JURNAL INFORMATIKA
Volume 3, Nomor 2, September 2024 ISSN 2809-3704

Aplikasi Pengukuran Kekuatan dan Rekomendasi Password Berdasarkan Input Pengguna dengan Metode Entropi

Bacilius Agung Suburdjati[1]*, Fransiska Nindita Iswari[2], Wilson Kenneth Jonathan[3]

[1]Program Studi Bisnis Digital, Universitas Santo Borromeus
[2,3]Sekolah Menengah Atas Santa Angela Bandung
Email: basnagung@gmail.com[1], fransiskaisw@gmail.com[2], wilsonkjonathan@gmail.com[3]

ABSTRAK - Penelitian ini mengembangkan aplikasi berbasis Python untuk mengevaluasi kekuatan *password* serta memberikan rekomendasi *password* yang lebih aman menggunakan alfabet fonetik NATO, *True Random Number Generator* (TRNG), dan *Pseudorandom Number Generator* (PRNG). Aplikasi ini bertujuan meningkatkan keamanan pengguna dengan menghasilkan *password* yang memiliki entropi tinggi. Entropi ini dipengaruhi oleh variasi karakter, panjang *password*, serta kombinasi simbol, angka, dan huruf. Studi ini menunjukkan bahwa semakin kompleks suatu *password*, semakin lama waktu yang dibutuhkan untuk peretasan. Namun, *password* yang terlalu kompleks sering kali sulit diingat oleh pengguna, sehingga diperlukan solusi yang tetap mempertahankan keamanan tanpa mengurangi aspek kemudahan penggunaan. Aplikasi yang dikembangkan tidak hanya mengukur kekuatan *password*, tetapi juga memberikan rekomendasi *password* yang lebih kuat melalui tiga metode. Metode pertama menggunakan alfabet fonetik NATO untuk menggantikan beberapa karakter dalam *password*, membuatnya lebih mudah diingat. Metode kedua menambahkan karakter acak berbasis PRNG, sementara metode ketiga menggunakan kombinasi TRNG dan fonetik NATO untuk meningkatkan kompleksitas *password*. Dengan pendekatan ini, aplikasi memberikan solusi praktis dalam meningkatkan keamanan digital, membantu pengguna menciptakan *password* yang lebih aman, namun tetap mudah digunakan dalam kehidupan sehari-hari.

Kata Kunci: *Password*, *Entropy*, *True Random Number Generator*, *Pseudorandom Number Generator*, Alfabet Fonetik NATO.

ABSTRACT - *This study develops a Python-based application to evaluate password strength and provide more secure password recommendations using the NATO phonetic alphabet, True Random Number Generator (TRNG), and Pseudorandom Number Generator (PRNG). The application aims to enhance user security by generating passwords with high entropy. This entropy is influenced by character variation, password length, and the combination of symbols, numbers, and letters. The study shows that the more complex a password is, the longer it takes to be cracked. However, overly complex passwords are often difficult to remember, requiring a solution that maintains security without compromising usability. The developed application not only measures password strength but also provides stronger password recommendations through three methods. The first method replaces some password characters with the NATO phonetic alphabet, making them easier to remember. The second method adds random characters based on PRNG, while the third method combines TRNG and NATO to enhance password complexity. With this approach, the application offers a practical solution to improving digital security, helping users create stronger passwords that remain easy to use in daily life.*

Keywords: *Password, Entropy, True Random Number Generator, Pseudorandom Number Generator, NATO Phonetic Alphabet.*

PENDAHULUAN

Dalam dunia digital saat ini, *password* memainkan peran yang sangat penting sebagai kunci utama dalam menjaga keamanan data pribadi dan informasi sensitif. *Password* yang kuat dapat melindungi berbagai aset seperti akun bank, media sosial, dan sistem informasi pribadi lainnya dari akses yang tidak sah. Tanpa perlindungan *password*, data seseorang dapat dengan mudah diakses oleh pihak yang tidak berwenang, yang berpotensi menyebabkan pelanggaran privasi dan kerugian. Seiring berkembangnya teknologi, serangan siber semakin canggih dalam menebak kata sandi. Oleh karena itu, pengguna dituntut untuk membuat *password* yang lebih kompleks menggunakan kombinasi berbagai karakter seperti huruf besar, huruf kecil, angka, dan simbol. Namun, metode yang ada saat ini, seperti *password*

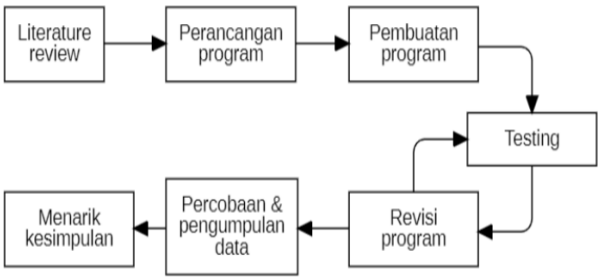
manager atau generator *password* acak, masih memiliki beberapa keterbatasan. Banyak generator hanya menghasilkan *password* acak tanpa mempertimbangkan kemudahan pengguna dalam mengingatnya, sehingga pengguna sering kali menyimpan *password* di tempat yang tidak aman atau menggunakan kembali *password* lama mereka.

Beberapa penelitian sebelumnya telah mengusulkan penggunaan *password* manager atau sistem autentikasi berbasis enkripsi untuk meningkatkan keamanan *password*. Namun, penelitian-penelitian tersebut umumnya berfokus pada kekuatan *password* tanpa memperhitungkan aspek usability atau kemudahan pengguna dalam mengingat *password* yang dihasilkan. Gap ini menjadi masalah yang perlu diatasi, karena solusi keamanan yang tidak mempertimbangkan kenyamanan pengguna dapat mengarah pada kebiasaan buruk, seperti penggunaan *password* yang lemah atau penyimpanan *password* secara tidak aman. Penelitian ini mengembangkan solusi yang lebih seimbang antara keamanan dan kemudahan penggunaan dengan menggunakan metode alfabet fonetik NATO, TRNG, PRNG. Pendekatan ini bertujuan untuk menghasilkan *password* yang memiliki tingkat entropi tinggi, sehingga lebih sulit diretas, tetapi tetap mudah diingat oleh pengguna.

Berdasarkan permasalahan tersebut, penelitian ini bertujuan untuk menjawab beberapa pertanyaan utama, yaitu: bagaimana mengukur kompleksitas *password* secara efektif; metode apa yang dapat digunakan untuk merekomendasikan *password* yang aman tetapi tetap mudah diingat oleh pengguna; serta apakah rekomendasi *password* yang dihasilkan telah terbukti lebih aman dari serangan siber. Dengan pendekatan ini, penelitian diharapkan dapat memberikan solusi praktis dalam meningkatkan keamanan digital tanpa mengorbankan aspek kemudahan penggunaan di era modern.

METODE

Penelitian ini dilakukan melalui beberapa tahap, yaitu: literature review, perancangan program, pembuatan program, testing, revisi program, percobaan dan pengumpulan data, serta menarik kesimpulan.



Gambar 1. Diagram Alur Tahap Penelitian

1. Literature Review
   Tahap pengumpulan artikel, jurnal atau penelitian lain yang berhubungan bahan penelitian berkaitan dengan pengukuran kekuatan dan rekomendasi *password*. Tahap ini digunakan untuk melengkapi studi pustaka, membantu perancangan serta pembuatan program aplikasi, dan untuk membandingkan penelitian sendiri dengan penelitian lain yang sejenis.
2. Perancangan Program
   Program dirancang sesuai dengan keinginan yang dibutuhkan berdasarkan pengumpulan data dari literature review. Tujuan tahap ini yaitu untuk mengatur struktur program secara keseluruhan dengan merincikan algoritma program dan menentukan IDE.
3. Pembuatan Kode Program
   Konsep dan algoritma yang telah dirancang diimplementasikan ke dalam bentuk program menggunakan bahasa Python dan IDE PyCharm. Program dimulai dengan adanya input *password* pengguna (variabel bebas). Lalu, dilanjutkan dengan proses pengukuran kekuatan *password* dan metode rekomendasi. Hingga pada akhirnya, dikeluarkan output (variabel terikat) berupa skala kekuatan *password* dan atau rekomendasi *password*.
   a. Pengukuran Kekuatan *Password*
      Pengukuran kekuatan *Password* dilakukan dengan menghitung entropi untuk mengukur tingkat ketidakpastian atau kerandoman *password*, yang berbanding lurus dengan tingkat kesulitannya untuk diretas. Rumus entropi yang digunakan dalam penelitian ini adalah sebagai berikut:
      H = log2 (NL)
      Dimana:
      H = Entropi *password* (dalam bit)
      N = Ukuran karakter set yang digunakan dalam *password*
      L = Panjang *password*

Ukuran karakter set (N) dihitung berdasarkan karakter yang terdapat dalam *password*: huruf besar (26 karakter), huruf kecil (26 karakter), angka (10 karakter) dan simbol (32 karakter). Sebagai contoh, jika *password* terdiri dari 10 karakter yang mengandung huruf besar, huruf kecil, angka, dan simbol, maka:
N = 26+26+10+32 = 94
H = log2(9410) = 65.85bit
Semakin tinggi nilai entropi, semakin sulit *password* untuk ditebak oleh serangan *brute-force*.

b. Rekomendasi *Password*
Tujuan rekomendasi *password* untuk menghasilkan karakter tambahan dengan pendekatan deterministik yang tetap memenuhi aspek keamanan. Rekomendasi dihasilkan dengan mengIntegrasikan PRNG dan TRNG. PRNG dalam penelitian ini menggunakan pustaka secrets di Python, yang lebih aman dibandingkan random karena lebih sulit diprediksi. Langkah-langkah PRNG dalam rekomendasi *password*:
1) Mengambil *password* awal yang diberikan oleh pengguna
2) Menambahkan karakter acak dari setiap kategori karakter (huruf besar, huruf kecil, angka, dan simbol) untuk meningkatkan kompleksitas
3) *Password* yang telah diperkuat diuji kembali dengan perhitungan entropi untuk memastikan peningkatan keamanan.

TRNG dalam penelitian ini menggunakan metode os.urandom(), yang menghasilkan byte acak berbasis sumber keacakan sistem operasi. Langkah-langkah TRNG dalam rekomendasi *password*:
1) Mengambil *password* awal yang diberikan oleh pengguna.
2) Mengganti beberapa karakter dalam *password* dengan padanan fonetik NATO untuk meningkatkan keterbacaan tanpa mengurangi keamanan.
3) Menambahkan dua karakter acak berbasis TRNG untuk meningkatkan entropi tanpa membuat *password* sulit diingat.
4) *Password* hasil akhir diuji kembali menggunakan perhitungan entropi untuk memastikan peningkatan keamanan.

4. Testing Program
Testing program merupakan tahap untuk memastikan bahwa program dapat berjalan sesuai dengan prediksi atau ketentuan yang diinginkan. Tahap ini dapat dilakukan berulang kali dan digunakan untuk memvalidasi kesesuaian program yang dibentuk dengan yang sudah dirancang sebelumnya.
5. Revisi Program
Tahap ini merupakan kelanjutan dari tahap testing. Jika ada bagian program yang tidak sesuai pada tahap testing, maka permasalahan akan dicari dan program akan direvisi. Lalu, ketika revisi sudah selesai, maka akan kembali ke tahap testing hingga program benar-benar sesuai dengan perancangan di awal dan keinginan pada penelitian.
6. Percobaan dan Pengumpulan Data
Ketika tahap revisi sudah selesai, maka akan dilakukan percobaan berupa input tiga *password* yang berbeda dalam program. Masing-masing *password* akan dilakukan percobaan 3 kali untuk mendapatkan rekomendasi *password* yang berbeda. Dalam percobaan ini, maka akan terdapat output kekuatan, entropi, dan rekomendasi *password*. Data percobaan akan dikumpulkan dalam bentuk tabel agar mudah untuk dibaca dan dimengerti.
7. Menarik Kesimpulan
Rangkuman dan gambaran akhir dari penelitian. Pada tahap ini dicantumkan kelebihan, keunggulan, dan kekurangan dalam program yang telah dibuat dan disertakan jawaban dari perumusan masalah.

HASIL DAN PEMBAHASAN

3.1. Kode Program yang Dibuat

1. Pengukuran Kekuatan *Password*
Program berikut berfungsi menghitung entropi *password* berdasarkan variasi karakter (huruf besar atau kecil, angka, simbol) dan panjang *password*, serta mengklasifikasikan kekuatan *password* berdasarkan nilai entropi.

```python
def password_entropy(password):
    char = 0
    if re.search(r'[A-Z]', password):
        char += 26
    if re.search(r'[a-z]', password):
        char += 26
    if re.search(r'[0-9]', password):
        char += 10
    if re.search(f'[{re.escape(string.punctuation)}]', password):
        char += 32
    password_length = len(password)
    entropy = math.log2(char ** password_length) # Entropy Formula
    return entropy
def password_strength_level(entropy):
    if entropy < 40:
        return "Very Weak"
    elif entropy < 60:
        return "Weak"
    elif entropy < 80:
        return "Moderate"
    elif entropy < 100:
        return "Strong"
    else:
        return "Very Strong"
```

Gambar 2. Blok Pengukuran Kekuatan *Password*

2. Rekomendasi *Password* Berbagai Metode

NATO Phonetic Alphabet

Mengganti karakter dalam *password* dengan representasi alfabet fonetik NATO untuk meningkatkan kompleksitas namun tetap mudah diingat.

```python
def for_measuring_nato_recommendation(password):
    alphabet = string.ascii_letters
    replaceable_indices = [i for i, char in enumerate(password)
                           if char.upper() in alphabet]
    chosen_indices = random.sample(replaceable_indices,
                    min(2, len(replaceable_indices)))
    new_password = list(password)
    for index in chosen_indices:
        char = password[index]
        phonetic = phonetic_alphabet.get(char.upper(), char)
        new_password[index] = phonetic
    recommended_password = ''.join(new_password)
    return recommended_password
```

Gambar 3. Blok Rekomendasi *Password* dengan NATO Phonetic Alphabet

PRNG (*Pseudorandom Number Generator*)

Menambahkan karakter acak (huruf besar, huruf kecil, simbol, angka) di akhir *password* menggunakan PRNG.

```python
def for_measuring_prng_recommendation(password):
    uppercase = secrets.choice(string.ascii_uppercase)
    lowercase = secrets.choice(string.ascii_lowercase)
    symbol = secrets.choice(string.punctuation)
    digit = secrets.choice(string.digits)
    recommended_password = password + uppercase +
                           lowercase + symbol + digit
    return recommended_password
```

Gambar 4. Blok Rekomendasi *Password* dengan PRNG

TRNG (*True Random Number Generator*)

Mengganti karakter tertentu dengan representasi fonetik NATO dan menambahkan karakter acak berbasis TRNG di akhir *password*.

```python
def for_measuring_trng_recommendation(password):
    replaceable_indices = [i for i, char in enumerate(password)
                           if char.isalpha()]

    if not replaceable_indices:
        return password

    index = secrets.choice(replaceable_indices)
    char = password[index]

    phonetic = phonetic_alphabet.get(char.upper(), char)

    new_password = password[:index] + phonetic + password[index + 1:]
    random_chars = os.urandom(2)
    random_hex = random_chars.hex()

    recommended_password = new_password + random_hex
    return recommended_password
```

Gambar 5. Blok Rekomendasi *Password* dengan TRNG

Program Utama

Mengelola interaksi dengan pengguna yang meliputi input *password*, memvalidasi kekuatan *password*, dan menampilkan rekomendasi berbasis metode NATO, PRNG, dan TRNG.

```python
def main():
    loop = True
    while loop:
        print("*********************************************************")
        print("PASSWORD STRENGTH MEASUREMENT AND RECOMMENDATION PROGRAM")
        print("*********************************************************")
        common_passwords = load_password()
        password = input("\tEnter password\t\t\t: ")
        if len(password) < 8:
            print("\tPassword is too short.")
            print("\tMinimum length of password is 8 characters.")
            return 0
        elif password.isdigit():
            print("\tPassword must contain alphabetical letter.")
            return 0
        elif password in common_passwords:
            print("\tPassword is commonly used.")
            print("\tPlease choose a stronger password.")
            return 0
        entropy = password_entropy(password)
        strength_level = password_strength_level(entropy)

        print(f"\tYour password strength \t: {strength_level}")
        print(f"\tPassword entropy\t\t: {round(entropy, 2)} bits")
        print("====== Password Recommendations ======")
        new_pass1 = for_measuring_nato_recommendation(password)
        new_pass2 = for_measuring_prng_recommendation(password)
        new_pass3 = for_measuring_trng_recommendation(password)
        print(f"\tNATO PHONETIC ALPHABET: {new_pass1}")
        print(f"\tPRNG METHOD: {new_pass2}")
        print(f"\tTRNG METHOD: {new_pass3}")
        a = str(input("Run again? (Y/N) "))
        if a.lower() == "n":
            loop = False
```

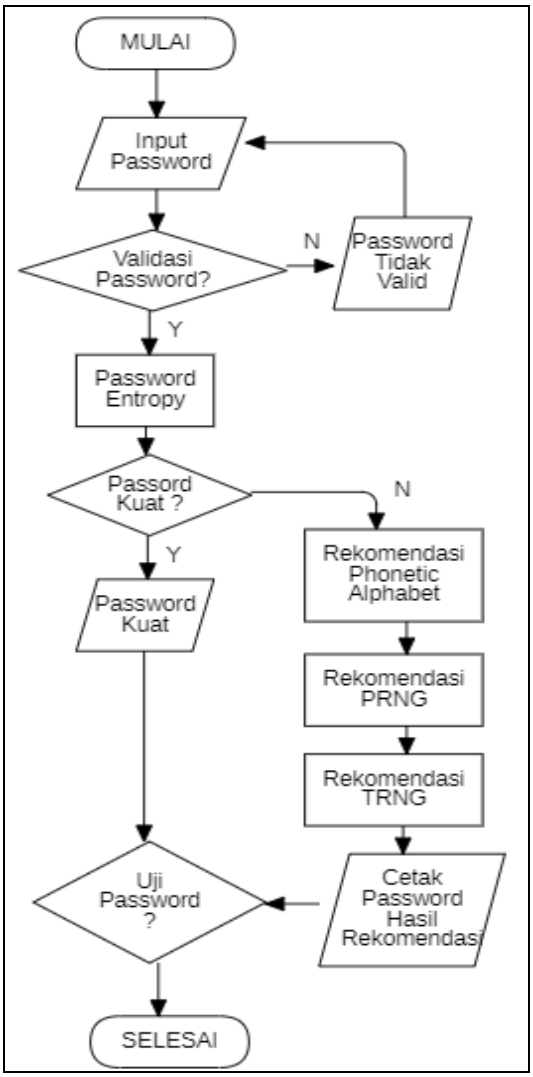
Gambar 6. Blok Kode Program Utama

3.2. Alur Program Mengukur Kekuatan dan Rekomendasi *Password*

Berikut adalah penjelasan mengenai alur logika dari program pengukuran kekuatan dan

rekomendasi *password*. Berikut adalah penjelasan untuk setiap bagian serta gambar flowchart yang terlihat pada gambar 7.

1. Start dan Input *Password*
   Program dimulai dengan meminta pengguna untuk memasukkan *password*.
   Tujuan: Menginisiasi proses validasi, pengukuran kekuatan, dan pemberian rekomendasi *password*.
2. Validasi *Password*
   Panjang *Password*
   Memeriksa apakah *password* memiliki panjang minimal 8 karakter. Jika tidak, program memberikan pesan kesalahan dan berhenti.
   Keberadaan Huruf
   Memastikan *password* mengandung setidaknya satu huruf alfabet. Jika tidak, program meminta pengguna mengganti *password*.
   Password Umum
   Membandingkan *password* dengan daftar *password* umum dari file common_*password*s.txt. Jika ditemukan dalam daftar, program meminta *password* yang lebih kuat.
3. Pengukuran Entropi
   Setelah *password* valid, program menghitung entropi berdasarkan variasi karakter (huruf besar/kecil, angka, simbol) dan panjang *password*.
4. Evaluasi Kekuatan *Password*
   Berdasarkan nilai entropi, *password* diklasifikasikan menjadi: Very Weak, Weak, Moderate, Strong, atau Very Strong. Tingkat kekuatan *password* ditampilkan kepada pengguna bersama nilai entropi.
5. Rekomendasi *Password*
   Program menghasilkan tiga jenis rekomendasi:
   NATO Phonetic Alphabet
   Mengganti beberapa karakter dalam *password* dengan alfabet fonetik NATO (contoh: "A" menjadi "Alpha").
   PRNG
   Menambahkan karakter acak (huruf besar, kecil, simbol, angka) menggunakan PRNG.
   TRNG
   Mengganti beberapa karakter dengan representasi fonetik NATO dan menambahkan karakter acak berbasis TRNG.
   Setiap rekomendasi disertai tingkat kekuatan dan nilai entropinya.
6. Pilihan Pengguna
   Program menanyakan apakah pengguna ingin menjalankan program kembali:
   Ya: Program mengulang dari awal (*password* baru).
   Tidak: Program berhenti.



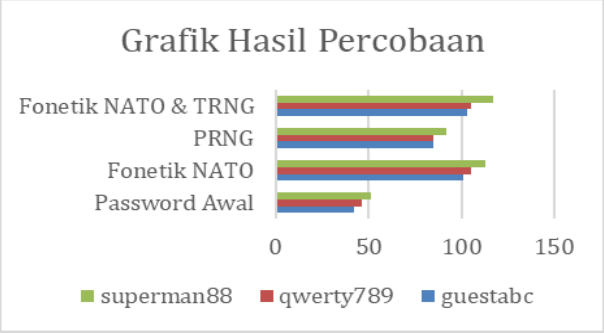
Gambar 7. Flow Diagram Program

3.3. Uji Program Pengukuran dan Rekomendasi *Password*

Di dalam percobaan, telah dilakukan input 3 *password* bersifat lemah yang berbeda ke dalam program. Dari masing-masing *password* tersebut, dilakukan percobaan input 3 kali *password* yang sama ke dalam program untuk memperoleh rekomendasi *password*. Pada tiap percobaan, muncul tiga tipe rekomendasi yang berbeda. Rekomendasi pertama merupakan rekomendasi yang muncul berdasarkan penggunaan alfabet fonetik NATO, sedangkan rekomendasi kedua diambil dari tambahan penggunaan karakter acak berdasarkan PRNG. Lalu, rekomendasi yang ketiga merupakan kombinasi dari penggunaan alfabet fonetik serta huruf dan angka acak berdasarkan TRNG.

Input Password	Entropi Awal	Percobaan ke-	Rekomendasi Password	Entropi Akhir
guestabcd	42.3 bits	1	(1) gUniformestabcDelta (2) guestabcdWu&8 (3) gueSierratabcd5ba2	108.31 bits 85.21 bits 107.18 bits
		2	(1) gUniformEchostabcd (2) guestabcdIn.0 (3) guestabcDelta4f10	102.61 bits 85.21 bits 101.22 bits
		3	(1) guEchostabcDelta (2) guestabcdAt@2 (3) guesTangoabcd6141	91.21 bits 85.21 bits 101.22 bits
qwerty789	46.53 bits	1	(1) qWhiskeyeRomeoty789 (2) qwerty789Ux}0 (3) Quebecwerty78946c5	113.13 bits 85.21 bits 107.18 bits
		2	(1) qweRomeoTangoy789 (2) qwerty789Zo>7 (3) Quebecwerty7894d22	101.22 bits 85.21 bits 107.18 bits
		3	(1) QuebecwEchorty789 (2) qwerty789Hb{7 (3) qwerTangoy789bd6a	101.22 bits 85.21 bits 101.22 bits
superman88	51.7 bits	1	(1) sUniformpEchorman88 (2) superman88Ai$1 (3) sUniformperman88e2e5	113.13 bits 91.76 bits 119.08 bits
		2	(1) suPapaermAlphan88 (2) superman88Nw>5 (3) supermaNovember8831ba	101.22 bits 91.76 bits 125.04 bits
		3	(1) supeRomeomaNovember88 (2) superman88Ch"2 (3) supermAlphan88bb9c	125.04 bits 91.76 bits 107.18 bits

Gambar 8. Tabel Hasil Percobaan Output Rekomendasi



Gambar 9. Grafik Hasil Percobaan Output Rekomendasi

Percobaan dinyatakan berhasil karena ketiga tipe rekomendasi selalu menghasilkan entropi yang lebih besar dibandingkan dengan entropi *password* pada awal input. Entropi yang diperoleh dari rekomendasi *password* juga sudah menemui kriteria yang diberikan oleh program untuk *password* yang dinyatakan kuat. Aplikasi juga dapat memberikan tiga tipe rekomendasi *password* yang mudah diingat oleh *user*, tetapi juga memiliki kompleksitas dan karakteristik yang kuat. Kompleksitas tersebut dipengaruhi oleh panjang dan variasi karakter yang ada. Semakin panjang dan banyak variasi karakter suatu *password*, maka akan semakin kuat dan sulit *password* tersebut untuk bisa diretak. Berdasarkan hasil percobaan yang dilakukan, penelitian ini memiliki suatu keunggulan dan kebaruan dibandingkan dengan penelitian lainnya. Berikut adalah pembuktiannya:

1. Al Maqbali & Mitchell, C. J. (2017). AutoPass: An Automatic *Password* Generator.
   Di dalam penelitian ini terdapat *password* manager AutoPass yang dimanfaatkan sebagai tempat penyimpanan sejumlah *password* untuk *user*. *Password* yang disimpan dapat berasal dari dua sumber, yaitu *password* buatan *user* yang sudah sangat kuat atau *password* yang dihasilkan secara acak oleh AutoPass. AutoPass hanya bisa menghasilkan *password* secara acak tanpa ada campur tangan dengan *user*. Sedangkan, aplikasi kami memberikan rekomendasi berdasarkan input *password* yang dimasukkan oleh *user* sebelumnya sehingga *user* memiliki kebebasan lebih dalam pembuatan kata sandi.
2. Oesch, S. & Ruoti, S. (2020). That was Then, This is Now: A Security Evaluation of *Password* Generation, Storage, and Autofill in Browser-based *Password* Managers.
   Di dalam penelitian ini telah dianalisis penggunaan berbagai *password* manager sebagai aplikasi penyimpanan kata sandi yang dapat mengusulkan sebuah *password* yang dihasilkan secara acak. Namun, walaupun penyaranan tersebut melayani penentuan jenis karakter apa saja yang ingin digunakan, *password* tetap dihasilkan secara acak demi keamanan sehingga sulit untuk diingat. Contohnya, jika hanya ingin membuat *password* yang terdiri dari huruf besar dan huruf kecil, maka contoh *password* yang dihasilkan adalah "iGbiQhoMEdqk".

Aplikasi yang dibangun dalam penelitian ini memiliki keunggulan dibandingkan dengan beberapa penelitian sebelumnya. Aplikasi tersebut mengkombinasikan penyaranan *password* berdasarkan metode alfabet fonetik sehingga mudah diingat dan dengan penggunaan random string agar tetap memenuhi ketentuan kata sandi yang kuat. Aplikasi yang dibuat juga menjadi aplikasi untuk mengecek kekuatan *password* sekaligus memberi rekomendasi *password* yang kuat.

KESIMPULAN

Berdasarkan hasil penelitian, aplikasi yang dibentuk dapat mengukur kekuatan dan memberikan rekomendasi *password* yang kuat serta mudah diingat berdasarkan input *password* oleh *user* yang ditambahkan dengan penggunaan

alfabet fonetik NATO, *True Random Generator* (TRNG), dan *Pseudorandom Number Generator* (PRNG). Rekomendasi yang dihasilkan dapat meningkatkan jumlah entropi pada *password* karena adanya penggunaan angka, simbol, atau karakter lain, serta penambahan panjang *password*. Hal itu dibuktikkan dengan adanya permutasi *password* yang menyatakan bahwa semakin panjang dan beragam karakter dalam sebuah *password*, maka akan membuatnya semakin aman dan kuat. Dengan demikian, entropi yang dihasilkan dari rekomendasi *password* telah meningkat dibandingkan dengan entropi *password* yang pertama kali dimasukkan oleh *user*.

REFERENSI

[1] D. Xu and D. E. Tamir, "Pseudo-random number generators based on the Collatz conjecture," Int. J. Inf. Technol., vol. 11, no. 3, pp. 453–459, 2019, doi: 10.1007/s41870-019-00307-9.

[2] F. Al Maqbali and C. J. Mitchell, "AutoPass: An automatic *password* generator," Proc. - Int. Carnahan Conf. Secur. Technol., vol. 2017-Octob, pp. 1–6, 2017, doi: 10.1109/CCST.2017.8167791.

[3] J. Marquardson, "*Password* policy effects on entropy and recall: Research in progress," 18th Am. Conf. Inf. Syst. 2012, AMCIS 2012, vol. 6, no. January, pp. 4824–4832, 2012.

[4] R. K. Abdullah and R. O. Hoan, "Penerapan Enkripsi Hibrida AES-RSA untuk Meningkatkan Keamanan Layanan Sistem Informasi Distribusi Slip Gaji Implementing AES-RSA Hybrid Encryption to Enhance the Security of Salary Slip Distribution Information System," vol. 7, pp. 33–40, 2025.

[5] Azhar, Arkarni Wais, and Atthariq, "Sistem Keamanan Pada Halaman Login Menggunakan One Time *Password*," J. Embed. Syst. Secur. Intell. Syst., vol. 01, no. 2, pp. 106–113, 2020.

[6] J. Tan, L. Bauer, N. Christin, and L. F. Cranor, "Practical Recommendations for Stronger, More Usable *Password*s Combining Minimum-strength, Minimum-length, and Blocklist Requirements," Proc. ACM Conf. Comput. Commun. Secur., pp. 1407–1426, 2020, doi: 10.1145/3372297.3417882.

[7] L. Bosnjak, J. Sres, and B. Brumen, "Brute-force and dictionary attack on hashed real-world *password*s," 2018 41st Int. Conv. Inf. Commun. Technol. Electron. Microelectron. MIPRO 2018 - Proc., no. May 2018, pp. 1161–1166, 2018, doi: 10.23919/MIPRO.2018.8400211.

[8] M. S. Turan, E. Barker, W. Burr, and L. Chen, "Recommendation for *password*-based key derivation: part 1: storage applications," NIST Spec. Publ., no. December, pp. 800–132, 2010, [Online]. Available: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf

[9] P. Tasevski, "*Password* Attacks and Generation Strategies," no. November, p. 2, 2019, doi: 10.13140/RG.2.1.1247.8807.

[10] M. F. Sanner, "Python: A programming language for software integration and development," J. Mol. Graph. Model., vol. 17, no. 1, pp. 57–61, 1999.

[11] S. Oesch and S. Ruoti, "Open access to the Proceedings of the 29th USENIX Security Symposium is sponsored by USENIX. That Was Then, This Is Now: A Security Evaluation of *Password* Generation, Storage, and Autofill in Browser-Based *Password* Managers That Was Then, This Is Now: A S," 2020, [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/oesch

[12] W. Y. Aditama, I. R. Hikmah, and D. F. Priambodo, "Analisis Komparatif Keamanan Aplikasi Pengelola Kata Sandi Berbayar Lastpass, 1*Password*, dan Keeper Berdasarkan ISO/IEC 25010," J. Teknol. Inf. dan Ilmu Komput., vol. 10, no. 4, p. 857, 2023, doi: 10.25126/jtiik.20231036544.

[13] M. M. Devillers. "Analyzing *password* strength." Radboud University Nijmegen, Tech. Rep, 2, 2010.

[14] K. P. L. Vu, R. W. Proctor, A. Bhargav-Spantzel, B. L. (Belin) Tai, J. Cook, and E. Eugene Schultz, "Improving *password* security and memorability to protect personal and organizational information," Int. J. Hum. Comput. Stud., vol. 65, no. 8, pp. 744–757, Aug. 2007, doi: 10.1016/j.ijhcs.2007.03.007.

[15] W. Ma, J. Campbell, D. Tran, and D. Kleeman, "*Password entropy and password* quality," in Proceedings - 2010 4th International Conference on Network and System Security, NSS 2010, 2010, pp. 583–587. doi: 10.1109/NSS.2010.18.

[16] J. Abbott, D. Calarco, and L. J. Camp, "Factors Influencing *Password* Reuse: A Case Study." [Online]. Available: https://ssrn.com/abstract=3142270.