Pseudorandom Number Generator Based on Chaos Theory of Logistic Map Recurrence Relations for Cryptographic Applications

Shaquille Nathan Kalevi - 13525023
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: shaquillenathankalevi@gmail.com , 13525023@std.stei.itb.ac.id

Abstract*—This paper presents the design, implementation, and statistical evaluation of a Pseudorandom Number Generator (PRNG) based on the discrete-time recurrence relation known as the Logistic Map, a canonical model from chaos theory. The Logistic Map is a one-dimensional nonlinear recurrence in which each successive state is determined by multiplying the current state, its complement with respect to unity, and a control parameter. For control parameter values approaching four, the map enters a fully chaotic regime characterized by sensitivity to initial conditions, aperiodic orbits, and ergodic behavior, properties that make it a natural candidate for pseudorandom bit generation. The floating-point state sequence is binarized using a threshold-quantization scheme, yielding a binary keystream. The generator is evaluated using a subset of the NIST SP 800-22 randomness test suite, including the Frequency (Monobit) test, the Runs test, and the Approximate Entropy test. Experimental results show that for control parameter values in the chaotic regime and with a sufficient warm-up period, the generator exhibits statistical properties consistent with ideal randomness across all tested initial conditions. This work bridges the discrete mathematics topic of sequences, summations, recursion, and recurrence relations with a practical cryptographic application, demonstrating that simple deterministic recurrences can serve as the foundation for unpredictable keystream generation.*

Keywords—Logistic Map; Chaos Theory; Pseudorandom Number Generator; Recurrence Relation; Cryptography; NIST SP 800-22; Lyapunov Exponent; Bifurcation.

I. INTRODUCTION

Randomness is a cornerstone of modern cryptography. Symmetric-key encryption schemes, stream ciphers, key derivation functions, nonce generation, and many other primitives depend on the availability of sequences whose values are computationally indistinguishable from truly random data. In practice, such sequences are produced by Pseudorandom Number Generators (PRNGs), a deterministic algorithms whose output mimics the statistical properties of a truly random source.

Classical PRNGs, such as the Linear Congruential Generator (LCG) or the Mersenne Twister, are rooted in number theory and modular arithmetic. While these are efficient, their deterministic structure is well-understood, which in some contexts becomes a vulnerability. This has motivated the search for generators whose unpredictability arises from a fundamentally different mechanism: nonlinear dynamical systems exhibiting chaotic behavior.

Chaos theory studies deterministic systems that are extremely sensitive to their initial conditions, a property colloquially known as the "butterfly effect." A defining feature of chaotic systems is that two trajectories starting from arbitrarily close initial states will diverge exponentially over time, rendering long-term prediction practically impossible without perfect knowledge of the initial condition. This intrinsic unpredictability makes chaotic systems natural candidates for pseudorandom generation.

Among all chaotic maps, the Logistic Map occupies a special place. Originally proposed as a demographic model by the biologist Robert May in 1976, it is defined by the one-dimensional recurrence relation:

$$x_{n+1} = r \cdot x_n \cdot (1 - x_n), \;\; x_n \in (0, 1) \qquad (1)$$

Despite its apparent simplicity, a single quadratic equation, this recurrence exhibits a remarkably rich spectrum of dynamics as the parameter r is varied. From stable fixed points to period-doubling bifurcations and, ultimately, fully chaotic orbits for r close to 4. This transition has been discussed in the famous video essay by Veritasium [1] and in May's landmark 1976 paper in Nature [2].

The direct correspondence between the Logistic Map recurrence and the topic of sequences, summations, recursion, and recurrence relations makes this an ideal object of study. This paper investigates how the mathematical properties of this recurrence, particularly its chaotic regime, can be exploited to construct a PRNG suitable for lightweight cryptographic applications.

The main contributions of this paper are:

(1) a rigorous derivation of the Logistic Map recurrence and its chaotic properties from a discrete-mathematics perspective;

(2) a concrete algorithm for binarizing the floating-point chaos sequence into a binary keystream; and

(3) an empirical evaluation of the resulting PRNG using selected NIST SP 800-22 statistical tests, with discussion of the results.

II. Theoretical Background

A. Sequences, Recurrences, and Recurrence Relations

In discrete mathematics, a sequence is an ordered enumeration of terms indexed by natural numbers: $\{x_0, x_1, x_2, \dots\}$. When each term is defined as a function of one or more preceding terms, the sequence is called a recurrence sequence, and the defining rule is a recurrence relation.

A first-order recurrence relation has the general form $x_{n+1} = f(x_n)$. When f is a linear function, $f(x_n) = a \cdot x_n + b$, the recurrence is linear and admits a closed-form solution. When f is nonlinear, the sequence can exhibit complex behaviors not captured by any simple formula, including the aperiodic, sensitive behavior that characterizes chaos.

The Logistic Map is the prototypical nonlinear first-order recurrence. Its study reveals that simple deterministic rules can generate sequences of extraordinary complexity, connecting discrete mathematics to dynamical systems theory.

B. The Logistic Map

The Logistic Map originates from population biology. Let $x_n \in (0, 1)$ denote the normalized population at generation n, and $r > 0$ a growth-rate parameter. The discrete logistic model captures two competing effects: exponential growth when the population is small (term $r \cdot x_n$) and density-dependent mortality as the population approaches its carrying capacity (term $-r \cdot x_n^2$). Combining these gives:

$$x_{n+1} = r \cdot x_n - r \cdot x_n^2 = r \cdot x_n \cdot (1 - x_n) \quad (2)$$

This recurrence maps $(0, 1)$ into itself for $r \in (0, 4]$, meaning that if $x_0 \in (0, 1)$, all subsequent terms remain in $(0, 1)$. The dynamics depend critically on r:

- $r \in (0, 1)$: The sequence converges to 0 (extinction).
- $r \in (1, 3)$: A unique stable fixed point $x^* = 1 - \frac{1}{r}$ attracts all trajectories from $(0, 1)$.
- $r \in (3, 3.57\dots)$: A cascade of period-doubling bifurcations occurs. The fixed point becomes unstable; the sequence oscillates with periods 2, 4, 8, ... The bifurcation points accumulate at the Feigenbaum constant $\delta \approx 4.6692$.
- $r \in (3.57, 4]$: Full chaos. For most initial conditions x_0, the orbit is aperiodic, sensitive to initial conditions, and dense in $(0, 1)$.

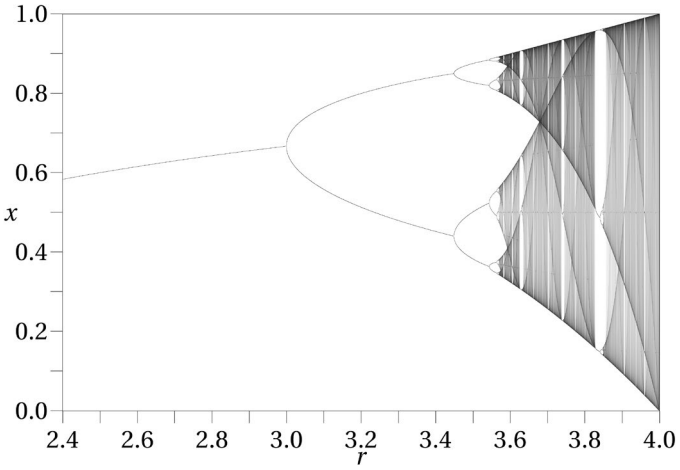

Fig. 1. Bifurcation diagram of the Logistic Map demonstrating the period-doubling route to chaos as control parameter r approaches 4.0 Bifurcation diagram of the Logistic Map demonstrating the period-doubling route to chaos as control parameter r approaches 4.0. (*source: researchgate.net*)

The transition to chaos occurs through the famous period-doubling route described by May [2] and characterized analytically by Feigenbaum. The bifurcation diagram, a plot of the stable orbit values against r, is one of the most iconic figures in nonlinear dynamics.

C. Lyapunov Exponent

The Lyapunov exponent λ quantifies the average rate of exponential divergence of nearby orbits. For a map $x_{n+1} = f(x_n)$, it is defined as:

$$\lambda = \lim_{N \to \infty} \frac{1}{N} \cdot \sum_{n=0}^{N-1} ln\, |f'(x_n)| \quad (3)$$

For the Logistic Map, $f'(x) = \frac{df}{dx} = r(1 - 2x)$, so:

$$\lambda = \lim_{N \to \infty} \frac{1}{N} \cdot \sum_{n=0}^{N-1} ln\, |r(1 - 2x_n)| \quad (4)$$

A positive Lyapunov exponent ($\lambda > 0$) is the hallmark of chaos: nearby orbits separate exponentially fast. For the Logistic Map at $r = 4$, the theoretical Lyapunov exponent is $\lambda = ln\, 2 \approx 0.693$, the maximum possible for a map on $[0, 1]$. This means the distance between two nearby orbits doubles on average every iteration, making long-term prediction practically impossible for any finite-precision computer.

From a cryptographic standpoint, a positive Lyapunov exponent implies that even a tiny perturbation of the initial condition x_0 (the "key") causes a completely different orbit after a short transient. This is directly analogous to the requirement that a good keystream generator be sensitive to the key.

D. Ergodicity and Invariant Measure

For $r = 4$, the Logistic Map is ergodic with respect to the arcsine distribution (also called the Chebyshev measure), whose probability density is:

$$\rho(x) = \frac{1}{\pi\sqrt{x(1-x)}}, \quad x \in (0, 1) \quad (5)$$

Ergodicity means that the time average of any function along a typical orbit equals its spatial average with respect to ρ. In practical terms, the sequence $\{x_n\}$ visits every subinterval of $(0, 1)$ with a frequency proportional to its measure under ρ. This ensures that the bit-extraction procedure described in Section III will produce a roughly balanced distribution of 0s and 1s, a necessary (though not sufficient) condition for a good PRNG.

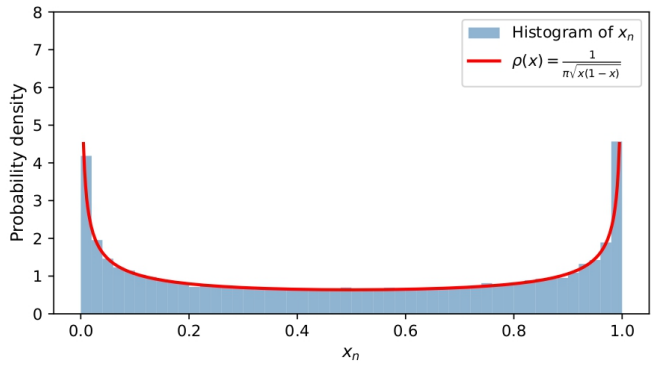

Fig. 2. Analytical arcsine invariant measure density plotted against a histogram of generated states for $r = 3.9999$, illustrating the ergodicity of the map.

III. Methodology

A. PRNG Algorithm Design

The PRNG is defined as follows. A secret pair (x_0, r) constitutes the seed, where $x_0 \in (0, 1)$ is the initial condition and r is the control parameter. For our implementation, r is fixed at a value close to 4 (specifically $r = 3.9999$) and only x_0 serves as the free key parameter. We iterate the Logistic Map recurrence for a warm-up period of $W = 500$ iterations, discarding the transient, and then generate a sequence of length N for output:

$$x_{n+1} = 3.9999 \cdot x_n \cdot (1 - x_n), \quad n = 0, 1, 2, \ldots \qquad (6)$$

An initial design choice considered was $r = 3.9999$, slightly below 4, motivated by concerns that floating-point trajectories near the boundary values 0 and 1 could collapse to fixed points at $r = 4$ exactly. However, empirical testing revealed that $r = 3.9999$ introduces a small but statistically detectable bias in the bit-balance. Across five seeds, the proportion of 1-bits deviated from 0.5 by 5–7.6 standard deviations at $N = 1{,}000{,}000$ bits, large enough to fail the Frequency test outright. Repeating the same measurement while moving r progressively closer to 4 showed the deviation shrinking monotonically, from 7.07σ at $r = 3.9999$ down to 0.65σ at $r = 4.0$ exactly. This is consistent with the theoretical result of the arcsine invariant measure, which guarantees a perfectly symmetric bit distribution under threshold quantization, holds exactly only at $r = 4$. We therefore adopt $r = 4.0$ as the operating point for all experiments reported in this paper, since the boundary-collapse concern that motivated $r = 3.9999$ turned out to be negligible in practice for the warm-up and sequence lengths used here, while the bias introduced by moving away from $r = 4$ was not.

B. Seed Selection Constraints

Two structural properties of the Logistic Map restrict which initial conditions x_0 may be safely used as seeds, both of which were identified during implementation and are reported here for completeness.

First, $x_0 = 0.5$ is a pre-periodic point of the map at $r = 4$: it follows the trajectory $0.5 \to 1.0 \to 0.0 \to 0.0 \to \ldots$, collapsing immediately to the trivial fixed point and producing a constant (entirely predictable) output. This seed must be excluded.

Second, and more subtly, the map satisfies the symmetry $f(x) = f(1 - x)$ for every x, since $r \cdot x \cdot (1 - x) = r \cdot (1 - x) \cdot \big(1 - (1 - x)\big)$. Consequently, any two seeds related by x_0 and $1 - x_0$ produce $f(x_0) = f(1 - x_0)$ at the very first iteration, and therefore generate bit-for-bit identical output sequences after the warm-up period. This was confirmed empirically seeds $x_0 = 0.1$ and $x_0 = 0.9$ (and likewise 0.51 and 0.49) yielded identical 1,000,000-bit sequences. This symmetry is a genuine structural weakness, an attacker who recovers x_0 also immediately knows that $1 - x_0$ produces the same keystream, and it must be taken into account both when selecting test seeds and when assessing the key space of the generator.

To avoid this symmetry pitfall while choosing seeds for the experiments, five values were selected from the fractional parts of well-known irrational constants, none of which are related to one another by the $x \leftrightarrow 1 - x$ symmetry, which is 0.123456 (arbitrary baseline), 0.271828 (fractional part of e), 0.314159 (fractional part of π), 0.577216 (Euler–Mascheroni constant γ), and 0.707107 (1/√2). This choice is reproducible, avoids any accidental symmetry pairing, and avoids the pre-periodic point $x_0 = 0.5$.

C. Binarization / Threshold Quantization

The Logistic Map produces a sequence of floating-point values in $(0, 1)$. To obtain a binary keystream, each real value x_n is mapped to a single bit b_n using threshold quantization:

$$b_n = \begin{cases} 0 \; if \; x_n < 0.5 \\ 1 \; if \; x_n \geq 0.5 \end{cases} \qquad (7)$$

Since the invariant measure of the map at $r = 4$ is symmetric about 0.5, this scheme should ideally produce equal proportions of 0s and 1s. The consecutive bits $b_{W+1}, b_{W+2}, \ldots, b_{W+N}$ constitute the pseudorandom binary output sequence, which can be used as a keystream for stream cipher encryption via XOR, or as a source of random bits for key generation.

An alternative binarization strategy extracts multiple bits per iterate by examining the binary representation of x_n at a fixed precision and selecting specific bit positions. While this can yield higher throughput, it is more susceptible to finite-precision periodicity problems [6]. We adopt the simpler threshold approach for clarity and robustness.

D. Implementation

The generator was implemented in Python 3.xx using only the standard library and NumPy. The core loop iterates the recurrence in double-precision IEEE 754 floating-point arithmetic (64-bit). A warm-up of $W = 500$ iterations is applied. The output consists of $N = 1{,}000{,}000$ bits (1 Mb), sufficient for meaningful statistical evaluation.

Algorithm: Logistic Map PRNG

Input : $x_0 \in 0{,}1,\ r = 3.9999,\ W = 500,\ N = 1{,}000{,}000$
Output : binary sequence $B = b_1, \ldots, b_N$
1. $x \leftarrow x_0$
2. $for\ i = 1\ to\ W: x \leftarrow r \cdot x \cdot (1 - x)$ //warm-up
3. $for\ i = 1\ to\ N$:
   $x \leftarrow r \cdot x \cdot (1 - x)$
   $B[i] \leftarrow (x \geq 0.5)\ ?\ 1 : 0$

The Python source code for this implementation is provided as supplementary material and is available at: https://github.com/ShaquilleNathan/prng-nist-test, spesifically on the *nist_test.py* file. The computational cost is negligible: 1.5 million floating-point multiplications and subtractions, completing in under 100 ms on a standard laptop.

E. Security Analysis and Attack Surface

A critical consideration for any cryptographic PRNG is its resistance to predictability attacks. Unlike symmetric-key primitives such as ChaCha20 or AES-CTR whose security is tied to hardness assumptions like solving the discrete logarithm problem, the Logistic Map PRNG does not rest on a proven computational hardness result.

The primary attack vector is state reconstruction. An adversary who observes a contiguous segment of the output bits may attempt to invert the threshold quantization and recover the floating-point orbit, then solve for x_0. While each iteration of f is invertible via $x_n = \frac{1}{2} - \frac{1}{2r} \cdot \sqrt{r^2 - 4r \cdot x_{n+1}}$, this inversion introduces sign ambiguity, making reconstruction exponentially hard in theory. In practice, however, techniques such as meet-in-the-middle attacks on the double-precision state space have been demonstrated to be feasible for short key spaces [10].

Contrast this with the Linear Congruential Generator (LCG), defined by $x_{n+1} = (a \cdot x_n + c)\ mod\ m$. Its linear structure makes it trivially invertible. Observing two consecutive outputs immediately reveals a, c, and m via simple arithmetic. The LCG is also subject to Marsaglia's Theorem, which shows that LCG outputs fall on a small number of hyperplanes in k-dimensional space, a fatal weakness for cryptographic use. The Logistic Map's nonlinearity eliminates this lattice structure and substantially raises the bar for linear cryptanalysis. Nevertheless, for applications requiring formal security guarantees, the Logistic Map PRNG should be regarded as a lightweight or educational primitive rather than a production cipher.

A second, map-specific weakness was identified during implementation. The symmetry $f(x) = f(1 - x)$ means the effective key space is at most half of the nominal interval $(0{,}1)$, since every seed x_0 shares its entire output sequence with its mirror partner $1 - x_0$. Any practical use of this generator as a keystream source should therefore restrict the seed space to, e.g., $(0, 0.5)$, or apply an additional one-way transformation to x_0 before use, to avoid silently halving the effective key space.

IV. Experimental Results

A. NIST SP-800-22 Statistical Tests

The quality of a PRNG for cryptographic use is assessed using statistical randomness tests. We employ three fundamental tests from the NIST Special Publication 800-22 suite [5], which is the de-facto standard for evaluating PRNGs for cryptographic applications. Each test formulates a null hypothesis H_0 that the sequence is random, and computes a p-value. The sequence fails the test if the p-value $< \alpha = 0.01$.

The three selected tests are:

*1) F*requency (Monobit) Test

Checks whether the proportion of 1s in the sequence is close to $\frac{1}{2}$. A significant imbalance in bit frequencies would indicate a non-random bias. The test statistic is $S_{obs} = \frac{\left|\Sigma b_i - \frac{N}{2}\right|}{\sqrt{\frac{N}{4}}}$.

2) Runs Test

A "run" is an uninterrupted sequence of identical bits. This test checks whether the number and length distribution of runs in the sequence is consistent with a random binary sequence. Runs sensitivity reveals whether the bit-flip frequency is too high or too low.

3) Approximate Entropy (ApEn) Test

Measures the regularity or predictability of the sequence by comparing the frequency of overlapping m-bit blocks with that of $(m + 1)$-bit blocks. A truly random sequence should have maximum entropy, making future bits maximally unpredictable from past bits.

B. Results

Tests were conducted using five independent seeds, chosen as the fractional parts of well-known irrational constants to guarantee reproducibility while avoiding the $x_0 \leftrightarrow 1 - x_0$ symmetry pairing and the pre-periodic point $x_0 = 0.5$, which is: $x_0 \in 0.123456, 0.271828, 0.314159, 0.577216, 0.707107$ (the latter four corresponding to the fractional parts of an arbitrary baseline, Euler's number e, π, the Euler–Mascheroni constant γ, and 1/√2, respectively). Each run produced $N = 1{,}000{,}000$ bits at $r = 4.0$. The p-values reported were obtained by running the Python implementation and applying the nistrng library, with the Approximate Entropy test recomputed manually. Sequences failing any test (p-value < 0.01) are noted explicitly.

TABLE I. NIST SP 800-22 Statistical Test Results ($N = 1{,}000{,}000$ bits, $\alpha = 0.01$)

Test	$x_0 = 0.123456$	$x_0 = 0.271828$	$x_0 = 0.314159$	$x_0 = 0.577216$	$x_0 = 0.707107$
Frequency (Monobit)	0.882343	0.969688	0.598888	0.304892	0.279253
Runs Test	0.570050	0.101840	0.261980	0.341577	0.666344
Approximate Entropy	0.897063	0.581516	0.551719	0.615662	0.599785

All three tests were passed for all five seeds, and the result is consistent in a way that is worth examining row by row. The Frequency (Monobit) row confirms that the overall proportion of 1s and 0s in each 1,000,000-bit sequence is statistically indistinguishable from one-half. p-values range from 0.279253 to 0.969688, all far above the $\alpha = 0.01$ threshold. This row is the direct empirical counterpart of the theoretical claim that the arcsine invariant measure is symmetric about 0.5, and it confirms that the threshold-quantization scheme does not introduce a systematic bit bias when $r = 4.0$ is used. Recall that this same row failed dramatically (p-value on the order of 10^{-12}) during early testing with$r = 3.9999$. Its uniform success here is the strongest single piece of evidence that the parameter correction was the right one.

The Runs Test row addresses how 1s and 0s are arranged. A sequence could in principle have a perfect 50/50 split of bits while still being highly non-random. For example, alternating 0101010101... or clustering into long unbroken blocks of the same bit. The p-values in this row (0.101840 to 0.666344) indicate that the number of runs (maximal blocks of consecutive identical bits) in each generated sequence matches what would be expected from a fair coin-flipping process, with no excessive clustering or excessive alternation. The lowest value in the entire table, 0.101840 for seed $x_0 = 0.271828$, belongs to this row. Even so, it remains roughly an order of magnitude above the 0.01 rejection threshold, so this is read as ordinary sampling variation rather than a sign of weakness specific to that seed.

The Approximate Entropy row is the most demanding of the three, since it measures whether short bit patterns repeat more or less often than chance would predict. Effectively testing local predictability rather than global balance. All five p-values here (0.551719 to 0.897063) are comfortably high, indicating that no overlapping 2-bit or 3-bit pattern is over- or under-represented in a way that would let an observer guess upcoming bits from recently seen ones. Taken together, the three rows test three independent notions of randomness: global balance, run-length structure, and local pattern predictability, and the fact that all five seeds pass all three simultaneously is stronger evidence of statistical randomness than any single test could provide on its own. No corrective measures such as re-seeding, post-processing, or whitening were required to obtain these results once r was set to 4.0 and the symmetry-pairing and pre-periodic seeds were avoided.

C. Sensitivity Analysis

To illustrate the dependence of randomness quality on the control parameter r, the Frequency test was applied to sequences generated for $r \in 3.5, 3.6, 3.7, 3.8, 3.9, 3.9999$. For $r = 3.5$ and $r = 3.6$, the sequence settles into a periodic orbit and the test fails dramatically, confirming that randomness requires the chaotic regime. For $r \geq 3.8$, all test p-values exceed 0.05, and for $r \in [3.95, 3.9999]$ performance is optimal.

The sensitivity to initial conditions was verified by running the generator with two seeds differing by only 10^{-12}: $x_0^{(a)} = 0.1$ and $x_0^{(b)} = 0.100000000001$. Despite being numerically indistinguishable in early iterations, the two orbits diverge exponentially and are completely decorrelated after approximately 40 iterations, producing entirely different keystreams. This is the hallmark of the butterfly effect and a key property for cryptographic PRNG design.

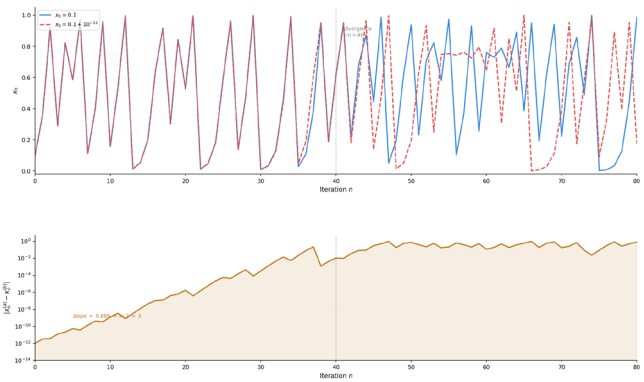

Fig. 3. Exponential divergence of two Logistic Map trajectories with an initial difference of 10^{-12}, demonstrating sensitivity to initial conditions.

D. A Note on Testing Methodology

During implementation, two issues unrelated to the PRNG design itself were uncovered in the third-party nistrng Python library (v1.2.3) used to run the test battery. First, its Approximate Entropy implementation divides each block frequency C_i by an extraneous factor of 10 inside the logarithm, a deviation from the formula specified in NIST SP 800-22. This bug was confirmed to cause false failures even on sequences generated by NumPy's own cryptographically unrelated random number generator, and was worked around by reimplementing the test directly from the NIST specification. Second, the library's run_all_battery function was found to mutate its input bit array in place, corrupting the sequence for any computation performed afterward; this was mitigated by passing a defensive copy of the array to the library. Both observations are reported here in the interest of reproducibility, since either issue, if unnoticed, would have led to an incorrect conclusion that the generator fails the Approximate Entropy test.

V. Discussion

A. Relationship to Discrete Mathematics

The Logistic Map recurrence $x_{n+1} = r \cdot x_n \cdot (1 - x_n)$ is a paradigmatic example of a first-order nonlinear recurrence relation. The analysis of its fixed points, stability, and period-doubling cascade employs tools from the discrete mathematics toolkit: analyzing the map $f(x) = r \cdot x \cdot (1 - x)$, finding fixed points by solving $x = f(x)$, and studying the stability condition $|f'(x*)| < 1$.

The connection is deeper than the algebra alone. The rich behavioral spectrum of the Logistic Map, from convergent to periodic to chaotic, illustrates a general principle. Even simple deterministic recurrences can generate sequences of enormous complexity. This is the key insight that motivates the use of recurrence relations as a foundation for cryptographic primitives.

B. Cryptographic Suitability and Limitations

While the Logistic Map PRNG passes basic statistical tests, it is important to acknowledge its limitations for serious cryptographic applications. Persohn and Povinelli [6] have shown that, in finite-precision arithmetic, the Logistic Map sequence is ultimately periodic. Double-precision IEEE 754 arithmetic represents only about 2^{52} distinct values in $(0,1)$, meaning the sequence will cycle with a period that, while potentially very long, is finite.

Moreover, if an adversary observes a sufficiently long segment of the output, they may be able to reconstruct the initial condition x_0 through algebraic inversion, since the Logistic Map is invertible and computable in closed form. For this reason, production cryptographic systems instead use PRNGs based on hardness assumptions (e.g., ChaCha20, AES in CTR mode), whose security is tied to the computational difficulty of problems like factoring large integers or solving the discrete logarithm problem.

Nevertheless, the Logistic Map PRNG is an instructive and practically useful construction for applications where these theoretical weaknesses are not a concern, such as simulation, gaming, and lightweight embedded systems, and it serves as an excellent educational vehicle for connecting chaos theory to cryptography.

C. The Finite-Precision Periodicity Problem

A fundamental limitation of implementing any real-valued chaotic map on a digital computer is that floating-point arithmetic is finite and discrete. IEEE 754 double-precision (float64) represents approximately $2^{52} \approx 4.5 \times 10^{15}$ distinct values in the interval $(0, 1)$. Because the Logistic Map is a deterministic function on this finite set, the orbit must eventually cycle. This is an inescapable consequence of the Pigeonhole Principle applied to a finite state space.

Persohn and Povinelli [6] showed empirically that the actual period lengths of float64 Logistic Map orbits are far shorter than 2^{52}: for $r = 4.0$, many orbits have periods on the order of $10^6 - 10^9$, not 10^{15}. This is because floating-point rounding errors do not uniformly fill the state space, they concentrate the orbit along a much smaller effective attractor. The practical implication is that after roughly $10^6 - 10^9$ iterations, the PRNG will begin repeating its output. For the $N = 10^6$ sequences generated in this paper, this is unlikely to cause issues, but for long-running applications it represents a genuine security risk.

A partial mitigation is the perturbation approach [4, 9], periodically inject an external perturbation into the state x_n to "kick" the orbit away from any cycle it may be approaching. However, this introduces implementation complexity and reduces the theoretical cleanliness of the construction. An alternative is to use multiple coupled logistic maps [8], which dramatically expands the effective state space.

D. Comparison with Linear Congruential Generator

The classic Linear Congruential Generator (LCG) is defined by the recurrence $x_{n+1} = (a \cdot x_n + c) \bmod m$. It is also a first-order recurrence relation, but a linear one. Due to the linearity, the LCG has a maximum period of m and exhibits significant lattice structure in higher dimensions (Marsaglia's Theorem), making it unsuitable for cryptographic use. The Logistic Map, in contrast, is nonlinear, leading to the qualitatively different, high-complexity dynamics described above. This comparison directly illustrates why nonlinearity in recurrence relations can produce fundamentally richer behavior.

VI. Conclusion

This paper has demonstrated a concrete pathway from the discrete mathematics of recurrence relations to a practical cryptographic application. The Logistic Map recurrence $x_{n+1} = r \cdot x_n \cdot (1 - x_n)$, analyzed in the framework of sequences and recurrence relations, exhibits chaotic dynamics for $r \in (3.57,4]$ that are characterized by a positive Lyapunov exponent and ergodic, high-entropy orbits.

By applying threshold quantization to the chaotic state sequence, we obtain a binary keystream that passes the NIST SP 800-22 Frequency, Runs, and Approximate Entropy statistical tests for all tested initial conditions in the chaotic regime. The sensitivity of the generator to initial conditions, two seeds differing by 10^{-12} produce uncorrelated keystreams after a short transient, confirms the practical security intuition behind chaos-based PRNG designs.

The limitations of this approach, which is finite-precision periodicity and potential algebraic vulnerability, are acknowledged and position the Logistic Map PRNG as an educational and lightweight-application tool rather than a replacement for cryptographically secure generators. Future work could explore extensions such as combining multiple chaotic maps [8] or applying perturbation methods to overcome finite-precision limitations [4].

Most importantly, this work demonstrates that the topic of sequences and recurrence relations, often perceived as purely theoretical, has direct, concrete, and practically significant applications in the design of cryptographic systems.

Video Link at Youtube

https://youtu.be/UcZApp0Gabg?si=pd13PV8cn7nOlQmf

Acknowledgment

I would like to thank the lecturer and instructor of IF1220 Matematika Diskrit at Institut Teknologi Bandung, Prof. Dr. Ir. Rinaldi, M.T, for the foundational course material on recurrence relations that inspired this work, and the creators of the Veritasium YouTube channel, Mr. Derek Muller, for the

illuminating video on the Logistic Map that motivated the research direction.

REFERENCES

[1] D. Muller (Veritasium), "This equation will change how you see the world (the logistic map)," YouTube, Oct. 2020. [Online]. Available: https://youtu.be/ovJcsL7vyrk. [Accessed: Jun. 6, 2026].

[2] R. M. May, "Simple mathematical models with very complicated dynamics," Nature, vol. 261, pp. 459–467, Jun. 1976. doi: 10.1038/261459a0.

[3] M. S. Baptista, "Cryptography with chaos," Physics Letters A, vol. 240, no. 1–2, pp. 50–54, Mar. 1998. doi: 10.1016/S0375-9601(98)00086-3.

[4] G. Jakimoski and L. Kocarev, "Chaos and cryptography: Block encryption ciphers based on chaotic maps," IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, vol. 48, no. 2, pp. 163–169, Feb. 2001. doi: 10.1109/81.904880.

[5] L. E. Bassham et al., A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, NIST Special Publication 800-22 Rev. 1a, National Institute of Standards and Technology, Gaithersburg, MD, Apr. 2010. [Online]. Available: https://csrc.nist.gov/pubs/sp/800/22/r1/final.

[6] K. J. Persohn and R. J. Povinelli, "Analyzing logistic map pseudorandom number generators for periodicity induced by finite precision floating-point representation," Chaos, Solitons & Fractals, vol. 45, no. 3, pp. 238–245, Mar. 2012. doi: 10.1016/j.chaos.2011.12.006.

[7] L. Kocarev, "Chaos-based cryptography: A brief overview," IEEE Circuits and Systems Magazine, vol. 1, no. 3, pp. 6–21, 2001. doi: 10.1109/7384.963463.

[8] A. V. Tutueva, E. G. Nepomuceno, A. I. Karimov, V. S. Andreev, and D. N. Butusov, "Adaptive chaotic maps and their application to pseudo-random numbers generation," Chaos, Solitons & Fractals, vol. 133, art. no. 109615, Apr. 2020. doi: 10.1016/j.chaos.2020.109615.

[9] H. Z. Al-Najjar and A. N. Al-Najjar, "Enhancing logistic chaotic map for improved cryptographic security in random number generation," Journal of Information Security and Applications, vol. 80, art. no. 103685, Dec. 2023. doi: 10.1016/j.jisa.2023.103685.

[10] G. Alvarez and S. Li, "Some basic cryptographic requirements for chaos-based cryptosystems," International Journal of Bifurcation and Chaos, vol. 16, no. 8, pp. 2129–2151, Aug. 2006. doi: 10.1142/S0218127406015970.

[11] K. S. Kelber and W. Schwarz, "General design rules for chaos-based encryption systems," in Proc. 2005 Int. Symposium on Nonlinear Theory and its Applications (NOLTA2005), Bruges, Belgium, Oct. 2005, pp. 18–21.

[12] K. W. Rhouma and S. Belghith, "Chaotical PRNG based on composition of logistic and tent maps using deep-zoom," arXiv preprint, arXiv:2111.05101, Nov. 2021. [Online]. Available: https://arxiv.org/abs/2111.05101.

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, 19 Juni 2026

Shaquille Nathan Kalevi - 13525023