Volume 5 Issue 1 (2025) Pages 21-34 P-ISSN:2798-2696 E-ISSIN:2798-0057
JOSTECH
Journal of Science and Technology
Website: https://ejournal.uinib.ac.id/jurnal/index.php/jostech Email: jostech@uinib.ac.id

Pengacakan Soal Ujian Menggunakan *Pseudo-Random Number Generator* Dengan Metode *Logistic Map*

Ozzy Secio Riza ✉[1], Rosalina[2], Aulia Arham[3], Rendi Saputra[4]
Program Studi Sistem Informasi UIN Imam Bonjol Padang, Indonesia[1,2,3,4]
email: ozzysecioriza@uinib.ac.id[1], rosalina@uinib.ac.id[2], auliaarham@uinib.ac.id[3], randisa.putra@uinib.ac.id[4]

Received 06 September 2024, Accepted 26 February 2025, Published 31 March 2025

Abstrak

Pengacakan soal ujian merupakan salah satu metode penting untuk menjaga integritas dan keadilan dalam proses evaluasi akademik. Penelitian ini bertujuan untuk mengembangkan dan menganalisis penggunaan *Pseudo-Random Number Generator* (PRNG) dengan metode Logistic Map dalam pengacakan soal ujian. *Logistic Map*, sebuah fungsi matematis sederhana yang berasal dari teori chaos, mampu menghasilkan urutan angka yang tampak acak melalui fungsi. Penelitian ini mengkaji efektivitas Logistic Map dalam menghasilkan angka acak dan membandingkannya dengan metode PRNG lainnya. Hasil penelitian menunjukkan bahwa *Logistic Map* mampu menghasilkan urutan soal ujian yang benar-benar acak dan tidak dapat diprediksi, sehingga dapat mengurangi potensi kecurangan di kalangan peserta ujian. Selain itu, metode ini terbukti sederhana dan efisien untuk diimplementasikan dalam sistem pengacakan soal ujian skala besar. Dengan demikian, penggunaan PRNG dengan metode Logistic Map dapat menjadi solusi efektif dalam meningkatkan kualitas dan keadilan proses evaluasi di lingkungan akademik.

Kata Kunci: *Pengacakan soal ujian; Pseudo-Random Number Generator; Logistic Map; integritas akademik evaluasi pendidikan*.

Abstract

Exam question randomization is an important method for maintaining integrity and fairness in the academic evaluation process. This study aims to develop and analyze the use of a Pseudo-Random Number Generator (PRNG) with the Logistic Map method in exam question randomization. The Logistic Map, a simple mathematical function derived from chaos theory, can produce a sequence of numbers that appear random through its function. This study examines the effectiveness of the Logistic Map in generating random numbers and compares it with other PRNG methods. The results show that the Logistic Map can produce a sequence of exam questions that are truly random and unpredictable, thus reducing the potential for cheating among exam participants. Additionally, this method has proven to be simple and efficient to implement in a large-scale exam question randomization system. Therefore, the use of a PRNG with the Logistic Map method can be an effective solution for improving the quality and fairness of the evaluation process in the academic environment.

Keywords: *exam question randomization, Pseudo-Random Number Generator, Logistic Map, academic integrity, educational evaluation*.

✉ Corresponding author

PENDAHULUAN

Perkembangan teknologi informasi telah menghasilkan peningkatan pesat dalam kemampuan komputasi. Prosesor yang lebih kuat, penyimpanan data yang lebih besar, dan kemampuan komputasi yang lebih tinggi telah memungkinkan pengembangan aplikasi dan sistem yang lebih canggih [1]. Ini memungkinkan pemrosesan data yang lebih cepat dan lebih efisien, memungkinkan pengolahan dan analisis data yang lebih kompleks, serta mempercepat inovasi dalam berbagai bidang. Secara garis besar perkembangan teknologi informasi telah memiliki dampak yang signifikan dalam berbagai aspek kehidupan, tidak terkecuali dunia pendidikan. Teknologi informasi telah membuka pintu akses yang luas terhadap informasi dan sumber belajar, mendorong pembelajaran jarak jauh, serta berdampak signifikan dalam pelaksanaan tes ujian [2].

Pengacakan soal ujian merupakan salah satu metode penting dalam dunia pendidikan untuk memastikan keadilan dan menghindari kecurangan. Pengacakan soal yang baik akan membuat setiap peserta ujian mendapatkan urutan soal yang berbeda, sehingga mengurangi kemungkinan terjadinya kerja sama yang tidak diinginkan di antara peserta ujian. Salah satu teknik yang digunakan untuk pengacakan soal ujian adalah dengan menggunakan *Pseudo-Random Number Generator* (PRNG).

Bilangan acak (*random*) banyak digunakan dalam kriptografi, misalnya untuk pembangkitan elemen – elemen kunci, pembangkitan *initialization vector* (*IV*), dan sebagainya [3] *Random Number Generator (RNG)* merupakan algoritma yang digunakan untuk menghasilkan angka acak yang telah banyak diterapkan dalam berbagai bidang dan aplikasi yang membutuhkan elemen keacakan [4], [5], [6].

Pseudo-Random Number Generator (PRNG) adalah algoritma yang digunakan untuk menghasilkan urutan angka yang tidak dapat diprediksi [7]. Salah satu metode PRNG yang populer adalah *Logistic Map,* sebuah metode yang berasal dari teori chaos. *Logistic Map* adalah fungsi matematis sederhana yang dapat menghasilkan perilaku yang sangat kompleks dan seolah-olah acak, meskipun deterministik. Pentingnya pengacakan soal ujian tidak bisa dipandang sebelah mata. Dalam lingkungan akademik, terutama di era digital seperti sekarang, kecurangan bisa terjadi dengan berbagai cara. Oleh karena itu, pengacakan soal menjadi salah satu strategi penting dalam menjaga integritas dan kredibilitas proses evaluasi. *Logistic Map* sebagai metode PRNG dipilih karena karakteristiknya yang unik dalam menghasilkan angka acak. Fungsi *Logistic Map* adalah $x_{n+1} = rx_n(1 - x_n)$, di mana r adalah parameter yang menentukan perilaku dari sistem tersebut, dan x_n adalah nilai pada iterasi ke-n. Dengan pemilihan nilai r yang tepat, Logistic Map dapat menghasilkan deret angka yang tampak acak dan tidak berulang dalam jangka panjang.

Pada Penelitian [8] menggunakan algoritma *Fisher Yates Shuffle* (FYS) untuk melakukan pengacakan soal dalam aplikasi ujian berbasis web dengan tujuan mengurangi kemungkinan kolusi antara siswa. Sementara itu, *Bubble Sort* digunakan untuk menyusun soal secara menurun dengan maksud mendapatkan soal terbaru dari database. Pengujian dilakukan dengan mengukur waktu yang diperlukan oleh kedua algoritma untuk melakukan proses pengacakan (FYS) dan pengurutan (Bubble Sort). Rata-rata kecepatan pengacakan data untuk kumpulan soal sebanyak 100, 200, 300,

400, dan 500 adalah 343 ms, sedangkan rata-rata kecepatan algoritma Bubble Sort untuk data sejumlah tersebut adalah 433 ms.

Dalam penelitian [9], digunakan algoritma Fisher-Yates Shuffle dalam aplikasi pengujian kompetensi wartawan. Algoritma ini digunakan untuk mengacak pertanyaan dalam formulir soal sehingga pertanyaan-pertanyaan tersebut akan disusun ulang sesuai dengan prinsip kerja algoritma tersebut. Proses pengacakan ini akan diterapkan di setiap aplikasi yang diinstal pada smartphone masing-masing wartawan, dengan tujuan mengurangi kemungkinan tindakan kecurangan.

Pada penelitian [10] mengembangkan sistem ujian standar online berbasis HOTS yang menggunakan algoritma *Fisher-Yates Shuffle* sebagai metode pengacakan soal. Hal ini bertujuan untuk mengurangi potensi kecurangan yang mungkin dilakukan oleh mahasiswa. Hasil dari penelitian ini menunjukkan bahwa penggunaan algoritma *Fisher-Yates Shuffle* dalam pengacakan soal dan jawaban memungkinkan setiap mahasiswa mendapatkan urutan yang berbeda dalam soal-soal pilihan ganda. Ini bertujuan untuk mengurangi tindakan kecurangan selama ujian. Berdasarkan hasil pengujian yang telah dilakukan, sistem yang dikembangkan terbukti berfungsi dengan baik sesuai dengan proses yang diterapkan dalam ujian standar online berbasis HOTS.

Pada penelitian [11] menggunakan algoritma *Fisher-Yates Shuffle* dalam pengacakan soal pada ujian masuk POLRI. sistem ini memberikan keunggulan dalam hal efektivitas metodenya dan kompleksitas algoritma yang optimal, yaitu $O(n)$. Penerapan algoritma pengacakan ini membuat sulit bagi peserta ujian untuk melakukan tindakan kecurangan, karena setiap orang mendapatkan urutan soal yang sudah diacak dan berbeda. Penelitian ini mengadopsi model pengembangan perangkat lunak *waterfall* dengan langkah-langkah berupa pengumpulan data, analisis kebutuhan, perancangan sistem, implementasi, dan pengujian sistem. Hasil pengujian dengan menggunakan metode *Blackbox* menunjukkan bahwa sistem yang dikembangkan berhasil dalam pengacakan soal dan semua komponen sistem berfungsi sebagaimana mestinya.

Pada penelitian [12] menggunakan algoritma *Fisher-Yates* dalam aplikasi *Computer Based Testing* (CBT) untuk proses penerimaan mahasiswa baru di Universitas Lancang Kuning. Aplikasi ini memungkinkan pengacakan soal yang menghasilkan perbedaan tampilan soal pada setiap peserta ujian. Ini berarti bahwa setiap mahasiswa akan melihat soal dengan nomor yang sama, tetapi dengan bentuk soal yang berbeda. Penerapan algoritma Fisher-Yates dalam pengacakan soal di aplikasi CBT ini memberikan hasil yang efektif dan merata dalam pengacakan soal-soal yang ada. Dengan adanya aplikasi ini, tindakan kecurangan dalam pelaksanaan ujian penerimaan mahasiswa baru dapat diminimalisir. Aplikasi ini juga membantu dalam proses penerimaan mahasiswa baru di Universitas Lancang Kuning. Rata – rata penelitian sebelumnya menggunakan algoritma *fisher-yates* untuk pengacakan soal, belum ada yang menggunakan *logistic map* sebagai metode pengacakan.

Penelitian ini akan mengeksplorasi penggunaan *Logistic Map* sebagai metode untuk menghasilkan angka acak yang digunakan dalam pengacakan soal ujian. *Logistic Map* memiliki kelebihan karena kesederhanaan dan efisiensinya dalam menghasilkan angka acak. *Logistic Map* sering diterapkan dalam penelitian kriptografi yang berkaitan dengan bilangan acak seperti pada penelitian[13], [14], [15] yang menggunakan *logistic map* pada pembangkitan kunci untuk enkripsi citra digital.

Dengan menggunakan metode ini, diharapkan proses pengacakan soal ujian dapat dilakukan dengan lebih efektif dan efisien, serta menghasilkan urutan soal yang benar-benar acak dan tidak dapat diprediksi.

METODOLOGI

A. Bilangan Acak

Bilangan acak merupakan bilangan yang kemunculannya tidak dapat di-prediksi. Bilangan acak dapat berwujud sebagai bilangan bulat, bilangan riil antara 0 dan 1, atau string biner. Hingga kini, belum ada metode komputasi yang mampu menghasilkan deret bilangan acak secara sempurna (*truly random*). Bilangan acak yang dihasilkan oleh komputer disebut bilangan acak semu (pseudo-random) karena dihasilkan melalui algoritma matematika, sehingga pembangkitan bilangan tersebut dapat diulang kembali. Alat yang digunakan untuk menghasilkan bilangan acak semu ini dikenal dengan nama *Pseudo-random Number Generator* (PRNG). PRNG bersifat deterministik, yang berarti bahwa proses pembangkitan bilangan acak dapat direproduksi jika menggunakan umpan (*seed*) yang sama.

Bilangan acak semu digunakan dalam berbagai aplikasi, termasuk dalam pengacakan soal ujian, simulasi Monte Carlo, enkripsi data, dan permainan komputer. Meskipun tidak sepenuhnya acak, PRNG cukup memadai untuk banyak keperluan praktis, terutama karena kecepatan dan efisiensinya dalam menghasilkan deret bilangan yang tampak acak. Untuk meningkatkan kualitas bilangan acak semu, berbagai metode PRNG telah dikembangkan, termasuk metode Logistic Map yang berasal dari teori chaos. Metode ini mampu menghasilkan urutan bilangan yang kompleks dan sulit diprediksi, menjadikannya salah satu pilihan yang efektif untuk berbagai aplikasi yang membutuhkan bilangan acak.

Dalam kriptografi, bilangan acak memiliki peran yang sangat krusial. Beberapa penerapan bilangan acak ini di antaranya adalah:

1. **Kunci Enkripsi:** Bilangan acak digunakan untuk menghasilkan kunci enkripsi yang aman. Kunci yang benar-benar acak sulit diprediksi, sehingga meningkatkan keamanan data yang dienkripsi.
2. ***Nonce*:** Nonce (number used once) adalah bilangan acak yang digunakan hanya sekali dalam protokol komunikasi. Nonce membantu mencegah serangan replay dengan memastikan bahwa setiap sesi komunikasi menggunakan bilangan unik.
3. ***Vektor Inisialisasi* (IV):** IV adalah bilangan acak yang digunakan untuk menginisialisasi algoritma enkripsi, terutama dalam mode enkripsi blok. IV memastikan bahwa enkripsi data yang sama dengan kunci yang sama akan menghasilkan ciphertext yang berbeda.
4. ***Salt*:** Dalam penyimpanan kata sandi, salt adalah bilangan acak yang ditambahkan ke kata sandi sebelum di-hash. Salt membantu melindungi kata sandi dari serangan kamus dan serangan tabel pelangi (rainbow table).
5. **Pembangkitan Kunci Asimetris:** Bilangan acak digunakan dalam algoritma pembangkitan kunci asimetris (seperti RSA atau ECC) untuk menghasilkan pasangan kunci publik dan privat yang aman.

6. ***Shuffling:*** Dalam beberapa algoritma kriptografi, data perlu diacak atau dikocok secara acak untuk memastikan distribusi yang merata dan menghindari pola yang dapat diprediksi.

B. *Cryptographically Secure Pseudo-random Number Generator* (CSPRNG)

Cryptographically Secure Pseudo-random Number Generator (CSPRNG) adalah jenis pembangkit bilangan acak semu (PRNG) yang dirancang untuk memenuhi standar keamanan kriptografi. CSPRNG menghasilkan bilangan acak yang tidak hanya tampak acak tetapi juga sulit diprediksi, bahkan dengan pengetahuan sebagian dari keluaran atau keadaan internal pembangkit[3], [16]. Karakteristik utama dari CSPRNG meliputi:

1. Keamanan: Bilangan acak yang dihasilkan tidak dapat diprediksi, meskipun sebagian dari nilai atau keadaan internal diketahui.
2. Uji Kekuatan: Lulus berbagai uji statistik untuk memastikan bahwa pola atau prediktabilitas dalam urutan bilangan acak tidak dapat ditemukan.
3. Resistensi terhadap Serangan: Tahan terhadap berbagai jenis serangan kriptografi, termasuk serangan analisis statistik dan serangan yang mencoba merekonstruksi keadaan internal.

Sebagian besar protokol kriptografi memerlukan pembuatan dan penggunaan nilai-nilai rahasia yang harus dijaga kerahasiaannya dari pihak yang tidak berwenang. Angka acak diperlukan untuk menciptakan *number used once* (*nonce*), tantangan, nilai blinding, dan byte padding [7].

CSPRNG digunakan dalam berbagai aplikasi keamanan, termasuk generasi kunci enkripsi, nonce, salt, dan berbagai protokol keamanan lainnya. Algoritma umum yang digunakan dalam CSPRNG termasuk algoritma berbasis hash, cipher block chaining (CBC), dan teknik lainnya yang memastikan keamanan dan ketidakpastian keluaran bilangan acak.

C. Teori Chaos

Teori chaos adalah cabang dari matematika dan fisika yang mempelajari sistem dinamis yang sangat sensitif terhadap kondisi awal, sebuah fenomena yang sering dikenal sebagai "efek kupu-kupu." Teori ini menunjukkan bahwa sistem-sistem yang tampaknya acak dan tidak dapat diprediksi sebenarnya memiliki pola, hukum, dan keteraturan yang mendasarinya [3]. Karakteristik utama dari teori chaos meliputi:

1. Sensitivitas terhadap Kondisi Awal: Perubahan kecil dalam kondisi awal dapat menghasilkan hasil yang sangat berbeda, membuat prediksi jangka panjang menjadi sangat sulit.
2. Dinamika Non-linear: Sistem chaos sering kali non-linear, artinya perubahan output tidak proporsional dengan perubahan input.
3. Fraktal: Banyak sistem chaos menunjukkan struktur fraktal, yang berarti mereka memiliki pola yang berulang pada skala yang berbeda.
4. Tak Terduga tetapi Teratur: Meskipun perilaku jangka panjang dari sistem chaos tampak acak, mereka mengikuti hukum matematis tertentu dan memiliki keteraturan dalam ketidakpastian.

Teori chaos memiliki aplikasi luas dalam berbagai bidang, termasuk meteorologi, ekologi, ekonomi, dan bahkan kriptografi. Dalam kriptografi, konsep

chaos digunakan untuk menciptakan algoritma yang menghasilkan bilangan acak atau pola yang sulit diprediksi, meningkatkan keamanan sistem enkripsi.

D. Chaos Map

Chaos map adalah fungsi matematika yang menggambarkan sistem dinamis yang menunjukkan perilaku chaos, yaitu perilaku yang sangat sensitif terhadap kondisi awal. Salah satu contoh paling terkenal dari chaos map adalah *Logistic Map*, yang digunakan untuk memodelkan populasi biologis tetapi juga diterapkan dalam kriptografi dan teori chaos untuk menghasilkan bilangan acak. *Logistic Map* adalah contoh dari fungsi polinomial derajat dua yang sering digunakan untuk menunjukkan kompleksitas sifat chaos yang dapat muncul dari persamaan yang sangat sederhana. Persamaan ini dipopulerkan oleh ahli biologi Robert May pada tahun 1976, yang melanjutkan karya sebelumnya dari Pierre Francois Verhulst yang mengembangkan persamaan logistik. Persamaan matematikanya adalah:

$$x_{n+1} = r x_n (1 - x_n) \quad (1)$$

Di mana:

- x_n adalah nilai pada iterasi ke-n dengan $0 \leq x_n \leq 1$
- r adalah parameter yang menentukan perilaku sistem, dengan $0 \leq r \leq 4$

Periaku Logistic Map

- Untuk $0 \leq r \leq 1$: Nilai x_n akan menuju nol terlepas dari nilai awal x_0.
- Untuk $1 \leq r \leq 3$: Nilai x_n akan menuju nilai tetap yang bergantung pada r.
- Untuk $3 \leq r \leq 3.57$: Nilai x_n akan berosilasi antara beberapa nilai tetap.
- Untuk $3.57 \leq r \leq 4$: Sistem menunjukkan perilaku chaos, dimana nilai x_n tampak acak dan sangat sensitif terhadap kondisi awal.

E. Metode yang Diusulkan

Metode yang diusulkan pada penelitian ini adalah penerapan metode *logistic map* dalam pengacakan soal ujian. Metode ini menggunakan persamaan Logistic Map untuk menghasilkan bilangan pseudo-acak yang digunakan dalam pengacakan soal ujian. Pengacakan soal dengan metode yang diusulkan menggunakan persamaan (1) dimana :

1. Nilai x_n adalah nilai pada iterasi ke-n dengan nilai yang didapatkan dari id peserta ujian yang dinormalisasikan menjadi nilai yang memenuhi ketentuan $0 \leq x_n \leq 1$
2. r adalah parameter yang menentukan perilaku sistem, dengan r = 4.
3. Dilakukan perhitungan menggunakan persamaan [1].
4. Hasil yang didapatkan x_n pada iterasi pertama digunakan untuk menentukan soal pertama dengan cara mengambil 3 digit terakhir dari nilai x_n kemudian dimodulus dengan jumlah bank soal.
5. Nilai x_n yang didapatkan akan digunakan untuk perhitungan soal selanjutnya.
6. Proses ini diulang hingga jumlah soal yang diperlukan terpenuhi.
7. Jika pada perhitungan poin 4 didapatkan nilai yang sudah didapatkan sebelumnya, maka perhitungan dilanjutkan tanpa menggunakan nilai tersebut.

HASIL DAN PEMBAHASAN

Penelitian ini mengusulkan kombinasi algoritma *chaos map* untuk pengacakan soal dan pilihan jawaban. Penambahan *chaos map* dalam penelitian ini diharapkan dapat meningkatkan hasil pengacakan soal sehingga meningkatkan keamanan soal ujian karena urutan soal dan pilihan jawaban yang teracak akan menyulitkan peserta untuk melakukan tindakan curang. Pengujian hasil ekstraksi soal ujian akan membandingkan pengacakan soal menggunakan algoritma *array_rand* yang selama ini umum digunakan dalam sistem ujian berbasis komputer yang menggunakan bahasa pemrograman PHP dengan hasil pengacakan soal menggunakan kombinasi algoritma *chaos map* yang diusulkan.

a. Skenario pengujian

Skenario pengujian yang dilakukan adalah sebagai berikut:

1. Terdapat bank soal yang berisikan 100 soal.
2. Bank soal akan terpilih 25 buah soal yang akan diberikan kepada 20 orang peserta ujian secara acak.
3. Dilakukan proses pengacakan soal menggunakan algoritma *array_rand* dan algoritma yang diusulkan kemudian dihitung perulangan soal yang muncul pada masing – masing peserta ujian.
4. Hasil soal yang muncul kemudian dilihat pola kemunculan soal pada masing – masing penggunaan algoritma.

b. Analisis Hasil Pengujian

Setelah dilakukan pengujian dengan menjalankan proses skenario pengujian didapatkan hasil sebagai berikut:

1. Hasil pengacakan

Hasil pengacakan soal yang muncul dapat dilihat pada Tabel 1 untuk pengujian menggunakan fungsi *array_rand* dan pada Tabel 2 untuk pengujian menggunakan fungsi *random* dengan metode *Logistic chaos*.

Tabel 1. Hasil Percobaan Menggunakan Fungsi array_rand

Soal Ke	Peserta Ujian																				KK	TK	PK
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20			
1	2	1	1	10	7	2	2	7	2	1	4	18	1	1	1	2	2	1	4	9	19	13	0.684
2	3	4	2	11	10	3	14	10	3	3	5	20	5	14	14	8	5	5	6	10	19	10	0.526
3	5	5	6	12	16	6	16	14	10	4	7	22	10	16	18	10	7	10	7	18	19	9	0.474
4	9	6	9	13	18	8	26	16	18	11	8	38	12	21	21	12	15	13	9	29	19	7	0.368
5	14	11	11	19	19	10	28	22	21	13	13	43	14	28	28	14	16	14	12	33	19	8	0.421
6	15	12	14	20	22	12	38	26	22	16	24	45	15	30	35	15	20	24	13	39	19	6	0.316
7	29	13	20	23	26	13	45	30	27	18	25	46	21	42	43	19	21	29	21	40	19	4	0.211
8	31	19	21	24	29	18	50	33	30	29	30	49	24	44	44	20	23	32	26	41	19	4	0.211
9	33	22	24	25	36	25	52	36	40	33	31	54	35	45	50	22	27	35	27	43	19	6	0.316
10	46	25	27	29	42	29	57	39	44	39	38	56	36	46	52	29	30	37	28	48	19	4	0.211
11	48	27	39	35	45	30	58	40	45	47	43	67	40	49	54	30	35	38	29	51	19	4	0.211
12	55	28	43	42	46	38	59	42	46	52	45	68	41	50	55	33	41	39	33	53	19	5	0.263
13	56	46	48	43	47	41	61	46	53	54	47	69	47	52	58	36	46	42	40	54	19	5	0.263
14	58	50	52	46	48	51	68	48	54	55	49	70	48	53	66	37	57	43	41	55	19	3	0.158
15	59	53	53	47	60	54	71	50	57	59	52	72	53	59	71	45	59	45	43	60	19	8	0.421
16	67	55	55	51	64	55	74	53	58	63	60	75	55	60	74	46	66	49	49	64	19	7	0.368
17	70	59	61	54	70	60	75	54	63	64	62	76	56	71	76	48	71	56	57	68	19	5	0.263
18	75	68	62	55	73	63	78	60	64	70	66	77	60	72	83	51	73	57	62	69	19	3	0.158
19	76	71	65	57	77	67	79	67	67	73	68	81	61	74	84	53	81	59	65	70	19	4	0.211
20	81	74	69	60	79	71	82	71	68	75	75	82	65	75	85	74	89	67	71	72	19	6	0.316
21	87	78	71	63	85	86	84	76	71	77	83	83	74	79	87	90	90	69	82	75	19	4	0.211
22	90	85	74	67	86	89	89	83	78	85	85	87	80	80	89	91	91	77	90	78	19	8	0.421
23	92	88	77	75	91	95	90	88	91	87	88	90	92	85	95	96	95	81	95	93	19	8	0.421
24	95	89	87	76	97	96	97	92	93	90	89	93	94	94	97	97	96	82	98	94	19	8	0.421
25	98	97	93	77	100	100	100	98	100	95	99	97	100	98	98	99	99	88	99	97	19	12	0.632
Rata-rata tingkat kesamaan (0 - 1)																							0.339

Pada Tabel 1 dapat dilihat bahwa secara keseluruhan banyak terjadi perulangan pada kemunculan soal kepada peserta ujian dengan rata – rata tingkat kesamaan sebesar 0,339 atau 33,9%. Kesamaan paling tinggi terjadi pada 5 soal pertama dengan persentase kesamaan tertinggi pada soal ke-1 dengan persentase kesamaan sebesar 68,4% dimana terjadi 13 dari 19 kemungkinan kesamaan yang terjadi. Untuk soal ke-1 hanya 3 peserta yang mendapatkan soal yang benar – benar berbeda dari peserta

lainnya. Kemudian diikuti dengan soal ke-2 dengan kemiripan sebesar 52,6%, soal ke-3 dengan kemiripan sebesar 47,4%, soal ke-5 dengan kemiripan sebesar 42,1%, dan soal ke-4 dengan kemiripan sebesar 36,8%. Soal ke-6 hingga soal ke-20 mengalami penurunan kemiripan, tapi meningkat kembali pada 4 soal terakhir dengan persentase kemiripan sebesar 42,1% pada soal ke-22 hingga soal ke-24, dan 63,2% pada soal ke-25.

Tabel 2. Hasil Percobaan Menggunakan Metode yang Diusulkan

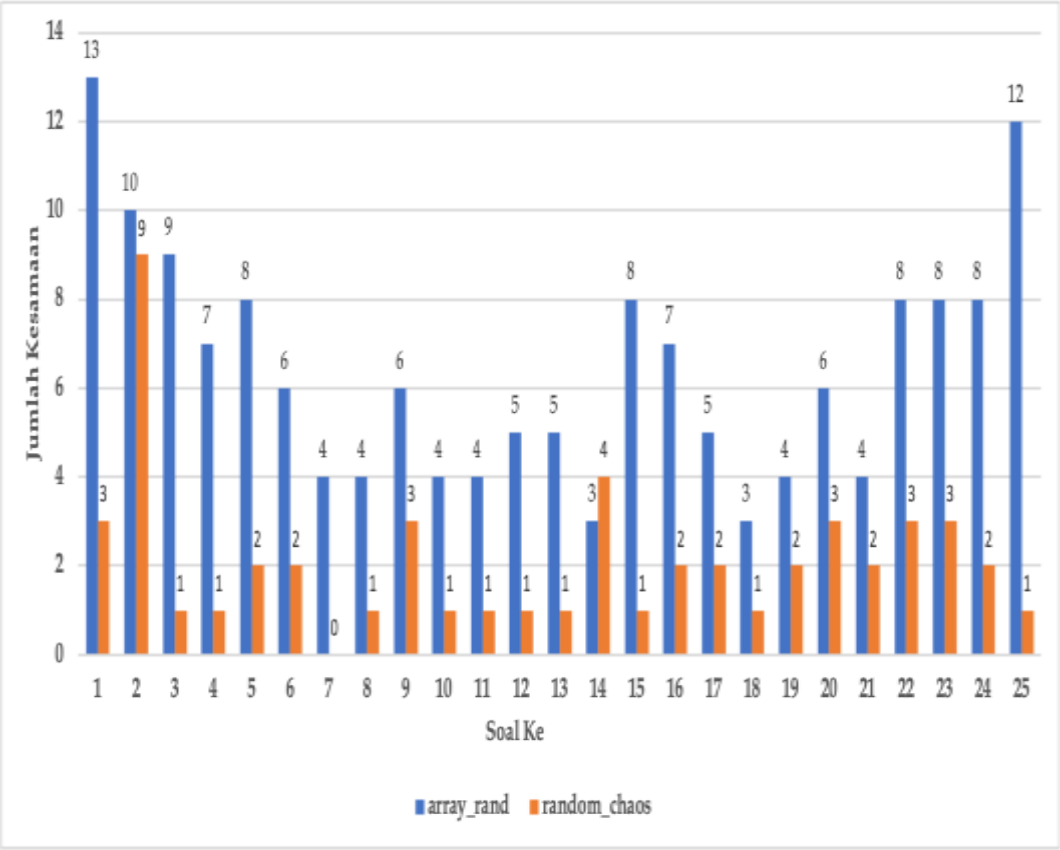
Soal Ke	Peserta Ujian																				KK	TK	PK
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20			
1	11	7	95	75	47	10	66	13	53	84	7	23	30	29	20	2	77	44	2	53	19	3	0.158
2	91	43	22	100	100	91	91	67	34	55	43	58	9	55	100	100	58	25	100	44	19	9	0.474
3	69	85	96	84	94	96	87	73	97	43	14	91	48	90	4	63	7	92	25	58	19	1	0.053
4	16	97	91	14	77	3	61	30	8	68	91	62	26	44	6	4	29	48	20	66	19	1	0.053
5	95	82	73	99	12	68	70	31	100	86	68	88	59	84	35	67	3	99	27	75	19	2	0.105
6	79	27	42	90	72	18	13	19	12	9	12	57	65	37	76	73	12	69	41	91	19	2	0.105
7	20	2	63	12	92	60	100	25	5	28	65	42	69	98	73	61	47	77	11	85	19	0	0.000
8	88	75	12	49	66	26	45	29	44	64	18	72	53	87	91	58	33	93	58	39	19	1	0.053
9	38	22	75	92	14	8	62	49	36	83	32	26	55	19	33	59	46	19	33	14	19	3	0.158
10	60	57	52	60	29	14	68	93	83	19	78	55	91	3	36	35	65	95	81	76	19	1	0.053
11	14	10	18	52	53	4	57	66	4	90	84	85	43	54	23	3	38	20	34	32	19	1	0.053
12	2	73	48	61	91	11	63	37	56	24	85	90	82	58	29	15	30	49	23	49	19	1	0.053
13	94	46	67	98	3	39	92	36	16	49	62	29	15	45	59	96	61	36	93	7	19	1	0.053
14	28	1	61	54	39	89	38	27	72	54	69	14	95	46	49	38	27	32	38	79	19	4	0.211
15	96	48	27	85	70	90	28	63	20	100	28	68	87	60	71	65	50	8	49	21	19	1	0.053
16	82	19	94	30	75	100	1	87	24	51	19	78	35	89	85	60	70	80	60	45	19	2	0.105
17	48	67	100	26	86	87	10	10	63	88	47	66	83	28	30	8	79	79	1	3	19	2	0.105
18	81	36	16	77	43	59	74	39	32	99	81	76	61	71	94	26	13	40	12	54	19	1	0.053
19	18	25	8	70	48	93	73	62	38	46	79	70	1	26	84	76	100	73	21	90	19	2	0.105
20	13	11	36	94	42	38	46	92	18	66	83	94	71	35	93	93	92	29	16	52	19	3	0.158
21	19	41	29	38	50	67	30	82	48	41	64	81	51	21	40	11	68	68	7	18	19	2	0.105
22	23	5	79	35	10	46	72	94	69	7	73	12	93	100	7	81	78	81	97	68	19	3	0.158
23	61	20	58	46	13	6	56	48	77	32	63	6	52	99	89	80	98	1	77	63	19	3	0.158
24	87	98	65	33	19	57	9	58	68	33	27	3	64	14	56	95	51	28	67	56	19	2	0.105
25	92	24	41	69	96	30	11	71	39	2	8	19	81	50	50	1	85	91	74	62	19	1	0.053
Rata-rata tingkat kesamaan (0 - 1)																							0.109

Tabel 2 menunjukkan rangkuman kemunculan soal menggunakan fungsi pengacakan *chaos* dengan metode *logistic map*. Dibandingkan dengan Tabel 1, Tabel 2 menunjukkan perbedaan yang cukup mencolok dari segi kemiripan soal yang muncul pada masing – masing peserta ujian. Rata – rata keseluruhan kemiripan soal yang muncul adalah sebesar 1,09%. Persentase kemiripan yang terjadi paling besar pada soal ke-2 sebesar 47,4%, sisanya kemiripan terjadi dibawah 2%. Pada soal ke-

3, soal ke-4, soal ke-8, soal ke-10, soal ke-11, soal ke-12, soal ke-13, soal ke-15, soal ke-18, dan soal ke-25 hanya terjadi 1 kesamaan soal yang artinya hanya terdapat dua peserta yang mendapat satu kali soal sama yang muncul pada saat ujian. Pada soal ke-7 tidak ada salah satu peserta ujian pun yang memiliki soal ujian yang sama satu sama lain. Tabel 1 dan Tabel 2 menunjukkan bahwa fungsi *chaos* dengan metode *logistic map* menunjukkan hasil pengacakan soal yang lebih baik dibandingkan fungsi *array_rand* dari sisi kesamaan soal yang muncul pada masing – masing peserta ujian di nomor yang sama.

2. Perbandingan Fungsi Pengacakan Soal

Analisis ini bertujuan untuk lebih melihat perbandingan jumlah kesamaan soal yang muncul pada masing – masing soal antara pengacakan soal menggunakan fungsi *array_rand* dengan fungsi *chaos* dengan metode *logistic map*.



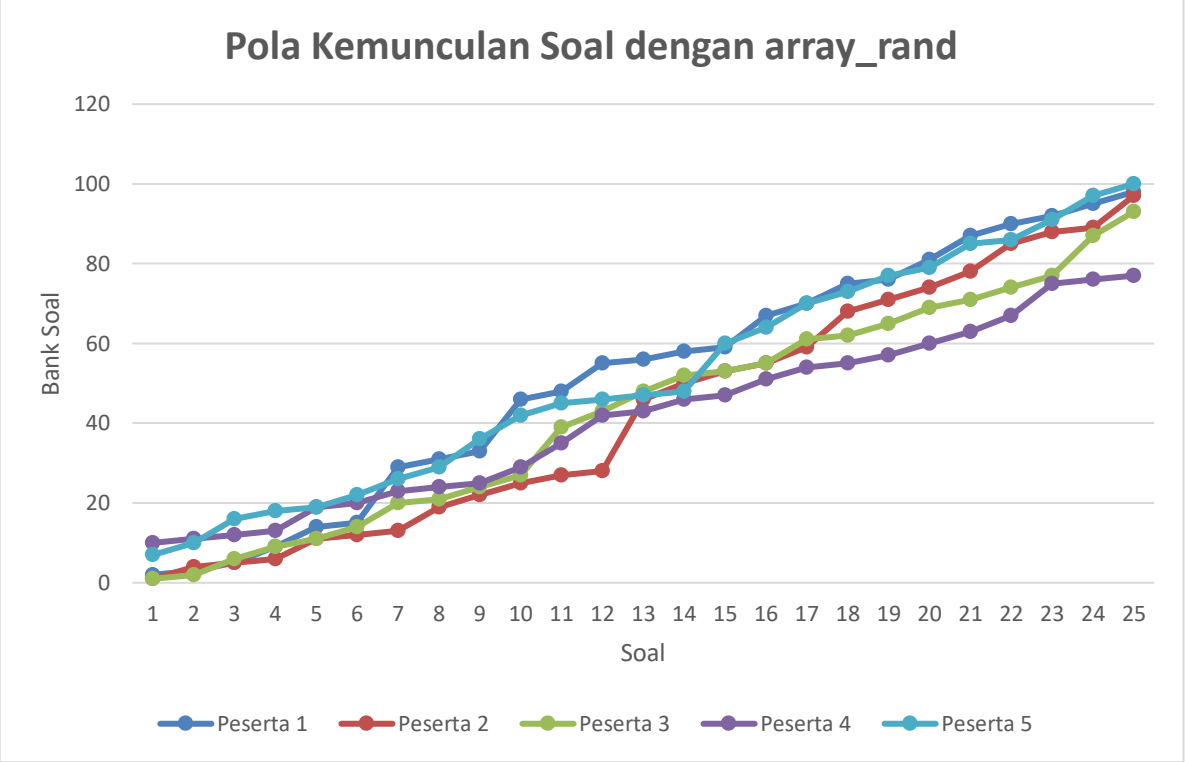
Gambar 1. Perbandingan Hasil Pengacakan Soal

Gambar 1 menunjukkan perbandingan hasil pengacakan soal menggunakan fungsi *array_rand* dengan fungsi *chaos* dengan metode *logistic map*. Dari perbandingan dapat dilihat bahwa secara umum, fungsi *array_rand* cenderung memiliki jumlah kesamaan yang lebih tinggi dibandingkan dengan fungsi *chaos* dengan metode *logistic map* di sebagian besar soal. Perbandingan yang cukup signifikan dapat dilihat pada perbandingan di soal ke-1 dan soal ke-25 dimana pada fungsi *array_rand* terjadi 13 kali kesamaan soal yang muncul sedangkan pada fungsi *chaos* dengan metode *logistic map* hanya terdapat 3 kesamaan. Pada soal ke-25

terdapat 12 kali kesamaan soal yang muncul pada fungsi *array_rand* sedangkan hanya terdapat 1 kali kesamaan soal yang muncul pada fungsi *chaos* dengan metode *logistic map*.

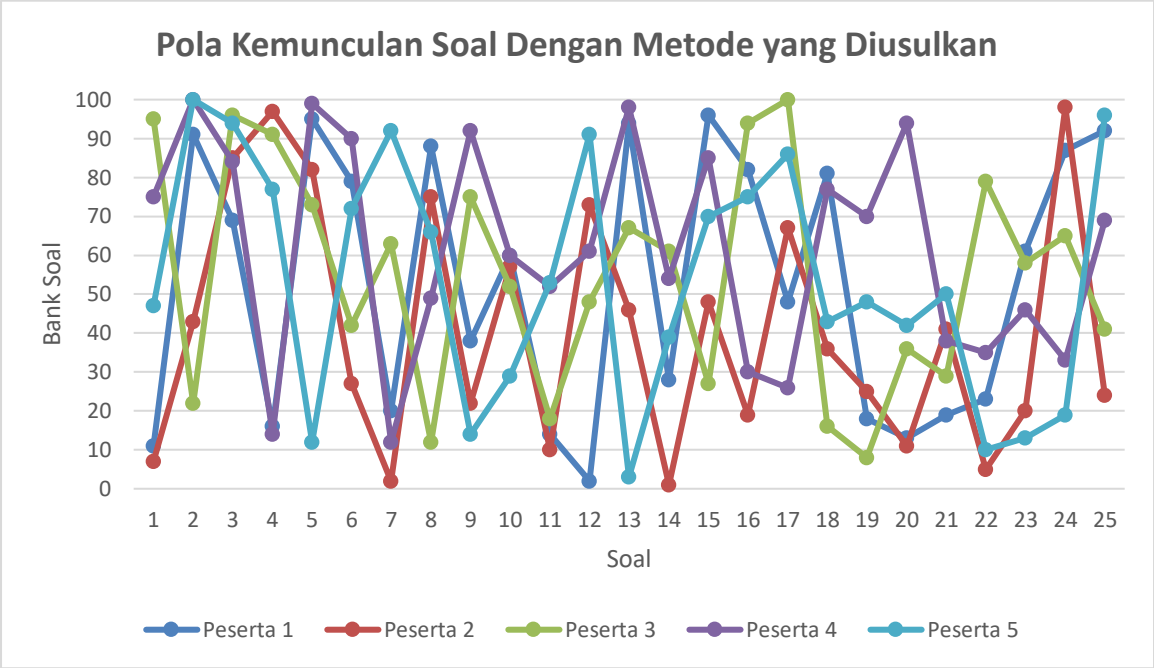
3. Pola Pengacakan Soal

Bagian ini akan menganalisis pola pengacakan soal yang dihasilkan oleh kedua fungsi yang digunakan.



Gambar 2. Hasil Pengujian Dengan Fungsi array_rand

Gambar 2 menunjukkan pola pengacakan soal menggunakan fungsi array_rand. Pada 25 soal yang diujikan, terlihat bahwa sistem akan mengeluarkan soal secara berurutan berdasarkan nomor, dimulai dari soal dengan nomor kecil hingga soal dengan nomor yang lebih besar. Artinya, dari 100 bank soal yang tersedia, soal berikutnya selalu memiliki nomor yang lebih besar daripada soal sebelumnya, sehingga tidak mungkin ada soal yang bernomor lebih kecil dari soal sebelumnya.



Gambar 3. Hasil Pengujian Dengan *chaos* dengan *logistic_map*

Berbeda dengan pola pengacakan soal yang ditunjukkan pada Gambar 3, soal-soal yang muncul bervariasi dan tidak mengikuti pola tertentu. Soal pertama dapat dimulai dari nomor besar atau kecil dalam bank soal, dan soal-soal berikutnya tidak bergantung pada soal sebelumnya. Pendekatan ini meningkatkan keamanan, karena urutan soal menjadi sulit ditebak dan tidak mengikuti pola yang dapat diprediksi.

SIMPULAN

Pengacakan soal ujian adalah metode penting dalam pendidikan untuk menjamin keadilan dan mencegah kecurangan. Dengan pengacakan yang efektif, setiap peserta ujian akan menerima urutan soal yang berbeda, sehingga mengurangi peluang terjadinya kerja sama yang tidak sah antar peserta. Penggunaan *Pseudo-Random Number Generator* (PRNG) dengan metode *Logistic Map* dalam pengacakan soal ujian terbukti efektif dalam menciptakan urutan soal yang acak dan sulit diprediksi.

Hasil pengujian menunjukkan bahwa rata-rata tingkat kesamaan soal yang muncul pada pengacakan menggunakan metode *array_rand* mencapai 33,9%, sedangkan dengan metode yang diusulkan turun signifikan menjadi 10,9%. Pola kemunculan soal dengan metode yang diusulkan lebih acak, berbeda dengan metode *array_rand* yang cenderung menghasilkan pola di mana nomor soal yang muncul selalu lebih besar daripada nomor soal sebelumnya.

Metode ini memberikan keamanan tambahan dalam pelaksanaan ujian, karena distribusi soal yang dihasilkan tidak mengikuti pola yang mudah ditebak. Dengan demikian, penerapan *Logistic Map* sebagai algoritma pengacakan soal dapat meningkatkan integritas ujian dengan mengurangi kemungkinan kecurangan serta memastikan bahwa setiap peserta ujian menerima urutan soal yang unik dan merata.

DAFTAR PUSTAKA

[1] A. Savitri, Revolusi industri 4.0: mengubah tantangan menjadi peluang di era disrupsi 4.0. Penerbit Genesis, 2019.

[2] R. Agustina, T. Rukhmana, N. Pitri, and S. Meirisa, SISTEM PENDIDIKAN DIGITAL. Cendikia Mulia Mandiri, 2023.

[3] R. Munir, Kriptografi, 1st ed. INFORMATIKA, 2006.

[4] H. Wu et al., "Design and implementation of true random number generators based on semiconductor superlattice chaos," Microelectronics J, vol. 114, p. 105119, Aug. 2021, doi: 10.1016/j.mejo.2021.105119.

[5] G. Yildirim and E. Tanyildizi, "An innovative approach based on optimization for the determination of initial conditions of continuous-time chaotic system as a random number generator," Chaos Solitons Fractals, vol. 172, p. 113548, Jul. 2023, doi: 10.1016/j.chaos.2023.113548.

[6] K. Seyhan and S. Akleylek, "Classification of random number generator applications in IoT: A comprehensive taxonomy," Journal of Information Security and Applications, vol. 71, p. 103365, Dec. 2022, doi: 10.1016/j.jisa.2022.103365.

[7] A. A. Pandit, A. Kumar, and A. Mishra, "LWR-based Quantum-Safe Pseudo-Random Number Generator," Journal of Information Security and Applications, vol. 73, p. 103431, Mar. 2023, doi: 10.1016/j.jisa.2023.103431.

[8] A. Prastya, "Kombinasi Fisher Yates Shuffle dan Bubble Sort Pada Aplikasi Ujian Berbasis Web," JATISI (Jurnal Teknik Informatika dan Sistem Informasi), vol. 9, no. 2, pp. 1738–1746, 2022, doi: 10.35957/jatisi.v9i2.1984.

[9] T. Sugihartono and R. R. C. Putra, "Penerapan Algoritma Fisher Yates untuk Pengacakan Soal Pada Sistem Ujian Kompetisi Wartawan," Infotek: Jurnal Informatika dan Teknologi, vol. 4, no. 2, pp. 238–248, 2021, doi: 10.29408/jit.v4i2.3635.

[10] M. W. Duha, A. R. Dewi, and E. Rahayu, "Online Standard Test System Based on HOTS Using Fisher-Yates Shuffle Algorithm on Final Level Students in Medan State University," CESS (Journal of Computer Engineering, System and Science), vol. 7, no. 2, pp. 442–448, doi: 10.24114/cess.v7i2.35657.

[11] F. P. Juniawan and H. Hengki, "Pengacakan Soal Ujian Penerimaan POLRI Menggunakan Algoritme Fisher Yates Shuffle," Telematika, vol. 12, no. 1, pp. 1–13, 2019, doi: 10.35671/telematika.v12i1.714.

[12] M. A. Hasan, S. Supriadi, and Z. Zamzami, "Implementasi Algoritma Fisher-Yates Untuk Mengacak Soal Ujian Online Penerimaan Mahasiswa Baru (Studi Kasus: Universitas Lancang Kuning Riau)," Jurnal Nasional Teknologi dan Sistem Informasi, vol. 3, no. 2, pp. 291–298, 2017, doi: 10.25077/TEKNOSI.v3i2.2017.291-298.

[13] T. Anna, M. A. I. Pakereng, and Y. R. Beeh, "Implementasi Algoritma Chaos-Based Feedback Stream Cipher pada Enkripsi-Dekripsi Data Citra Digital."

[14] E. Sukirman, E. Nurpeti, and D. Widya dan Wiwit Widhianto, "Penerapan Multi Logistic Map dalam Algortima Enkripsi Citra Digital."

[15] R. Arianty and D. T. Susetianingtias, "KOMBINASI LOGISTIC MAP DAN PSEUDO-RANDOM NUMBER GENERATOR PADA PEMBANGKITAN KUNCI UNTUK ENKRIPSI CITRA DIGITAL," Jurnal Ilmiah Teknologi dan Rekayasa, vol. 25, no. 3, pp. 187–198, 2020, doi: 10.35760/tr.2020.v25i3.3120.

[16] R. Gupta, P. Gupta, and J. Singh, "Security and Cryptography," in Software Engineering for Embedded Systems, Elsevier, 2019, pp. 501–547. doi: 10.1016/B978-0-12-809448-8.00014-X.