# Implementasi Pembangkit Bilangan Acak Menggunakan *Logistic Map* untuk Pengacakan Urutan Soal Ujian

Diky Restu Maulana - 13520017
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13520017@std.stei.itb.ac.id

***Abstrak*—Ujian selalu menuntut kejujuran kepada para pesertanya. Namun, kecurangan dalam ujian masih sering terjadi, misalnya tindakan menyontek. Banyak cara telah dilakukan oleh pengajar untuk mencegah hal ini terjadi. Salah satunya adalah dengan mengacak urutan soal ujian sehingga menyulitkan peserta ujian untuk saling berbagi jawaban. Namun, bertambahnya jumlah soal ujian dan jumlah peserta ujian membuat pengacakan ini semakin sulit dilakukan. *Logistic Map* sebagai pembangkit bilangan acak yang memenuhi teori *chaos* dapat dimanfaatkan sebagai solusi dari permasalahan ini.**

***Kata kunci—soal ujian; bilangan acak; teori chaos; Logistic Map;***

## I. Pendahuluan

Ujian merupakan tahapan dalam pendidikan untuk menguji sejauh mana pemahaman peserta didik terhadap materi yang telah dipelajarinya di kelas. Ujian identik dengan sesuatu yang menegangkan, bahkan menyeramkan bagi pesertanya. Tindak kecurangan pun sering terjadi di dalam ujian, misalnya tindakan menyontek sesama teman.

Upaya preventif dilakukan oleh para pengajar untuk mencegah kecurangan dalam ujian. Salah satunya adalah dengan mengacak urutan soal ujian untuk setiap peserta. Hal ini penting untuk memastikan keadilan dan objektivitas dalam penilaian. Pengacakan urutan soal dapat membantu mengurangi bias atau pola yang mungkin muncul ketika urutan soal dibuat secara terstruktur. Salah satu metode yang dapat digunakan dalam pengacakan adalah model logistik atau yang dikenal dengan istilah *logistic map*.

*Logistic map* adalah salah satu model matematika yang dapat menghasilkan deret angka yang terlihat acak, tetapi memiliki sifat deterministik. Model ini awalnya digunakan dalam studi populasi biologi, namun kemudian diterapkan dalam berbagai bidang, termasuk pengacakan. Dalam implementasi *Logistic Map*, parameter awal yang digunakan sebagai titik awal dalam model akan mempengaruhi deret angka yang dihasilkan.

Makalah ini bertujuan untuk membahas implementasi *logistic map* dalam konteks pengacakan urutan soal ujian. Dalam penggunaannya, model *logistic map* akan digunakan sebagai alat untuk menghasilkan urutan acak yang dapat diterapkan pada urutan soal ujian. Dengan menerapkan metode ini, diharapkan dapat menghasilkan urutan soal yang tidak terstruktur, sehingga setiap peserta ujian mendapatkan urutan soal yang berbeda-beda secara acak.

Pada makalah ini, penulis akan menjelaskan konsep dasar model *logistic map*, mempresentasikan langkah-langkah implementasi *logistic map* dalam pengacakan urutan soal ujian. Selain itu, penulis juga akan menyajikan hasil analisis dan eksperimen yang menunjukkan efektivitas penggunaan *logistic map* dalam pengacakan urutan soal ujian.

Diharapkan makalah ini dapat memberikan pemahaman yang jelas dan komprehensif tentang implementasi *Logistic Map* dalam pengacakan urutan soal ujian serta memberikan wawasan tentang potensi penggunaan metode ini dalam bidang pendidikan.

## II. Dasar Teori

### A. Bilangan Acak

Bilangan acak adalah bilangan yang tidak dapat diprediksi kemunculannya. Bilangan acak dapat berupa integer, bilangan riil antara 0 sampai 1, atau string biner. Hingga saat ini, tidak ada komputasi yang benar-benar menghasilkan deret bilangan acak secara sempurna (*truly random*). Bilangan acak yang dihasilkan oleh komputer adalah bilangan acak semu (*pseudo random*) karena menggunakan rumus-rumus matematika dan pembangkitannya dapat diulang kembali. Pembangkit bilangan acak seperti itu disebut dengan *pseudo-random number generator* (PRNG). PRNG bersifat deterministik, artinya pembangkitan suatu bilangan acak dapat diulang kembali dengan umpan yang sama.

Dalam dunia kriptografi, bilangan acak memegang peran yang sangat penting. Beberapa kegunaan bilangan acak ini antara lain:

- Pembangkitan nilai-nilai parameter kunci di dalam algoritma kriptografi kunci publik
- Pembangkitan nilai acak *k* dalam algoritma enkripsi ElGamal
- Pembangkitan *Initialization Vector* (IV) di dalam *block cipher*
- Pembangkitan string di dalam mekanisme *challenge and response* untuk otentikasi

- Pembangkitan kunci sesi oleh *client* di dalam *Secure Socket Layer* (SSL)

### B. CSPRNG

*Cryptographically secure pseudo-random number generator* (CSPRNG) adalah pembangkit bilangan acak yang aman secara kriptografi. Terdapat beberapa syarat bagi suatu pembangkit bilangan acak untuk menjadi CSPRNG, antara lain:

- Secara statistik lolos uji keacakan (*randomness test*).
- Tahan terhadap serangan (*attack*) yang serius. Serangan ini bertujuan untuk memprediksi bilangan acak yang dihasilkan dari nilai-nilai sebelumnya.

Untuk persyaratan yang kedua ini, maka CSPRNG hendaklah memenuhi dua persyaratan sebagai berikut:

- Setiap CSPRNG seharusnya memenuhi "uji bit-berikutnya" (*next-bit test*) sebagai berikut:

  Diberikan k buah bit barisan acak, maka tidak ada algoritma dalam waktu polinomial yang dapat memprediksi bit ke-(k+1) dengan peluang keberhasilan lebih dari 1/2.

- Setiap CSPRNG dapat menahan "perluasan status", yaitu jika sebagian atau semua statusnya dapat diungkap (atau diterka dengan benar) maka tidak mungkin merekonstruksi aliran bilangan acak.

### C. Teori Chaos

Dalam dunia matematika, teori *chaos* menggambarkan kebiasaan dari suatu sistem dinamis, yang keadaannya selalu berubah seiring dengan berubahnya waktu, dan sangat sensitif terhadap kondisi awal dirinya sendiri. Teori *chaos* ini juga sering disebut dengan sebutan *butterfly effect*.

Teori *chaos* memiliki sifat peka terhadap nilai awal (*sensitive dependence on initial condition*) sehingga perubahan nilai awal yang sangat kecil pun dapat benar-benar mengubah hasilnya. Hal ini terjadi walaupun sistem yang digunakan bersifat deterministik, dalam arti perubahan kondisi dari kondisi yang ada sekarang bersifat statis atau tetap. Salah satu contoh nyata dari teori *chaos* ini dapat kita lihat pada kehidupan sehari-hari, terutama pada sistem alamiah seperti cuaca.

**Gambar 1**. Gambaran Teori *Chaos*
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2022-2023/37-Pembangkit-bilangan-acak-2023.pdf

### D. Chaos Map

Dalam matematika, sebuah *chaos map* berarti fungsi pemetaan yang memiliki beberapa jenis perilaku *chaotic*. *Chaos map* sering ditemukan dalam penelitian sistem dinamik. Contoh dari *chaos map* antara lain *Henon Map*, *Tent Map*, *Lorenz Attractor*, dan *Logistic Map*.

*Tent Map* adalah fungsi iterasi, dalam bentuk tenda, membentuk suatu sistem dinamik berbasis waktu diskrit. Dibutuhkan titik $x_n$ pada garis real. Kemudian, titik tersebut dipetakan ke titik lain.

$$x_{n+1} = \begin{cases} \mu x_n, & x < 1/2 \\ \mu(1 - x_n), & x \geq 1/2 \end{cases}$$

Dengan μ bernilai positif konstan.

*Lorenz Attractor*, yang diberi nama sesuai dengan penemunya Edward N. Lorenz, adalah struktur 3-dimensi yang bersesuaian dengan perilaku jangka panjang dari aliran *chaotic*. Fungsi ini terkenal karena bentuk kupu-kupunya.

Persamaan yang terdapat dalam *Lorenz Attractor* adalah sebagai berikut.

$$\frac{dy}{dx} = \sigma(y - x)$$

$$\frac{dy}{dt} = x(\rho - z) - y$$

$$\frac{dz}{dt} = xy - \beta z$$

σ = nomor *Prandtl*
ρ = nomor *Rayleigh*
$\sigma, \rho, \beta > 0$, biasanya $\sigma = 10, \beta = \frac{8}{3}$, ρ bervariasi

*Logistic Map* merupakan contoh fungsi polinomial derajat dua yang sering digunakan sebagai contoh bagaimana rumitnya sifat *chaos* (kacau) yang dapat muncul dari suatu persamaan yang sangat sederhana. Persamaan ini dipopulerkan oleh seorang ahli biologi yang bernama Robert May pada tahun 1976, melanjutkan persamaan logistik yang dikembangkan oleh Pierre Francois Verhulst.

*Henon Map* adalah peta prototipikal 2D teriterasi dengan solusi *chaotic* yang diusulkan oleh astronom Perancis Michel Henon sebagai model sederhana dari *Poincare Map* untuk model *Lorenz*.

$$x_{n+1} = 1 + ax_n^2 + bx_{n-1}$$

Secara matematis, persamaan logistik dapat dinyatakan dengan persamaan berikut.

$$x_{i+1} = r\, x_i (1 - x_i)$$

r = laju pertumbuhan ($0 \leq x \leq 4$)
x = nilai-nilai chaos ($0 \leq x \leq 1$)

## III. Implementasi

Bagian ini bertujuan untuk menjelaskan secara rinci langkah-langkah implementasi dari pembangkitan bilangan acak menggunakan *Logistic Map* dan penggunaannya dalam melakukan pengacakan urutan soal ujian. Implementasi dari penelitian ini dibuat dalam bahasa pemrograman *python* dan akan dijelaskan lebih lanjut dalam beberapa sub bab setelah ini, termasuk algoritma yang digunakan dan perhitungan matematis yang terlibat.

### A. *Pembacaan File Soal*

Pada tahap pertama implementasi, program akan meminta beberapa masukan sebagai berikut.

```
jumlah_siswa = int(input("Masukkan jumlah siswa: "))
file_soal = input("Masukkan nama file soal (.json): ")
x = float(input("Masukkan nilai awal (di antara 0 sampai 1): "))
r = float(input("Masukkan rasio (di antara 0 sampai 4): "))
```

Soal disimpan di dalam file .json dengan format sebagai berikut.

```
[
  {
    “number” : 1,
    “question : “...”,
    “options” : {
      “a” : “...”
      “b” : “...”
      “c” : “...”
      “d” : “...”
      “e” : “...”
    },
    “answer” : “...”
  },
  {
    “number” : 2,
    “question : “...”,
    “options” : {
      “a” : “...”
      “b” : “...”
      “c” : “...”
      “d” : “...”
      “e” : “...”
    },
    “answer” : “...”,
  },
  ...
]
```

Hasil pembacaan file disimpan ke dalam sebuah variabel yang tidak akan diubah nilainya selama proses pengacakan urutan soal.

### B. *Pembangkitan Bilangan Acak*

Pembangkitan bilangan acak dilakukan sebanyak jumlah siswa untuk mendapatkan hasil yang berbeda untuk setiap siswa. Deretan bilangan acak direpresentasikan dalam bentuk *array* yang memiliki elemen sejumlah banyaknya soal.

$$numbers = [x_1, x_2, x_3, ..., x_n]$$

$n$ = jumlah soal

$x_1 = rx_0(1 - x_0)$

$x_2 = rx_1(1 - x_1)$

dst...

Fungsi yang menjalankan proses pembangkitan bilangan acak dapat dilihat pada gambar di bawah ini.

```
def generate(x, n, r):
    numbers = []
    for i in range(n):
        num = r * x * (1 - x)
        decimal = str(num).split(".")[1]
        non_zero = decimal.lstrip("0")
        modified_num = int(non_zero[:6]) if len(non_zero) > 6 else int(non_zero)
        numbers.append(modified_num)
        x = num
    return numbers
```

Fungsi `generate` menerima tiga nilai masukan, yaitu:

$x$ = Nilai awal

$n$ = Jumlah soal

$r$ = Laju pertumbuhan atau rasio

Nilai awal dan rasio didapatkan langsung dari masukan pengguna, sedangkan jumlah soal didapatkan dari panjang *array* hasil pembacaan file soal. Fungsi `generate` melakukan iterasi sebanyak jumlah soal. Pada setiap iterasi, nilai variabel num akan diperbarui dengan nilai hasil perhitungan *logistic map*.

*Logistic map* akan selalu menghasilkan nilai *floating point* di antara 0 dan 1. Sementara itu, yang kita inginkan adalah sebuah bilangan bulat supaya dapat dicari sisa pembagiannya. Oleh karena itu, *floating point* tersebut hanya akan diambil *n significant bit*. Dalam penelitian ini, setiap nilai *floating point* diambil 6 digit pertama dari kiri yang bukan nol. Alasan pemilihan jumlah digit ini adalah untuk memperbesar “kekacauan” pada perhitungan *logistic map* pada iterasi berikutnya sehingga deretan angka yang dihasilkan semakin acak.

Pada bagian utama program, fungsi `generate` akan dipanggil sebanyak jumlah siswa. Artinya, setiap siswa mendapatkan deretan bilangan acak yang berbeda. Untuk setiap iterasi, nilai awal perhitungan *logistic map* akan bertambah. Penambahan nilai cukup dengan angka yang kecil saja, bahkan sangat kecil. Pada penelitian ini, penambahan nilai awal dilakukan pada angka 0.001. Tujuannya adalah untuk membuktikan sifat *chaotic* pada *logistic map*. Seperti yang telah dijelaskan pada bagian sebelumnya, perubahan kecil pada nilai awal dapat berpengaruh besar pada hasil perhitungan *logistic map*.

### C. *Pengacakan Urutan Soal*

*Array* berisi deretan bilangan acak yang sudah didapatkan sebelumnya akan digunakan untuk melakukan pengacakan urutan soal. Bilangan yang merupakan elemen ke-*i* pada *array* bilangan acak ini akan menentukan nomor soal pada soal asli yang akan menjadi soal nomor *i* pada soal yang telah diacak. Selain itu, angka ini juga akan menentukan urutan pilihan jawaban pada soal tersebut.

Bagian program untuk melakukan pengacakan urutan soal dapat dilihat pada gambar berikut.

```python
for i in range(jumlah_siswa):
    # generate random numbers
    numbers = generate(x + i * 0.001, len(soal_ori), r)

    # salin soal
    soal_copy = soal_ori.copy()

    # acak soal dan opsi jawaban
    soal_acak = []
    jumlah_soal = len(soal_copy)
    for j in range(jumlah_soal):
        number = numbers[j]
        soal = soal_copy[number % len(soal_copy)]
        soal_copy.remove(soal)
        soal["number"] = j + 1
        opsi = list(soal["options"].values())
        opsi_acak = soal["options"]

        # acak opsi jawaban
        for k in range(len(opsi)):
            opsi_acak[chr(k + 97)] = opsi[number % len(opsi)]
            opsi.remove(opsi[number % len(opsi)])

        soal["options"] = opsi_acak
        soal_acak.append(soal)
```

Sebelum dilakukan pengacakan, soal asli perlu disalin terlebih dahulu. Salinannya ini yang akan dimodifikasi untuk keperluan pengacakan. Selanjutnya, dilakukan iterasi sebanyak jumlah soal. Pada iterasi ke-*i*, diambil sebuah bilangan yang merupakan elemen ke-*i* dari *array* bilangan acak. Bilangan ini yang akan menentukan nomor soal yang akan dipilih dan seperti apa urutan opsi jawaban dari soal tersebut.

Mekanisme pemilihan nomor soal adalah sebagai berikut.

1. Pada iterasi ke-*i*, misal $x$ adalah bilangan acak dan $n$ adalah jumlah soal
2. Nomor soal yang terpilih pada iterasi ke-*i* adalah

$$x \bmod n$$

3. Hapus soal tersebut dari salinan soal
4. Misal $p$ adalah banyaknya opsi jawaban pada soal tersebut, maka opsi jawaban "a" dari soal tersebut diubah menjadi opsi dengan urutan ke-

$$x \bmod p$$

5. Hapus opsi jawaban yang telah terpilih dari salinan soal
6. Nilai $p$ berkurang satu karena ada opsi yang dihapus
7. Ulangi langkah ke-4 untuk pilihan jawaban "b", "c", dan seterusnya hingga seluruh opsi terpilih
8. Ubah nomor soal menjadi *i*
9. Ulangi dari langkah ke-2 hingga seluruh soal terpilih

Mekanisme di atas diulangi sebanyak jumlah siswa. Apabila rangkaian mekanisme selesai dijalankan, artinya urutan soal sudah teracak dan dapat disimpan dalam format .json.

## IV. Pengujian dan Analisis

Pada bagian ini, akan dilakukan pengujian kode program dan analisis hasil uji. Bahan uji yang disiapkan berupa file .json berisi soal dengan sepuluh nomor yang masing-masing memiliki lima pilihan jawaban, yaitu dari a hingga e.

### *A. Pengujian*

Pada pengujian ini, ditetapkan nilai untuk beberapa variabel yang dapat dilihat pada tabel berikut.

**Tabel 1**. Variabel Pengujian

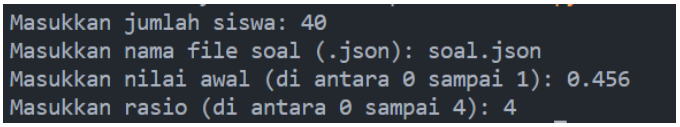
| Variabel | Nilai | Keterangan |
|---|---|---|
| Nilai awal | 0.456 | - |
| Rasio | 4.0 | - |
| Jumlah siswa | 40 | - |
| Jumlah soal | 10 | - |
| Jumlah opsi jawaban per soal | 5 | a sampai e |

Soal disimpan di dalam file bernama `soal.json`. Isi file ini adalah sebagai berikut.

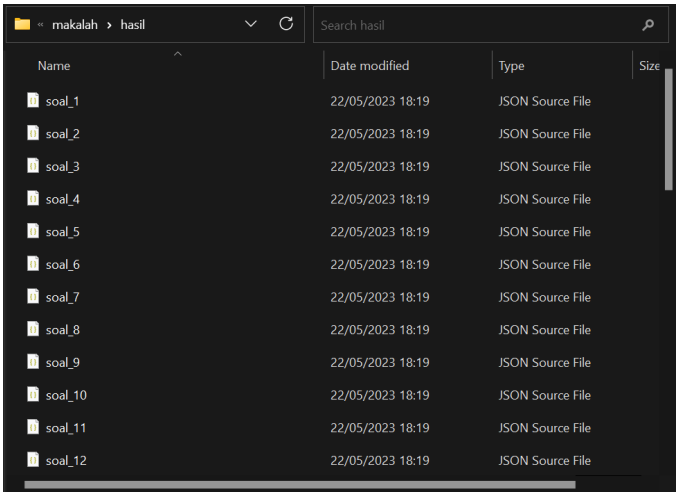
```
[
    {
        "number" : "1",
        "question" : "Mengubah susunan/posisi huruf
plainteks ke posisi lainnya merupakan teknik ...",
        "options" : {
            "a" : "Transposisi",
            "b" : "Substitusi",
            "c" : "Permutasi",
            "d" : "Kriptografi",
            "e" : "Enkripsi"
        },
        "answer" : "Transposisi"
    },
    {
        "number" : "2",
        "question" : "One-time pad berisi deretan
huruf-huruf kunci yang dibangkitkan secara ...",
        "options" : {
            "a" : "Acak",
            "b" : "Berurutan",
            "c" : "Berulang",
            "d" : "Berkesinambungan",
            "e" : "Berulang-ulang"
        },
        "answer" : "Acak"
    },
    {
        "number" : "3",
        "question" : "Kriptografi menurut schneier (1996)
adalah ...",
        "options" : {
            "a" : "Teknik untuk menjaga keaslian data",
            "b" : "Metode untuk menjaga pesan rahasia",
            "c" : "ilmu dan seni untuk menjaga keamanan
pesan",
            "d" : "Metode untuk enkripsi dan dekripsi",
            "e" : "Teknik untuk menjaga keutuhan data"
        },
        "answer" : "ilmu dan seni untuk menjaga keamanan
pesan"
    },
    {
        "number" : "4",
        "question" : "Steganografi pertama kali digunakan di
...",
        "options" : {
            "a" : "Mesir",
            "b" : "Yunani",
            "c" : "Roma",
            "d" : "Cina",
            "e" : "Arab"
        },
```

```
        "answer" : "Yunani"
    },
    {
        "number" : "5",
        "question" : "Mesin enkripsi yang digunakan pada
perang dunia II oleh pemerintahan Nazi adalah ...",
        "options" : {
            "a" : "Enigma",
            "b" : "Caesar",
            "c" : "Vigenere",
            "d" : "Playfair",
            "e" : "One-time pad"
        },
        "answer" : "Enigma"
    },
    {
        "number" : "6",
        "question" : "Vigenere Cipher berhasil dipecahkan
oleh ...",
        "options" : {
            "a" : "Charles Babbage",
            "b" : "Alan Turing",
            "c" : "Blaise Pascal",
            "d" : "Leonardo Da Vinci",
            "e" : "Leonhard Euler"
        },
        "answer" : "Charles Babbage"
    },
    {
        "number" : "7",
        "question" : "Cipher yang Termasuk ke dalam cipher
abjad-majemuk adalah ...",
        "options" : {
            "a" : "Caesar",
            "b" : "Vigenere",
            "c" : "Playfair",
            "d" : "Enigma",
            "e" : "One-time pad"
        },
        "answer" : "Vigenere"
    },
    {
        "number" : "8",
        "question" : "Steganografi pada dasarnya adalah
...",
        "options" : {
            "a" : "Teknik untuk menyembunyikan pesan",
            "b" : "Teknik untuk mengubah pesan",
            "c" : "Teknik untuk mengenkripsi pesan",
            "d" : "Teknik untuk mengubah posisi pesan",
            "e" : "Teknik untuk memusnahkan pesan"
        },
        "answer" : "Teknik untuk menyembunyikan pesan"
    },
    {
        "number" : "9",
        "question" : "Kunci enkripsi yang sama dengan proses
dekripsi disebut ...",
        "options" : {
            "a" : "Kunci simetris",
            "b" : "Kunci asimetris",
            "c" : "Kunci publik",
            "d" : "Kunci privat",
            "e" : "Kunci kriptografi"
        },
        "answer" : "Kunci simetris"
    },
    {
        "number" : "10",
        "question" : "Teknik menyisipkan informasi yang
mengacu pada pemilik gambar disebut ...",
        "options" : {
            "a" : "Watermarking",
            "b" : "Steganografi",
            "c" : "Cryptography",
            "d" : "Enkripsi",
            "e" : "Dekripsi"
        },
        "answer" : "Watermarking"
    }
]
```


Saat program dijalankan, muncul *prompt* input seperti pada gambar di bawah ini. Kemudian, masukkan nilai sebagaimana pada Tabel 1.




Program menghasilkan sebanyak 40 file .json dengan nama file `soal_1.json`, `soal_2.json`, dan seterusnya berisi soal yang telah diacak.




**Gambar 2**. File Soal yang Telah Diacak
Sumber: dokumentasi pribadi

### *B. Analisis Kesamaan dengan Soal Asli*

Analisis dilakukan dengan cara menghitung jumlah kesamaan urutan pertanyaan dan urutan opsi jawaban antara soal asli dengan setiap soal yang sudah diacak. Hasil analisis dapat dilihat pada tabel berikut.

**Tabel 2**. Analisis Pengujian

| Nomor File Soal | Jumlah Kesamaan | Persentase Kesamaan |
|---|---|---|
| 1 | 0 | 0 % |
| 2 | 2 | 3.33 % |
| 3 | 0 | 0 % |
| 4 | 0 | 0 % |
| 5 | 1 | 1.66 % |
| 6 | 4 | 6.66 % |
| 7 | 1 | 1.66 % |
| 8 | 0 | 0 % |
| 9 | 2 | 3.33 % |
| 10 | 1 | 1.66 % |

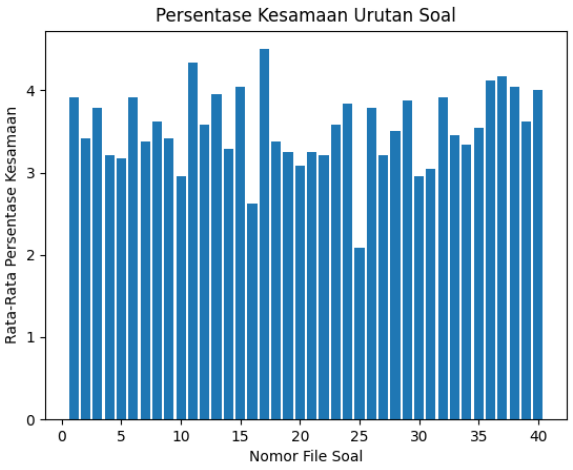
| | | |
|---|---|---|
| 11 | 3 | 5 % |
| 12 | 0 | 0 % |
| 13 | 3 | 5 % |
| 14 | 0 | 0 % |
| 15 | 7 | 11.66 % |
| 16 | 2 | 3.33 % |
| 17 | 7 | 11.66 % |
| 18 | 3 | 5 % |
| 19 | 1 | 1.66 % |
| 20 | 1 | 1.66 % |
| 21 | 1 | 1.66 % |
| 22 | 3 | 5 % |
| 23 | 3 | 5 % |
| 24 | 2 | 3.33 % |
| 25 | 1 | 1.66 % |
| 26 | 3 | 5 % |
| 27 | 0 | 0 % |
| 28 | 6 | 10 % |
| 29 | 5 | 8.33 % |
| 30 | 2 | 3.33 % |
| 31 | 3 | 5 % |
| 32 | 3 | 5 % |
| 33 | 0 | 0 % |
| 34 | 0 | 0 % |
| 35 | 1 | 1.66 % |
| 36 | 2 | 3.33 % |
| 37 | 2 | 3.33 % |
| 38 | 3 | 5 % |
| 39 | 3 | 5 % |
| 40 | 2 | 3.33 % |
| **Rata-rata** | 2.075 | 3.45575 % |

Nilai persentase didapatkan dengan cara membandingkan jumlah kesamaan dengan total baris pada soal asli. Total baris adalah perkalian jumlah soal dengan jumlah baris persoal. Jumlah baris persoal adalah 1 (baris pertanyaan) ditambah jumlah opsi jawaban.

Berdasarkan Tabel 2, didapatkan nilai rata-rata jumlah kesamaan sebesar 2.075 dan persentase kesamaan sebesar 3.45575%.

### *C. Analisis Kesamaan Antar Soal*

Analisis dilakukan dengan cara menghitung persentase kesamaan urutan pertanyaan dan urutan opsi jawaban antar soal. Setiap soal yang sudah diacak dilakukan perbandingan dengan soal lain, termasuk soal asli. Grafik batang berikut menunjukkan persentase rata-rata kesamaan baris pada setiap soal.




**Gambar 3**. Grafik Persentase Kesamaan Urutan Soal
Sumber: dokumentasi pribadi

Berdasarkan grafik di atas, terlihat bahwa seluruh soal memiliki rata-rata persentase kesamaan di bawah 5%. Persentase ini sangat kecil mengingat jumlah soal yang digunakan pada pengujian hanya 10. Untuk jumlah soal yang lebih banyak, kemungkinan kesamaan baris akan semakin kecil.

## V. KESIMPULAN DAN SARAN

Dapat disimpulkan bahwa pembangkitan bilangan acak dengan *logistic map* untuk melakukan pengacakan urutan soal ujian telah berhasil dilakukan. Rata-rata persentase kesamaan baris antara soal yang sudah diacak dengan soal asli terbilang sangat kecil. Hal ini sekaligus membuktikan sifat *chaotic* pada *logistic map* menjadikannya efektif dalam proses pengacakan urutan soal ujian.

Program yang dibuat dalam penelitian ini hanya dapat menerima masukan file dengan format .json. Perlu pengembangan lebih lanjut untuk menyempurnakannya. Akhir kata, penulis berharap penelitian ini dapat memberikan wawasan mengenai potensi ilmu kriptografi untuk memajukan pendidikan di Indonesia.

## Ucapan Terima Kasih

Puji syukur ke hadirat Allah Swt. atas rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan makalah ini. Penulis juga mengucapkan terima kasih kepada Bapak Dr. Ir. Rinaldi Munir, M.T., selaku dosen pengampu mata kuliah IF4020 Kriptografi yang telah memberikan pengetahuan terhadap topik yang diangkat dalam makalah ini. Tidak lupa, penulis juga mengucapkan terima kasih kepada keluarga dan kerabat yang selalu menyemangati dalam proses penulisan makalah ini.

## Referensi

[1] Munir, Rinaldi. (2023). *Bahan Kuliah IF4020 Kriptografi*. Program Studi Informatika ITB.

[2] Sutanto, Arnold Nugroho. (2010). *Studi dan Implementasi dari Teori Chaos dengan Logistic Map sebagai Pembangkit Bilangan Acak Semu dalam Kriptografi*. Program Studi Informatika ITB.

[3] Susanto, Alvin. (2010). *Penerapan Teori Chaos di Dalam Kriptografi*. Program Studi Informatika ITB.

[4] Phatak, S. C. and S. Suresh Rao. (1995). *Logistic map: A possible random-number generator*. Phys. Rev. E 51, 3670.

## 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, 22 Mei 2023

Diky Restu Maulana