah. Dengan menggunakan kaidah inklusi-eksklusi kita dapat menghitung cara yang unik saja. Berikut adalah rumusan mencari total cara untuk cara A1 dan cara A2 yang memiliki kesamaan :

$$|A1 \cup A2| = |A1| + |A2| - |A1 \cap A2|$$

Sebagai contoh, ada berapa banyak *binary string* dengan panjang 8 yang diawali dengan '11' dan diakhiri dengan '11'? Banyaknya *string* yang diawali dengan '11' sama dengan banyaknya *string* yang diakhiri dengan '11' yaitu  $1 \times 2^6 = 64$ . Namun jika kita langsung mengaplikasikan kaidah penjumlahan, hasil yang didapat,  $64 + 64 = 128$ , akan salah. Kita harus mengurangi hasil tersebut dengan kasus yang memenuhi kedua kondisi awalan '11' dan akhiran '11' yaitu  $1 \times 1 \times 2 \times 2 \times 2 \times 2 \times 1 \times 1 = 16$ . Hasil yang benar adalah  $64 + 64 - 16 = 112$  *string*.

2) Permutasi

Permutasi adalah jumlah urutan berbeda dari pengaturan objek-objek. Permutasi merupakan bentuk khusus dari aplikasi kaidah perkalian. Perlu diingat bahwa urutan dalam permutasi adalah penting. Rumus dari permutasi ialah :

$$P(n, r) = \frac{n!}{(n - r)!}$$

3) Kombinasi

Kombinasi adalah bentuk khusus dari permutasi dimana pada permutasi urutan kemunculan diperhitungkan, sedangkan pada kombinasi urutan kemunculan diabaikan. Secara umum, jumlah cara  $r$  yang diambil dari  $n$  adalah sebagai berikut :

$$C(n, r) = \frac{n!}{r! (n - r)!}$$

4) Permutasi dan Kombinasi Bentuk Umum

Pada kasus ini, permutasi dan kombinasi memiliki rumusan yang sama. Direpresentasikan sebagai  $n$  buah bola yang tidak seluruhnya berbeda warna. Jika bola seluruhnya berbeda, rumusan pengaturan  $n$  buah bola ke dalam  $n$  buah kotak adalah  $n!$ . Maka rumus utama adalah sebagai berikut :

$$P(n; n_1, n_2, \dots, n_n) = \frac{n!}{n_1! n_2! \dots n_n!}$$

5) Kombinasi dengan Pengulangan

Kasus khusus dari kombinasi biasa dimana untuk setiap perhitungan cara  $n$  dapat memiliki lebih dari satu kombinasi benda atau objek  $r$ . Berikut adalah rumus dari kombinasi tersebut :

$$C(n + r - 1, r) = C(n + r - 1, n - 1)$$

B. Password Entropy

Password entropy adalah ukuran seberapa acak sebuah password atau seberapa sulit suatu password untuk ditebak. Perhitungan ini digunakan untuk menentukan kekuatan suatu password dengan memanfaatkan bidang ilmu kombinatorial. Password entropy sangat bermanfaat dan bahkan sudah tidak dapat lepas lagi dari sistem password kita.

Konsep dasar dari password entropy cukup mudah. Menggunakan kombinatorial, kita dapat menghitung total kemungkinan suatu password berdasarkan aspek-aspek atau parameter dari password itu sendiri. Hasil dari kombinatorial tersebut lalu akan diubah menjadi satuan password entropy, yaitu bits. Nilai dari bits inilah yang digunakan untuk menentukan apakah suatu password mudah atau sulit untuk ditebak oleh orang lain.

Berikut adalah rumus dari password entropy :

$$E = \log_2(R^L)$$

E pada rumus adalah nilai password entropy dengan satuan bits, R adalah banyak kemungkinan untuk karakter di dalam password, dan L adalah panjang dari password itu sendiri.

Dapat dikatakan password entropy adalah ukuran kesulitan bagi seorang *hacker* untuk mendapatkan password pengguna. *Hacker* berkemungkinan mencoba menebak password pengguna dengan sandi-sandi umum seperti 'password', melakukan riset terhadap pribadi pengguna misal sosial media ataupun rekam jejak digital, menggunakan program untuk mencoba seluruh kemungkinan password yang didasari oleh probabilitas matematis.

Sebuah password terdiri dari sangat banyak kemungkinan karakter pembentuk. Namun secara umum, karakter pembentuk yang mungkin dibatasi dan dikategori sebagai huruf kecil, huruf besar, angka, dan simbol. Berikut adalah informasi terkait kompleksitas dari tiap kategori :

Charset	a-z	a-z, A-Z	a-z, A-Z, 0-9	a-z, A-Z, 0-9, symbols	Full UTF-8	Dictionary
Charset size	26	52	62	95	137,000	171,476 words
Chars/ words	Bits of entropy					
4	19	23	24	26	68	70
6	28	34	36	39	102	104
8	38	46	48	53	137	139
12	56	68	71	79	205	209
16	75	91	95	105	273	278
32	150	182	191	210	546	556
60	282	342	357	394	1,024	1,043

Gambar 3. Tabel bits of entropy untuk beberapa kategori karakter

Sumber : <https://cloud.google.com/solutions/modern-password-security-for-users.pdf>

Terlihat pada tabel di atas, bits entropy untuk beragam

panjang password dan banyak kemungkinan karakter pembentuk. Namun bagaimana kita bisa mengklasifikasikan kompleksitas mana yang tepat untuk password kita?

Bits of Entropy	Password Entropy
0 – 35	Very Weak
36 – 59	Weak
60 – 119	Strong
≥ 120	Very Strong

**Tabel 1.** Kategori kekuatan password berdasarkan bits entropy  
Sumber : Dokumentasi Pribadi

Password dengan klasifikasi *very weak* secara umum tidak baik digunakan untuk hal-hal penting. Sedangkan, password *weak* mungkin cocok digunakan untuk sandi *login* akun sekolah ataupun akun lainnya yang tidak terlalu krusial. Password dengan kekuatan *strong* merupakan pilihan yang tepat untuk menjaga informasi finansial ataupun informasi penting lainnya. Walaupun password dengan klasifikasi *very strong* akan sangat sulit untuk ditebak, dalam penggunaannya akan kurang sesuai pada sebagian besar situasi dikarenakan password yang terlalu panjang dan juga kesulitan pengguna untuk mengisi password itu sendiri.

Bagi kita manusia, akan cukup sulit untuk membuat password dengan kompleksitas tinggi atau menentukan tingkatan kompleksitas dari password yang kita buat. Maka dari itu, kita membutuhkan bantuan dari teknologi atau program seperti *password manager* dan juga program lainnya. Ada beberapa metode yang bisa kita gunakan untuk meningkatkan kompleksitas password seperti men-substitusi karakter dengan simbol, *me-mirror* password, dan menggunakan sebanyak mungkin variasi karakter dari berbagi kategori namun masih nyaman digunakan.

Untuk menekankan pentingnya membuat password dengan kompleksitas baik, berikut adalah gambaran waktu yang diperlukan untuk menebak sebuah password oleh komputer dengan metode *brute force* dengan asumsi 10 juta tebakan tiap detik :

Charset	a-z	a-z, A-Z	a-z, A-Z, 0-9	a-z, A-Z, 0-9, symbols	Full UTF-8	Dictionary
Charset size	26	52	62	95	137,000	171,476 words
Chars/ words	Max time to crack @ 10,000,000 hash/sec					
4	< 1 sec	< 1 sec	1.5 sec	8.1 sec	1.1 million years	2.7 million years
6	30.9 sec	33 min	1.6 hours	20.4 hours	21 trillion years	> universe lifespan
8	5.8 hours	2 months	8.3 months	21 years	> universe lifespan	> universe lifespan
12	302.6 years	1.2 million years	10 million years	1.7 billion years	> universe lifespan	> universe lifespan
16	138 million years	9 trillion years	151 trillion years	> universe lifespan	> universe lifespan	> universe lifespan
32	> universe lifespan	> universe lifespan	> universe lifespan	> universe lifespan	> universe lifespan	> universe lifespan
60	> universe lifespan	> universe lifespan	> universe lifespan	> universe lifespan	> universe lifespan	> universe lifespan

**Gambar 4.** Tabel waktu untuk menebak password dari beberapa kategori karakter

Sumber : <https://cloud.google.com/solutions/modern-password-security-for-users.pdf>

### III. ANALISIS DAN IMPLEMENTASI

#### A. Kombinasi Password

Penerapan konsep kombinatorial untuk menentukan kekuatan

suatu password berada pada bagian perhitungan banyaknya kemungkinan suatu password berdasarkan atribut-atribut dari password itu sendiri. Sebuah password dapat terdiri seluruhnya dari huruf kecil, namun juga dapat berupa gabungan dari karakter-karakter dari kategori lain seperti huruf besar, angka, maupun simbol. Hal yang perlu kita ketahui sebelum mencoba menentukan banyak kemungkinan password adalah menentukan banyak jumlah karakter yang mungkin digunakan dalam suatu password.

Seperti yang kita tahu, alfabet terdiri dari 26 huruf. Maka keseluruhan karakter huruf kecil ada sebanyak 26, begitu juga dengan karakter huruf besar. Selain itu, password juga dapat terdiri dari angka. Banyaknya angka yang mungkin adalah 10, yaitu angka 0 sampai 9. Sebuah password sering kali juga memperbolehkan penggunaan simbol-simbol seperti &, @, \*, dan lainnya. Jumlah tepat dari karakter simbol berubah-ubah berdasarkan simbol apa saja yang dapat diterima oleh peminta password. Namun pada implementasi ini, kita asumsikan karakter simbol yang dapat digunakan sebanyak 33 karakter.

Banyak jumlah karakter yang mungkin digunakan bergantung pada kelompok karakter yang memang digunakan pada password tersebut. Kita tidak mengikutsertakan kelompok karakter yang tidak ada dalam password agar hasil banyak password lebih akurat. Misal terdapat password berisi 'kucing'. Password tersebut hanya menggunakan karakter huruf kecil, maka jumlah karakter yang mungkin digunakan adalah 26 karakter. Jika password berisi 'ITB123', maka jumlah karakter yang mungkin adalah jumlah huruf besar ditambah jumlah angka,  $26 + 10 = 36$  karakter.

Jika password hanya berupa 1 digit, maka banyak password yang mungkin sama dengan banyak karakter yang mungkin. Jika panjang password adalah 5 dan banyak karakter yang mungkin adalah 62, maka menggunakan *rule of product*, banyak password yang mungkin dapat dihitung sebagai berikut :

$$\begin{aligned} \text{-----} &= 36 \times 36 \times 36 \times 36 \times 36 \\ 36^5 &= 60466176 \text{ kemungkinan} \end{aligned}$$

Mengapa kita tidak menggunakan kemungkinan karakter spesifik terhadap posisi dia digunakan? Misal password berupa "ITB59", maka banyak password yang mungkin berupa dihitung seperti berikut :

$$\begin{aligned} \text{-----} &= 26 \times 26 \times 26 \times 10 \times 10 \\ \text{total} &= 1757600 \text{ kemungkinan} \end{aligned}$$

Perhitungan dengan cara seperti ini dapat dilakukan dengan anggapan kita mengetahui banyak karakter mungkin untuk tepat semua posisi password. Hasil yang didapatkan bukanlah banyak password seperti yang kita inginkan, melainkan banyak password dengan kemungkinan perubahan pada tiap digitnya, karakter tidak dapat berpindah posisi. Sedangkan hasil banyak password yang kita inginkan melingkupi segala kemungkinan termasuk karakter yang berada di posisi lain.

Dikarenakan kemungkinan karakter untuk tiap digit adalah sama, yaitu jumlah karakter untuk tiap kategori, maka rumus untuk menentukan banyak password dapat kita sederhanakan menjadi :

$$\text{banyak password} = \text{banyak karakter}^{\text{panjang password}}$$

Hasil kemungkinan inilah yang akan digunakan pada rumusan *password entropy* untuk menentukan bits password. Pada dasarnya, rumusan *entropy* tersebut hanya berguna untuk menseederhanakan hasil banyak kemungkinan password sehingga lebih mudah untuk kita kategorikan kepada klasifikasi kekuatan password.

### B. Penerapan Kode

Berikut adalah penerapan konsep kombinatorial di atas dalam kode *python* :

```

import numpy as np
password = input(str("Input password : "))
L = len(password)

hurufKecil = False
hurufBesar = False
angka = False
simbol = False

```

**Gambar 5.** Source code (bagian 1)

Sumber : Dokumentasi Pribadi

Kode memanfaatkan library *numpy* untuk perhitungan matematis dalam program. Program akan meminta input sebuah password dalam bentuk string yang dapat berupa apa saja. Lalu akan melakukan inisialisasi pada variabel *L* berupa panjang password dan juga memasang nilai *boolean false* pada 4 variabel tersebut.

```

for i in range(L) :
    if (ord(password[i]) >= 97 and
ord(password[i]) <= 122) :
        hurufKecil = True
    elif (ord(password[i]) >= 65 and
ord(password[i]) <= 90) :
        hurufBesar = True
    elif (ord(password[i]) >= 48 and
ord(password[i]) <= 57) :
        angka = True
    else :
        simbol = True

R = 0
if (hurufKecil) : R += 26
if (hurufBesar) : R += 26
if (angka) : R += 10
if (simbol) : R += 33

```

**Gambar 6.** Source code (bagian 2)

Akan dilakukan iterasi untuk tiap karakter pada password untuk mengecek karakter tersebut termasuk dalam kategori apa. Pada program ini, hanya terdapat 4 kategori karakter yaitu huruf kecil, huruf besar, angka, dan juga simbol.

```

kombinasi = float(pow(R,L))
E = np.round(np.log2(kombinasi))

if (E >= 0 and E <= 35) :
    print("Your Password is" + "\033[91m
{}\033[00m" .format("Very Weak!"))
elif (E >= 36 and E <= 59) :
    print("Your Password is" + "\033[91m
{}\033[00m" .format("Weak!"))
elif (E >= 60 and E <= 119) :
    print("Your Password is" + "\033[92m
{}\033[00m" .format("Strong!"))
elif (E >= 120) :
    print("Your Password is" + "\033[92m
{}\033[00m" .format("Very Strong!"))
else :
    print("\033[91m {} \033[00m"
.format("error!"))

```

**Gambar 7.** Source code (bagian 3)

Variabel *kombinasi* akan menyimpan banyak password yang mungkin berdasarkan rumus yang telah diturunkan di atas. Lalu variabel *E* akan mencari bits dari password *entropy*. Pesan kekuatan password akan ditampilkan pada layer dengan format pewarnaan sesuai dengan kategori kekuatan yang telah dicari.

```

waktu = kombinasi / 10000000

if (waktu < 1) :
    print("it will take less than 1 seconds to
guess!")
elif (waktu < 60) :
    print("it will take" ,
"{:.1f}".format(waktu), "seconds to guess!")
else :
    waktu /= 60
    if (waktu < 60) :
        print("it will take" ,
"{:.1f}".format(waktu), "minutes to guess!")
    else :
        waktu /= 60
        if (waktu < 24) :
            print("it will take" ,
"{:.1f}".format(waktu), "hours to guess!")
        else :
            waktu /= 24
            if (waktu < 31) :
                print("it will take" ,
"{:.1f}".format(waktu), "days to guess!")
            else :
                waktu /= 31
                if (waktu < 12) :
                    print("it will take" ,
"{:.1f}".format(waktu), "months to guess!")
                else :
                    waktu /= 12
                    if (waktu < 1000) :
                        print("it will take" ,
"{:.1f}".format(waktu), "years to guess!")
                    else :
                        waktu /= 1000
                        if (waktu < 1000) :
                            print("it will
take" , "{:.1f}".format(waktu), "thousand years
to guess!")
                        else :
                            waktu /= 1000
                            if (waktu < 1000) :
                                print("it will
take" , "{:.1f}".format(waktu), "million years
to guess!")

```

```

else :
    waktu /= 1000
    if (waktu <
1000) :
        print("it
will take" , "{:.1f}".format(waktu), "billion
years to guess!")
    else :
print("it will take" + "\033[93m {} \033[00m"
.format("forever") + " to guess!")
print("\n")

```

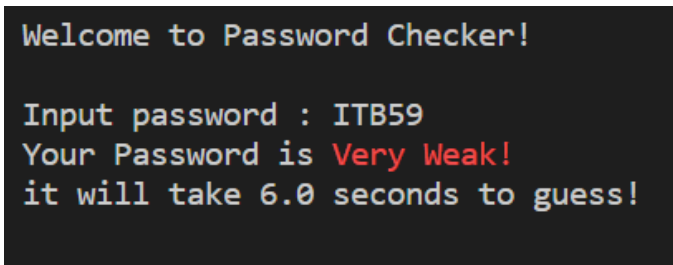
**Gambar 8.** Source code (bagian 4)

Tambahan kode tersebut berfungsi untuk menunjukkan berapa lama waktu yang dibutuhkan untuk menebak password. Mekanisme penebakan diasumsikan dilakukan oleh komputer dengan kecepatan penebakan 10 juta / detik. Dengan tampilan waktu seperti ini, akan lebih menekankan seberapa kuat password yang diinput.

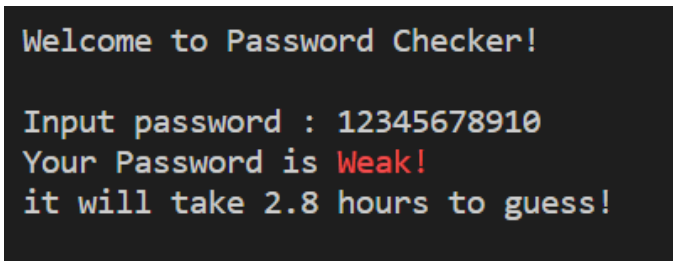
*C. Hasil Implementasi Program*

Berikut adalah beberapa hasil pengecekan kekuatan password menggunakan program yang telah dibuat :

- 1) Kompleksitas rendah

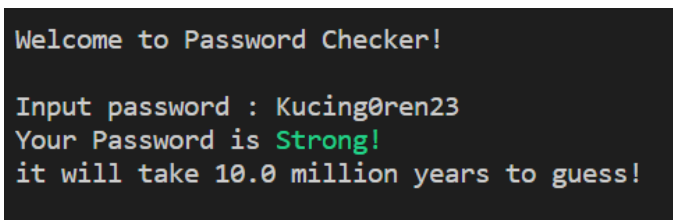


**Gambar 9.** Hasil program 1  
Sumber : Dokumentasi Pribadi

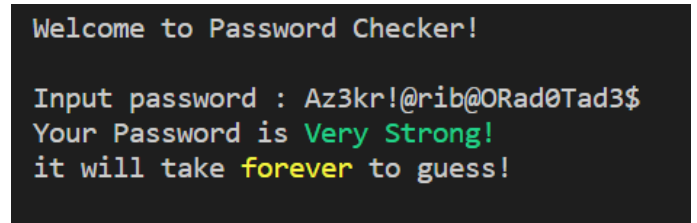


**Gambar 10.** Hasil program 2  
Sumber : Dokumentasi Pribadi

- 2) Kompleksitas tinggi



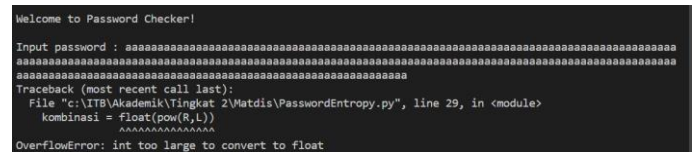
**Gambar 11.** Hasil program 3  
Sumber : Dokumentasi Pribadi



**Gambar 12.** Hasil program 4  
Sumber : Dokumentasi Pribadi

Program tersebut dapat berjalan dengan baik hingga kapasitas password yang cukup panjang. Dapat dilihat untuk tiap kategori kekuatan password, waktu yang dibutuhkan untuk menebak password akan melompat jauh sekali. Hal ini terjadi karena lama waktu bergantung kepada kompleksitas atau banyak kombinasi kemungkinan password. Sedangkan untuk tiap digit dari password, kompleksitas akan berkembang secara eksponen sehingga waktu penebakan juga akan bertambah secara eksponen.

Namun jika diberikan masukan password yang terlalu panjang, dikarenakan maksimum kapabilitas komputer, hasilnya akan *error* karena *overflow*.



**Gambar 913.** Program *overflow*  
Sumber : Dokumentasi Pribadi

Namun tentu saja pada kenyatannya, tidak ada password dengan panjang seperti pada contoh di atas.

IV. KESIMPULAN

Kombinatorial merupakan bagian ilmu matematis yang sangat bermanfaat. Salah satu pemanfaatan teori kombinatorial adalah dalam perhitungan kompleksitas password yang sekarang digunakan di segala sistem keamanan di dunia. Berdasarkan hasil kombinatorial password, kita dapat menemukan bits entropy yang menunjukkan tingkat kompleksitas password kita. Semakin kompleks password yang kita miliki, maka akan semakin sulit data kita untuk ditembus atau dengan kata lain, lebih aman. Kita juga dapat menghitung kekuatan password kita sendiri dengan mengikuti rumusan dalam makalah ini ataupun menggunakan program yang telah dipaparkan.

V. PENUTUP

Terima kasih dan puji syukur penulis haturkan sebesar-besarnya kepada Tuhan yang mahakuasa, atas berkat dan rahmatnya, penulis diperbolehkan untuk menyelesaikan makalah ini tepat waktu dan dengan baik. Penulis juga mengucapkan terimakasih kepada dosen mata kuliah Matematika Diskrit, Dr. Nur Ulfa Maulidevi, S. T, M. Sc., atas bimbingan dan pengajarannya selama di kelas. Tidak lupa

penulis juga berterimakasih kepada dosen Dr. Ir. Rinaldi Munir, M. T., karena telah memfasilitasi penulis dengan *website* berisi materi-materi sehingga dapat dijadikan bahan pembelajaran. Terima kasih juga kepada seluruh pihak yang karyanya penulis jadikan referensi dalam pembuatan makalah ini.

Penulis mengucapkan permintaan maaf apabila ada kesalahan dalam penulisan makalah ini.

#### REFERENCES

- [1] M. Dr. Ir. Rinaldi Munir. “Kombinatorial (Bagian 1)”. <https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/17-Kombinatorial-Bagian1-2023.pdf> (diakses pada 8 Desember 2023).
- [2] M. Dr. Ir. Rinaldi Munir. “Kombinatorial (Bagian 2)”. <https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/18-Kombinatorial-Bagian2-2023.pdf> (diakses pada 8 Desember 2023).
- [3] Google. “modern password security for users”. <https://cloud.google.com/solutions/modern-password-security-for-users.pdf> (diakses pada tanggal 9 Desember 2023).
- [4] Pleacher, David. “Calculating Password Entropy”. <https://www.pleacher.com/mp/mlessons/algebra/entropy.html> (diakses pada tanggal 9 Desember 2023).
- [5] NordVPN. “Password entropy and how to calculate it”. <https://nordvpn.com/blog/what-is-password-entropy/> (diakses pada tanggal 9 Desember 2023).
- [6] Okta. “Password Entropy: The Value of Unpredictable Passwords”. <https://www.okta.com/identity-101/password-entropy/> (diakses pada tanggal 8 Desember 2023).
- [7] GeeksforGeeks. “Print Colors in Python Terminal”. <https://www.geeksforgeeks.org/print-colors-python-terminal/> (diakses pada tanggal 9 Desember 2023).

#### 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, 9 Desember 2023



Vanson Kurnialim  
13522049