ADBA Chaos and Fractals
Cutting-Edge Scientific Solutions

e-ISSN: 3023-8951
REVIEW ARTICLE
Vol.3 / No.2 / 2026 / pp.125-136
https://doi.org/10.69882/adba.chf.2026076

From Deterministic Chaos to Machine Learning: A Comparative Review of Pseudo-Random Number Generators for Modern Computing and Cryptography

Mutala Abdul Karim [*,1], **Peter Awon-natemi Agbedemnab** [β,2] **and Salamudeen Alhassan** [β,3]

[*]Department of Mathematics, C. K. Tedam University of Technology and Applied Sciences, Navrongo, Ghana, [β]Department of Information Systems and Technology, C. K. Tedam University of Technology and Applied Sciences, Navrongo, Ghana.

ABSTRACT Pseudo-random number generators are central to simulation, cryptography, embedded systems, privacy-preserving computation, and machine-learning workflows. This review compares traditional, chaos-based, hybrid, residue-number-system-based, and machine-learning-assisted approaches using computational efficiency, period length, entropy, autocorrelation, statistical-test performance, scalability, and cryptographic suitability. Traditional generators such as linear congruential generators, linear feedback shift registers, Xorshift, and the Mersenne Twister remain attractive for high-throughput simulation because they are simple, reproducible, and fast. Their linearity and recoverable internal states, however, limit their suitability for security-sensitive applications. Chaos-based generators improve nonlinearity, sensitivity to initial conditions, and entropy, but their performance depends strongly on parameter tuning and finite-precision implementation. Hybrid and residue-number-system designs offer a promising middle ground by combining modular arithmetic, nonlinear dynamics, and parallel residue computation. Machine-learning techniques further expand generator evaluation and design by detecting hidden bias, learning complex output patterns, and supporting adaptive generation, although they introduce training cost, reproducibility concerns, and hardware requirements. Future research should prioritize secure hybrid architectures, precision-aware chaotic maps, adaptive reseeding, hardware acceleration, and standardized testing frameworks for nonlinear and data-driven randomness systems.

KEYWORDS
Pseudo-random number generators
Cryptography
Chaotic maps
Machine learning
Hybrid generators
Randomness testing
Residue number system
Security

INTRODUCTION

Random number generation is a foundational requirement in modern computing. It supports simulation, statistical sampling, secure key generation, cryptographic protocols, randomized algorithms, gaming, embedded security, and machine-learning workflows (Maheshwari *et al.* 2014; Noibate 2023). In practice, random number generators are commonly grouped into true random number generators (TRNGs), which derive uncertainty from physical processes, and pseudo-random number generators (PRNGs), which produce deterministic sequences from mathematical rules and initial seeds.

PRNGs are especially important because they are fast, reproducible, and easy to implement in software or hardware. Their deterministic structure allows the same seed to reproduce the same sequence, which is useful in simulation and experimental repeatability (Maheshwari *et al.* 2014; Martini and Bruno 2017). This same determinism creates security risks when internal states, recurrence rules, or seeds can be inferred from observed outputs. Therefore, a useful PRNG must balance statistical randomness, computational cost, scalability, reproducibility, and resistance to prediction.

This review compares five major families of PRNG-related methods: traditional arithmetic and bitwise generators, cryptographically secure PRNGs, chaos-based generators, hybrid/residue-number-system designs, and machine-learning-assisted approaches. Rather than treating description and comparison as separate stages, Section embeds a short critical synthesis immediately after each family is introduced, so that similarities, differences, advantages, and limitations are visible where the algorithms are first discussed. Section consolidates these observations into an explicit, criterion-by-criterion comparative analysis, Section discusses the resulting trade-offs and future directions, and Section concludes the review.

Manuscript received: 16 June 2026,
Revised: 14 July 2026,
Accepted: 14 July 2026.

[1]mutalaak@gmail.com (**Corresponding author**)
[2]pagbedemnab@cktutas.edu.gh
[3]salhassan@cktutas.edu.gh

The evaluation criteria used throughout are period length, entropy, autocorrelation, NIST/Diehard test performance, throughput, memory requirement, hardware suitability, and cryptographic applicability. The central argument is that no single generator is universally optimal; the strongest design depends on the target application and the acceptable trade-off between speed, security, and implementation complexity.

Figure 2 presents a taxonomy of all PRNG families discussed in this review, and it is referenced throughout Section as each branch is introduced in turn.

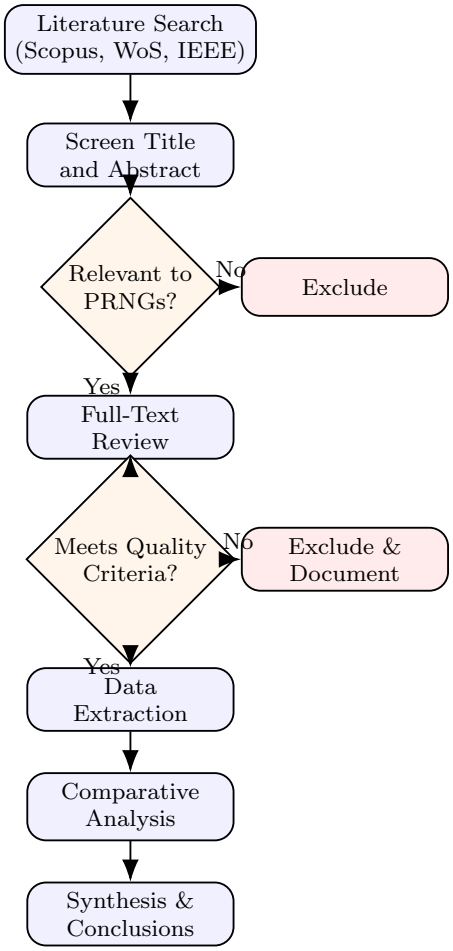

Figure 1 Review methodology flowchart. Articles retrieved from Scopus, Web of Science, and IEEE Xplore were screened by title and abstract, assessed for relevance, quality-filtered, then data-extracted for comparative analysis

The literature underlying this review was retrieved from Scopus, Web of Science, and IEEE Xplore, and was screened in the four stages summarized in Figure 1: an initial title/abstract screen for relevance to PRNGs, a full-text review, a quality-criteria filter, and a final data-extraction step feeding the comparative analysis and synthesis presented in Sections and . Records that failed the relevance screen or the quality filter were excluded and, where applicable, the reason for exclusion was documented. This staged process produced the reference set summarized in Tables 4 and 5 and discussed in Section .

LITERATURE REVIEW

Traditional PRNG Approaches

Traditional PRNGs, shown on the left branch of Figure 2, were developed mainly for numerical simulation, Monte Carlo analysis, and general-purpose computation. They rely on fixed recurrence relations, modular arithmetic, bitwise transformations, or finite-field operations rather than adaptive learning. Their main advantages are simplicity, speed, and low memory cost, while their main weaknesses are predictability and limited resistance to state-recovery attacks (Noibate 2023; Saeed *et al.* 2024).

Linear Congruential Generator (LCG): The LCG is a simple and popular approach to producing a string of pseudo-random numbers, defined by

$$X_{n+1} = (aX_n + c) \bmod m. \tag{1}$$

The series depends on an initial seed X_0; the Hull–Dobell theorem establishes conditions for achieving the longest period m, and Marsaglia showed that LCG outputs lie on a finite number of parallel hyperplanes, a structural weakness exploited in state-recovery attacks (Martini and Bruno 2017; Noibate 2023).

Quadratic Congruential Generator (QCG): The QCG is a nonlinear extension of the LCG,

$$X_{n+1} = (aX_n^2 + bX_n + c) \bmod m, \tag{2}$$

where $m > 0$ is the modulus and a, b, and c are integer parameters (Noibate 2023; Sayed *et al.* 2016).

Lagged Fibonacci Generator (LFG):

$$X_n = (X_{n-j} \oplus X_{n-k}) \bmod m, \tag{3}$$

with $0 < j < k$ integer lags and the initial k values forming the seed vector.

Mersenne Twister (MT19937): Developed by Matsumoto and Nishimura in 1998, MT19937 produces 32-bit integers with period $2^{19937} - 1$ (Martini and Bruno 2017; Oliveira 2024), with recurrence over GF(2):

$$X_{n+1} = f(X_n, X_{n+1}, \ldots, X_{n+k}). \tag{4}$$

Wichmann–Hill Combined LCG (CLCG):

$$X_n = \left(\sum_{i=1}^{r} a_i X_n \bmod m_i \right) \bmod 1, \tag{5}$$

$$U_n = \left(\sum_{i=1}^{r} \frac{X_{n,i}}{m_i} \right) \bmod 1, \tag{6}$$

where r is the number of combined generators, a_i the multiplier, and m_i the modulus of generator i.

Middle Square Method: One of the earliest pseudo-random methods, it squares a four-digit seed to produce an eight-digit number and uses the middle four digits as the next seed. It has a very short period ($< 10^4$) and is unsuitable for large-scale simulation (Martini and Bruno 2017; Saeed *et al.* 2024).

Xorshift Generator:

$$X_i = X_{i-1} \oplus (X_{i-1} \ll a) \oplus (X_{i-1} \gg b) \oplus (X_{i-1} \ll c), \tag{7}$$

with $\oplus$ denoting bitwise exclusive OR and $\ll$, $\gg$ denoting left and right bit-shifts.

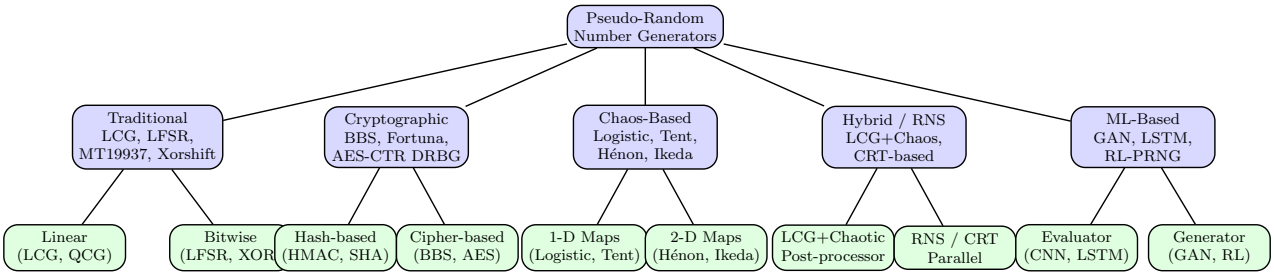

Figure 2 Taxonomy of PRNG families and sub-types. Five major families branch from the root: traditional, cryptographic, chaos-based, hybrid/residue-number-system, and machine-learning-based. Each family is further decomposed into its principal sub-types. Machine-learning-based generators serve both generative and evaluative roles in PRNG research

Linear Feedback Shift Register (LFSR):

$$b_i = c_1 b_{i-1} \oplus c_2 b_{i-2} \oplus \cdots \oplus c_N b_{i-N}, \tag{8}$$

with binary tap coefficients $c_1, \ldots, c_N \in \{0, 1\}$.

Synthesis: Table 1 consolidates the recurrence, period, speed, and security profile of every traditional generator discussed above. Throughput is highest exactly where the recurrence is linear and bitwise, as in Xorshift and LFSR, and this same linearity is what a Berlekamp–Massey-style analysis or hyperplane analysis exploits to recover the internal state. None of the generators in this family should therefore be used unmodified for key generation or nonce derivation. MT19937 stands out for its exceptionally long period and strong empirical statistics, but its state is fully recoverable from 624 consecutive 32-bit outputs, so its suitability is restricted to simulation and non-adversarial sampling. This trade-off motivates the cryptographic designs discussed next.

Cryptographically Secure PRNGs

Where traditional generators fail is exactly where cryptographically secure PRNGs are designed to succeed: unpredictability under adversarial observation.

Blum–Blum–Shub (BBS): Proposed in 1986, BBS is founded on the hardness of integer factorization, the same hard problem underlying RSA:

$$X_{n+1} = X_n^2 \bmod M, \qquad M = pq, \tag{9}$$

with p and q large safe primes.

Fortuna: Described by Ferguson and Schneier in 2003, Fortuna improves on Yarrow by addressing entropy-estimation flaws and providing perpetual reseeding from multiple entropy sources using AES-based entropy-accumulation pools.

Yarrow: Invented by Schneier, Kelsey, and Ferguson in the late 1990s, Yarrow applies cryptographic post-processing through block ciphers and hash functions to produce sequences that remain unpredictable in a hostile environment (Almaraz Luengo and Román Villaizán 2023).

ANSI X9.17 (1985) / X9.31 (1998): This family uses block ciphers such as Triple-DES or AES in feedback mode (Oliveira 2024):

$$X_n = E_k(DT) \oplus R_{n-1}, \tag{10}$$

where DT is a date/time value and E_k is encryption under key k.

AES-CTR DRBG, HMAC-DRBG, and SHA-256 Hash DRBG: These modern deterministic random-bit generators are used throughout current cryptographic systems (Guisande *et al.* 2023). AES-CTR DRBG encrypts successive counter values,

$$R_i = E_k(V + i), \tag{11}$$

HMAC-DRBG replaces encryption with a keyed hash,

$$V_i = \mathrm{HMAC}_k(V_{i-1}), \quad K = \mathrm{HMAC}_k(V_i \| \mathrm{entropy}), \tag{12}$$

and Hash-DRBG applies a cryptographic hash to a seed-derived state,

$$R_i = \mathrm{Hash}(V + i), \quad V \leftarrow \mathrm{Hash}(V \| \mathrm{const} \| \mathrm{entropy}). \tag{13}$$

Synthesis: As the cryptographic rows of Table 1 show, every cryptographically secure design trades throughput for security. Each relies on either a hard number-theoretic problem, a block cipher, or a keyed hash precisely to remove the linear recoverability that limits traditional generators, at the cost of lower speed.

Background: TRNGs and the Physical Basis of Randomness

True random numbers derive from unpredictable physical processes rather than deterministic algorithms; unlike PRNGs, true randomness comes from stochastic natural phenomena (Martini and Bruno 2017). Table 2 summarizes the principal TRNG source types, their entropy rates, and typical applications.

Table 2 shows a clear entropy ordering: quantum-mechanical sources such as photon detection, quantum tunnelling, and radioactive decay reach the highest entropy classification because their underlying physical models are fundamentally stochastic rather than merely complex. Classical noise sources and oscillator/metastability-based sources are rated lower because residual correlations or environmental coupling can, in principle, be modeled. Environmental sources such as keystrokes and mouse movement are the weakest and are typically used only to seed a PRNG or cryptographically secure generator. This entropy hierarchy provides the physical justification for chaos-based and hybrid PRNGs, which attempt to approximate high-entropy behavior algorithmically when a physical source is unavailable or too slow.

Chaos-Based PRNGs

Chaos maps generate complex, seemingly random behavior from deterministic systems. Despite being governed by simple equations, they exhibit sensitivity to initial conditions, the "butterfly effect" making them useful in cryptography, random number generation, and nonlinear-dynamics research (Saeed *et al.* 2024).

Table 1 Comparative study of traditional and related pseudo-random number generator approaches

PRNG type	Application	Method	Equation/operation	Period	Speed	Security
LCG	Simulation	Modular linear	$X_{n+1} = (aX_n + c) \bmod m$	Up to m	Fast	Low
QCG	Simulation	Quadratic modular	$X_{n+1} = (aX_n^2 + bX_n + c) \bmod m$	Up to m	Moderate	Low
LFG	Monte Carlo	Recurrence	$X_n = (X_{n-j} \oplus X_{n-k}) \bmod m$	$> 10^{17}$	Moderate	Low
MT19937	Scientific	Recurrence over GF(2)	$X_{n+1} = f(X_n, \ldots, X_{n+k})$	$2^{19937} - 1$	Fast	Low
Wichmann–Hill	Simulation	Multi-modular	$U_n = (\sum X_{n,i}/m_i) \bmod 1$	$> 10^{18}$	Moderate	Low
Middle Square	Historical	Digit extraction	Middle digits of X_n^2	$< 10^4$	Slow	Very low
Xorshift	Simulation, GPU	XOR and shift	$X_i = X_{i-1} \oplus (\cdots)$	$2^{128} - 1+$	Very fast	Low
LFSR	Hardware, communications	Shift and feedback	$b_i = c_1 b_{i-1} \oplus \cdots$	$2^n - 1$	Very fast	Low
BBS	Cryptography	Modular squaring	$X_{n+1} = X_n^2 \bmod M$	$\sim 2^n$	Slow	Very high
Fortuna	Cryptography	AES and entropy	AES-based reseeding	Extremely long	Moderate	Very high
Yarrow	Cryptography	Entropy and reseeding	Block-cipher pools	Extremely long	Moderate	Very high
ANSI X9.17/31	Financial	AES and timestamp	$X_n = E_k(DT) \oplus R_{n-1}$	Extremely long	Moderate	Very high
AES-CTR DRBG	Cryptography	Hash/cipher	$R_i = E_k(V + i)$	Extremely long	Moderate	Very high
Chaotic PRNGs	Cryptography, research	Chaotic dynamics	$x_{n+1} = rx_n(1 - x_n)$	Parameter-dependent	Moderate	High
Hybrid PRNGs	Cryptography, simulation	Arithmetic and chaos	LCG $\oplus$ chaotic map	Very long	Moderate	High
RNS-based	Cryptography, HPC	Multi-modular	CRT reconstruction	Very long	Moderate	High

Table 2 Comparative overview of true random number generator sources. Entropy rate is relative to the class of generators considered

Type	Randomness source	Generation method	Mathematical model	Entropy	Applications
Thermal noise	Electron thermal agitation	Digitize analog voltage noise	$V_n^2 = 4kTR\Delta f$	High	Hardware RNGs; secure chips
Shot noise	Discrete electron flow	Current fluctuation detection	$I_n^2 = 2qI\Delta f$	High	Embedded security hardware
Avalanche noise	Reverse-biased diode breakdown	Amplify and digitize diode noise	$I_n = \sqrt{2qI_a\Delta f}\, G$	Very high	TPM modules; crypto hardware
Photonic TRNG	Photon detection/quantum uncertainty	Arrival times or polarization	$P(n) = \lambda^n e^{-\lambda}/n!$	Extremely high	Quantum cryptography
Quantum tunnelling	Electron tunnelling through barriers	Tunnelling-current measurement	$T = e^{-2\kappa d}$	Extremely high	Quantum key generation
Radioactive decay	Nuclear decay events	Inter-arrival-time sensors	$P(t) = \lambda e^{-\lambda t}$	Extremely high	Scientific randomness
Oscillator jitter	Clock phase noise	Jitter between asynchronous oscillators	$\Delta t_n = \phi_n/(2\pi f)$	Moderate	Microcontrollers; FPGAs
Metastability	Unstable flip-flop states	Latch uncertain state and sample	$\tau = \tau_0 e^{\Delta V/V_t}$	Moderate	FPGA; IoT devices
SRAM startup	Initial memory-cell states	Read startup bits after power-on	$P(1) = \frac{1}{2} + \delta, \delta \to 0$	Moderate	Embedded security; IoT
Environmental	Keystrokes, mouse, weather	Sample real-world events	$H = -\sum p_i \log_2 p_i$	Low–moderate	Seeding; monitoring; simulation

Representative Chaotic Maps

Tent Map:

$$x_{n+1} = \begin{cases} \mu x_n, & x_n < \frac{1}{2}, \\ \mu(1 - x_n), & x_n \geq \frac{1}{2}, \end{cases} \qquad \mu \in [0, 2]. \tag{14}$$

(Sayed *et al.* 2016).

Ikeda Map:

$$x_{n+1} = 1 + \mu(x_n \cos t_n - y_n \sin t_n), \tag{15}$$
$$y_{n+1} = \mu(x_n \sin t_n + y_n \cos t_n), \tag{16}$$

where $t_n = 0.4 - 6/(1 + x_n^2 + y_n^2)$ and $\mu \approx 0.9$ (Oliveira 2024).

Logistic Map:

$$x_{n+1} = rx_n(1 - x_n), \qquad x_n \in [0, 1], \quad r \in [0, 4]. \tag{17}$$

For $r > 3.57$ the map exhibits deterministic chaos (Aniszewska 2018).

Tinkerbell Map:

$$x_{n+1} = x_n^2 - y_n^2 + ax_n + by_n, \tag{18}$$
$$y_{n+1} = 2x_n y_n + cx_n + dy_n, \tag{19}$$

with control parameters $a = 0.9$, $b = -0.6013$, $c = 2$, and $d = 0.5$ (Sathya *et al.* 2021).

Hénon Map:

$$x_{n+1} = 1 - ax_n^2 + y_n, \qquad y_{n+1} = bx_n, \tag{20}$$

with $a = 1.4$ and $b = 0.3$ (Guisande *et al.* 2023).

Chirikov Standard Map:

$$P_{n+1} = P_n + K\sin(\theta_n) \bmod 2\pi, \tag{21}$$
$$\theta_{n+1} = \theta_n + P_{n+1} \bmod 2\pi, \tag{22}$$

where θ_n is the angular position, P_n the conjugate momentum, and K the nonlinearity parameter (Silva and Prado 2023).

Discussion of Table 3: No single map dominates across all criteria. Logistic and Tent maps are preferred where speed and implementation simplicity matter most, but both are known to collapse to short cycles under finite-precision arithmetic and are not considered secure in isolation. Two-dimensional maps such as Hénon, Ikeda, and Tinkerbell trade higher computational cost for stronger, higher-dimensional mixing and are consequently favored in image encryption and hybrid designs. Chirikov and Chebyshev maps offer strong ergodic mixing at the highest floating-point cost. This trade-off between map dimensionality, mixing strength, and computational cost recurs throughout the review. Figure 3 illustrates the transition to chaos for the logistic map, the most widely studied map in PRNG design.

Review of Related Literature on Chaos-Based PRNGs: Al-Daraiseh *et al.* (2023) introduced the State-Based Tent-Map, a modified chaotic random-number generator addressing key limitations of the original Tent map, and reported a 97.4% pass rate across 91,200 Diehard tests. De Micco *et al.* (2021) presented ROSEUR-MOD, which uses modified Euler discretization to convert continuous chaotic systems into discrete maps; it passes all NIST and Diehard tests with reduced precision and is efficient on field-programmable gate arrays. Irfan *et al.* (2020) proposed HC-MRLM with Control of Chaos, achieving a uniform binary distribution and passing all 15 NIST SP 800-22 tests. Wang and Cheng (2019) proposed a PRNG based on the logistic chaotic system, transforming logistic outputs into 15-digit numbers and applying modular reduction to form binary sequences. Yu *et al.* (2019) introduced a PRNG combining a four-wing memristive hyperchaotic system with a Bernoulli map, achieving 62.5 Mbps in an FPGA implementation while passing rigorous statistical tests. Ismail *et al.* (2018) presented a hybrid Tent–Logistic PRNG for secure communications that passes NIST SP 800-22 while remaining computationally lightweight.

Machicao and Bruno (2017) proposed a deep-zoom technique to extract the k rightmost digits of chaotic orbits, competing with the Mersenne Twister in statistical performance. Jiang *et al.* (2017) developed a true random number generator based on stochastic threshold switching in an Ag:SiO_2 diffusive memristor, passing all 15 NIST tests without post-processing. Avaroglu (2017) proposed a PRNG combining two Arnold cat maps, passing NIST tests with a hardware-friendly design. Stoyanov and Kordov (2015) proposed a PRNG based on two Tinkerbell maps with a key space of 2^{183}, passing NIST, DIEHARD, and ENT tests; earlier work combined two Chirikov standard maps with a modified shrinking rule and also passed all three suites (Stoyanov and Kordov 2014).

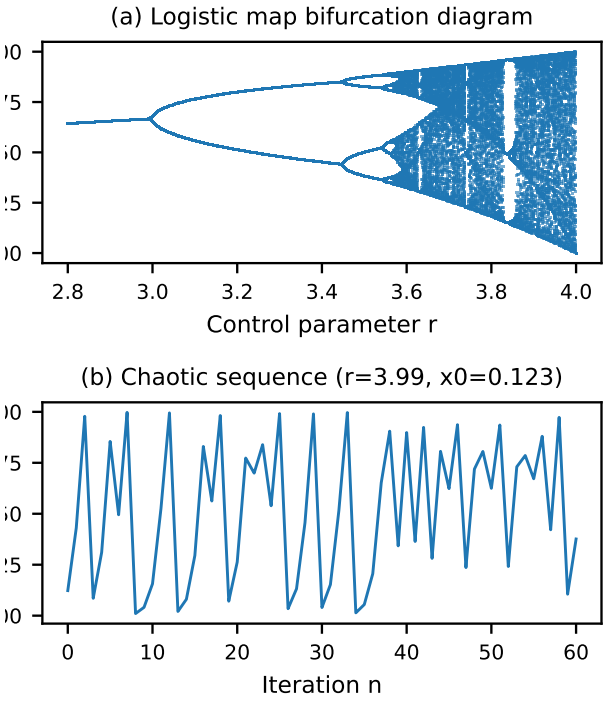

Figure 3 Chaotic behavior of the logistic map in Equation (17). (a) Bifurcation diagram for $r \in [2.8, 4.0]$: period-doubling cascades begin near $r \approx 3.0$, and deterministic chaos emerges near $r \approx 3.57$. (b) A representative chaotic sequence at $r = 3.99$, $x_0 = 0.123$, illustrating aperiodic, noise-like output

Discussion of Tables 4 and 5: Read together, the tables show that essentially every chaos-based design surveyed reports passing NIST SP 800-22 and/or Diehard once appropriate post-processing, such as XOR combination, shrinking rules, or CRT-style reconstruction, is applied. Statistical adequacy is therefore largely achievable across map choices; the differentiator across studies is instead throughput and hardware cost. FPGA-targeted designs report the highest throughput and lowest resource use but the least adversarial evaluation, while software- or simulation-only designs report larger key spaces and more exhaustive test-suite coverage at the cost of speed. This is the same speed-versus-assurance trade-off quantified later in Table 8 and Figure 7.

Synthesis: Taken as a family, chaos-based PRNGs close much of the security gap left open by traditional generators without incurring the full computational cost of cryptographically secure designs. Their principal shared weakness is implementation rather than mathematics: finite-precision arithmetic, poorly chosen control parameters, and inadequate post-processing can silently degrade entropy or shorten the effective period. Nearly every strong result in Tables 4 and 5 therefore depends on an explicit precision- or post-processing-related design choice.

Hybrid and Residue-Number-System-Based PRNGs

Hybrid PRNGs combine linear generators such as LCGs with nonlinear chaotic maps. A general hybrid LCG, chaotic-map generator

Table 3 Comparative properties of selected chaotic maps used in PRNG design. "Chaotic range" denotes parameter values that produce well-characterized chaos

Map	Chaotic range	Advantages	Limitations	Applications
Logistic	$r \in (3.57, 4]$	Simple; fast; high sensitivity	Finite precision collapses chaos; insecure alone	Image encryption; keys; simulation
Tent	$\mu \in (1, 2]$	Uniform distribution; piecewise linear	Periodic if poorly tuned	Hardware RNGs; secure communications
Sine	$\mu \in (0, 1]$	High entropy; smooth nonlinearity	Limited dynamic range; scaling needed	Encryption; random-bit generation
Chebyshev	$k \geq 2$	Strong mixing; high unpredictability	Trigonometric overhead	Chaotic cryptosystems
PWLCM	$p \in (0, 0.5)$	Good uniformity; reduced periodicity	Sensitive to digital quantization	Image encryption; FPGA PRNGs
Ikeda	$\mu \approx 0.9$	High-dimensional chaos; strong nonlinearity	Complex and slower	Hybrid PRNGs; secure streams
Hénon	$a = 1.4$, $b = 0.3$	Bivariate chaos; high entropy	Precision loss; parameter sensitivity	Image encryption; masking
Lorenz	$\sigma = 10$, $\rho = 28$, $\beta = 8/3$	Continuous chaos; strong unpredictability	Numerical solvers; heavy computation	Physical modeling; communications
Tinkerbell	$a = 0.9$, $b = -0.6013$	Strong chaos; complex attractor	Parameter- and quantization-sensitive	Pixel permutation; encryption
Chirikov	$K > 0.9716$	Excellent mixing; ergodic behavior	High floating-point cost	Secure communications

Table 4 Comparative analysis of PRNGs based on chaotic systems (Part A). NIST = NIST SP 800-22; DH = Diehard; ENT = ENT test suite

Author (year)	Map(s)	Key findings and methodology	Strengths	Limitations
Al-Daraiseh et al. (2023)	Modified Tent (SBTM)	Circular state array and irrational seeds; 97.4% DH pass rate (88,830/91,200).	Flexible initialization; near-zero failure rate.	Large state arrays degrade performance.
De Micco et al. (2021)	Continuous chaotic (ROSEUR-MOD)	Modified Euler discretization; all NIST and DH tests passed with reduced precision; FPGA-efficient.	Resource-efficient; reduced-precision capable.	Output quality depends on time-step.
Irfan et al. (2020)	HC-MRLM + CoC	Control of Chaos for uniform binary distribution; all 15 NIST tests passed.	Strong unpredictability; large key space.	CoC region selection not fully justified.
Wang and Cheng (2019)	Logistic	Fifteen-digit transform and modular reduction; large dynamic key space.	Flexible key space; strong statistical properties.	Complex processing pipeline; susceptible to cryptanalysis.
Yu et al. (2019)	Four-wing memristive + Bernoulli	RK4 integration and XOR post-processing; 62.5 Mbps FPGA throughput; all NIST tests passed.	Dual-entropy architecture; high throughput.	Evaluated only on a single FPGA platform; no adversarial analysis.
Ismail et al. (2018)	Tent + Logistic hybrid	XOR combination outperformed individual maps; passed NIST SP 800-22; lightweight implementation.	High sensitivity; lightweight design.	Simulation-based validation only; no hardware implementation.

Table 5 Comparative analysis of PRNGs based on chaotic systems (Part B)

Author (year)	Map(s)	Key findings and methodology	Strengths	Limitations
Machicao and Bruno (2017)	Logistic (deep-zoom)	Extracts the k rightmost digits of chaotic orbits; achieves statistical quality comparable to MT19937.	Excellent statistical properties; passes multiple randomness test suites.	Computationally intensive; conjugacy regions must be excluded.
Jiang et al. (2017)	Diffusive memristor (TRNG)	Uses stochastic threshold switching in Ag:SiO_2 devices; generated 76 million bits and passed all 15 NIST tests.	Ultra-low power consumption; no post-processing required.	Sensitive to pulse parameters; device lifetime limited to approximately 10^7 cycles.
Avaroglu (2017)	Two Arnold cat maps	Employs sampler, comparator, and bit generator; successfully passes NIST tests with hardware-oriented implementation.	Low computational complexity; suitable for hardware implementation.	Sampling process decreases effective bit generation rate; lacks cryptanalysis.
Stoyanov and Kordov (2015)	Two Tinkerbell maps	Uses preprocessed real fractions; provides a key space of 2^{183} and passes NIST, Diehard, and ENT tests.	Large key space; resistant to brute-force attacks.	Lower generation speed than conventional non-chaotic PRNGs.
Stoyanov and Kordov (2014)	Two Chirikov maps + shrinking rule	Implements the Jabri shrinking generator; achieves near-ideal entropy and passes all NIST tests.	Simple architecture with excellent statistical performance.	Requires careful selection of parameters K_1 and K_2.

is

$$X_{n+1} = (aX_n + c) \bmod m, \tag{23}$$

$$x_{n+1} = f(x_n), \tag{24}$$

$$R_n = X_{n+1} \oplus \left(\lfloor x_{n+1} \times 10^k \rfloor \bmod m \right), \tag{25}$$

where $f(x) = rx(1-x)$ for the logistic map, $\oplus$ denotes XOR, k controls chaotic scaling precision, and R_n is the final hybrid output.

Residue-number-system-based PRNGs use modular arithmetic over coprime moduli combined through the Chinese Remainder Theorem (De Micco *et al.* 2021; Irfan *et al.* 2020). A number X_n is represented as

$$X_n \equiv (x_{n,1}, \ldots, x_{n,k}) \pmod{(m_1, \ldots, m_k)}, \tag{26}$$

with each residue evolving independently,

$$x_{n+1,i} = (a_i x_{n,i} + c_i) \bmod m_i, \qquad i = 1, \ldots, k, \tag{27}$$

and the combined output reconstructed through

$$X_{n+1} = \left(\sum_{i=1}^{k} x_{n+1,i} M_i M_i^{-1} \right) \bmod M, \tag{28}$$

where $M = \prod m_i$, $M_i = M/m_i$, and M_i^{-1} is the modular inverse of M_i modulo m_i.

Synthesis: Hybrid and residue-number-system designs occupy the middle ground between traditional, cryptographic, and chaos-based generators. The LCG stage in Equation (23) preserves the throughput of a linear generator, the chaotic stage in Equations (24)–(25) breaks the linear recoverability of a standalone LCG, and the formulation in Equations (26)–(28) removes carry propagation, making this family attractive for FPGA and parallel-hardware acceleration. This is reflected quantitatively in Table 8 and Figure 7, where hybrid and residue-number-system designs sit closer to the high-speed, high-security region than either purely traditional or purely chaotic designs alone.

Machine-Learning-Based PRNGs

Machine learning provides two important contributions to PRNG research. First, models serve as evaluators that learn hidden patterns and biases beyond what classical statistical tests reveal. Second, machine-learning architectures can become components of new PRNG designs (Boanca 2024; Kumar *et al.* 2023). Supervised learning uses labeled data to learn a mapping $f(x) \to y$; unsupervised learning finds structure in unlabeled data; semi-supervised learning combines a small labeled set with a larger unlabeled one; reinforcement learning trains an agent by trial and error; and deep learning uses multilayer networks to learn hierarchical representations.

Literature review: Boanca (2025) investigated artificial neural networks for enhancing LFSRs and the Geffe generator, proposing a Geffe-learning approach with 100% accuracy, though limited to known LFSR degrees. Egan (2024) examined high-speed secure random-number co-processors for privacy-preserving machine learning, identifying inadequacies of standard PRNGs for cryptographic security. Almardeny *et al.* (2023) proposed an upside-down reinforcement-learning system grounded in generalized information entropy that produces high-quality random sequences without training data. Crespo *et al.* (2024) assessed random-number quality using convolutional and long short-term memory networks, achieving approximately 79% classification accuracy, with wavelet features outperforming Fourier-based inputs. Truong *et al.* (2019) employed a recurrent convolutional network to analyze classical-noise impact on a quantum random-number generator, demonstrating robustness of post-processing against machine-learning attacks. Nagy and Suciu (2021) investigated neural architectures for random-number verification across quantum and linear congruential generators. Antunes and Hill (2024) compared Mersenne Twister, Philox, PCG, and Mrg32k3a across Python, NumPy, TensorFlow, and PyTorch using the Big Crush test, noting that PyTorch's default generator fails certain statistical tests. Dahiya *et al.* (2024) showed that PRNG tampering attacks, including bit-flipping, can degrade certified model robustness while evading NIST tests. Aljohani and Al-Barhamtoshy (2025) demonstrated that a 6–30–20–2 neural network detects implicit biases in cryptographically secure PRNGs more effectively than NIST SP 800-22. Fischer (2018) showed that long short-term memory testing detects hidden flaws missed by Diehard tests, while Wen and Yu (2019) proposed a physically unclonable-function-integrated PRNG achieving only 52.6% prediction accuracy, demonstrating resistance while maintaining high entropy.

On the residue-number-system and hardware side, Mehrabi *et al.* (2021) proposed an RNS-based GLV-ECC core resistant to machine- and deep-learning side-channel attacks. Mukhtar *et al.* (2022) conducted an extensive machine-learning side-channel analysis of RNS-ECC and reported 95–99% key-recovery accuracy. Valueva *et al.* (2020) implemented convolutional layers through residue arithmetic on FPGAs, with reported hardware-cost savings of 7.86–37.78 times. Peng *et al.* (2020) proposed DNNARA, a hybrid opto-electronic accelerator using residue arithmetic and wavelength-division multiplexing, achieving a 19.3-fold speedup over graphics processors.

Figure 5 summarizes the dual, and at times adversarial, roles that machine learning plays in this literature: evaluating existing generators for hidden bias, attacking generators to recover state, and directly generating new candidate PRNGs.

Synthesis: Two consistent findings emerge. First, machine-learning evaluators repeatedly find structure that classical NIST and Diehard tests miss (Fischer 2018; Aljohani and Al-Barhamtoshy 2025), suggesting that machine-learning-based testing should complement rather than replace standard suites. Second, machine-learning generation and residue-number-system hardware accelerators achieve strong statistical quality but at substantially higher computational and training cost than other families, which is the trade-off quantified next.

COMPARATIVE ANALYSIS

Methodology for Quantitative Comparison

The normalized comparisons in Tables 8 and 9, and Figures 6–7, are literature-derived syntheses rather than results from a single controlled experiment. For each of the fourteen criteria in Tables 6 and 7, the reported numerical values or ranges from the primary studies reviewed in Sections – were collected by family. Where a study reported a range, the midpoint was used as its representative value; where only a qualitative claim was available, such as "passes all NIST tests," it was scored at the upper end of the corresponding criterion for that family. Values were then min–max normalized to a common 0–10 scale within each criterion across all three families, and the per-family scores in Figure 6 are the resulting family-level averages. This approach follows the same aggregative logic used in prior PRNG comparison studies (Antunes and Hill 2024; Antunes 2025). It is intended to expose relative ordering and trade-offs across families rather than assert precision at the level of individual decimal values.

Traditional, Chaos-Based, and Machine-Learning-Based PRNGs

Traditional PRNGs such as LCGs are fastest, at approximately 10^6 numbers per second, but have limited security or statistical quality. Chaos-based PRNGs use nonlinear dynamics to provide moderate speed with good statistical quality, with reported NIST compliance around 97–98%. Machine-learning-based generators can approach 99% NIST compliance but are the slowest, at approximately 10^3 numbers per second. This heterogeneity drives research toward hybrid solutions combining advantages of all types. Tables 6 and 7 provide a detailed side-by-side comparison across fourteen criteria.

Discussion of Tables 6 and 7: The tables show a near-inverse relationship between randomness quality/security and computational efficiency/energy cost as one moves from traditional to chaos-based to machine-learning-based PRNGs. Traditional generators dominate every efficiency-oriented row but trail on quality-oriented rows such as entropy and attack resistance. Chaos-based generators sit between the two extremes on almost every criterion, making them attractive for resource-constrained but security-relevant deployments such as Internet-of-Things nodes. Machine-

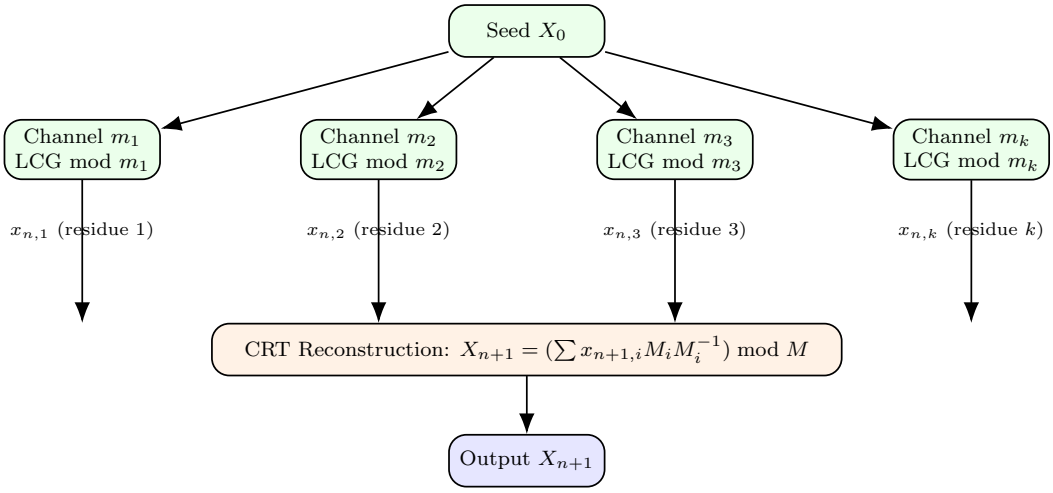

Figure 4 Residue-number-system-based PRNG architecture. The seed is decomposed into k independent residue channels; each evolves in parallel through a modular LCG recurrence; outputs are reconstructed into a full-period pseudo-random integer through the Chinese Remainder Theorem. This parallelism avoids carry propagation and enables hardware acceleration

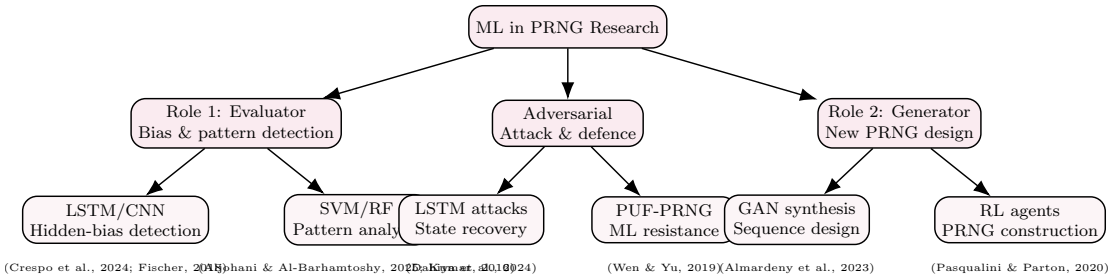

Figure 5 Dual and adversarial roles of machine learning in PRNG research. Machine learning contributes as an evaluator for bias and pattern detection, as an adversarial tool for state-recovery attacks and resistant design, and as a generator constructing new PRNG architectures through generative adversarial networks and reinforcement learning

Table 6 Comparison of Traditional, Chaos-Based, and Machine-Learning-Based PRNGs (Part A). ↑ = higher is better; ↓ = lower is better.

Criterion	Traditional PRNGs	Chaos-Based PRNGs	ML-Based PRNGs	Future ML Directions
Randomness quality ↑	Moderate; depends on seed quality	High; nonlinear sensitivity	Potentially very high with high-quality training data	Hybrid chaos–ML models; adaptive GAN-based entropy
Complexity ↓	Low; simple arithmetic operations	Moderate; nonlinear chaotic maps	High; computationally intensive training	Lightweight and edge-oriented ML architectures
Computational efficiency ↑	Very high; highly optimized implementations	Moderate; slower than conventional PRNGs	Lower due to training overhead	Optimized inference and edge deployment
Scalability ↑	Excellent in software and hardware	Moderate; synchronization required for coupled maps	High on GPU/cloud platforms; limited on embedded systems	Federated and distributed ML-based PRNGs
Security ↑	Limited unless cryptographically strengthened	High; unpredictable chaotic dynamics	Very high; neural parameters difficult to reconstruct	Adversarial learning and quantum-enhanced security
Adaptability ↑	Static after initialization	Moderate; parameter tuning possible	Highly adaptive through retraining	Self-learning PRNGs using reinforcement learning
Resource constraints ↓	Minimal CPU and memory usage	Moderate; floating-point arithmetic often required	High computational and memory requirements	Quantized neural models for IoT devices
Execution time ↓	Fastest	Fast to moderate	Slow due to model inference	Model pruning and hardware acceleration

learning-based generators top quality-oriented rows but perform worst on efficiency and hardware suitability; at present, they are therefore better suited to evaluation and offline generation than to real-time, high-throughput use.

Normalized Criterion-by-Criterion Comparison

Table 8 restates the comparison from a complementary angle, reporting representative numerical ranges for typical period, throughput, entropy, autocorrelation, NIST pass rate, security

Table 7 Comparison of traditional, chaos-based, and machine-learning-based PRNGs (Part B).

Criterion	Traditional PRNGs	Chaos-based PRNGs	ML-based PRNGs	Future ML Directions
Hybrid potential ↑	Common with secure generators and hashing	Very high; often hybridized	Emerging ML–chaos hybrids	ML–chaos–quantum frameworks
Entropy level ↑	Moderate (0.7–0.9 bits/bit)	High (≈1 bit/bit)	Very high (≈0.99 bits/bit)	Continuous-feedback entropy optimization
Period length ↑	Finite and known, up to $2^{19937}-1$	Long effective chaotic periods	Variable; architecture-dependent	Dynamic period control with recurrent ML
Autocorrelation ↓	Moderate linear dependency	Low chaotic correlation	Very low after training	Correlation-penalty loss functions
Uniformity ↑	Good with large moduli	Excellent with proper tuning	Near-perfect after calibration	Entropy-aware calibration networks
Hardware suitability ↑	Excellent for FPGA/ASIC	Moderate hardware cost	Limited by large model sizes	Efficient ML-PRNG accelerators
Energy consumption ↓	Low	Moderate	High during training and inference	Low-power inference; neuromorphic PRNGs
NIST/Diehard compliance ↑	Passes most tests; weak LCGs fail	Passes when parameters are optimal	Full compliance after training	Standardized ML-PRNG protocols
Attack resistance ↑	Low if seed is exposed	High; parameters difficult to predict	Very high; model parameters difficult to reverse	Adversarial robustness metrics

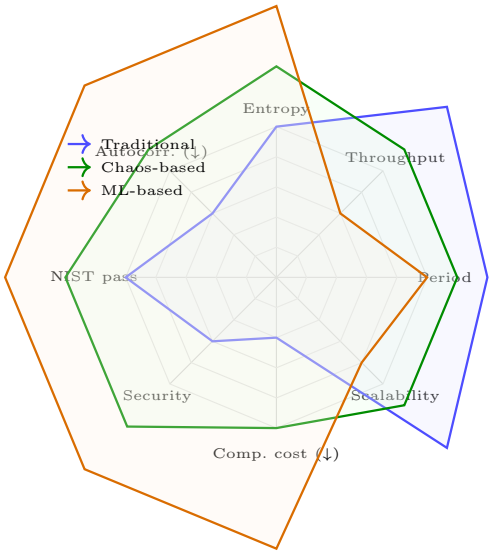

Figure 6 Normalized comparison of traditional, chaos-based, and machine-learning-based PRNGs across eight performance criteria on a 0–10 scale, derived using the aggregation method described in Section . Lower values are better for autocorrelation and computational cost

strength, computation cost, memory requirement, scalability, hardware suitability, determinism, and cryptographic applicability.

Discussion of Table 8: The absolute ranges reinforce the ordering seen in Tables 6–7 and Figure 6. Machine-learning-based PRNGs report the highest entropy and lowest autocorrelation but the lowest throughput and highest memory footprint. Traditional generators report the inverse. Chaos-based PRNGs again occupy the intermediate position on nearly every row, consistent with their use as a compromise choice for embedded and Internet-of-Things cryptography.

Security–Speed Trade-Off

The security–speed trade-off is a central design constraint in PRNG selection, and Table 9 summarizes it for four representative designs. LFSRs and Xorshift generators are extremely fast and hardware-friendly but linear, making them unsuitable for cryptographic use. MT19937 offers excellent statistical quality and a very long period but is not designed to resist state-recovery attacks. Hybrid PRNGs sacrifice some throughput for stronger nonlinearity, better entropy, and improved resistance to prediction. Figure 7 maps thirteen generators onto this two-dimensional landscape and confirms that cryptographically secure and hybrid/residue-number-system designs are the only families approaching the high-speed, high-security region simultaneously.

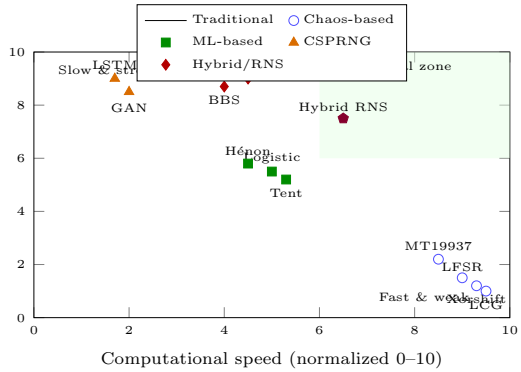

Figure 7 Security–speed landscape for 13 representative PRNGs, using the normalization methodology. Traditional generators cluster in the fast-but-weak region; chaos-based generators occupy the moderate zone; machine-learning-based PRNGs are strong but slow; cryptographically secure and hybrid/residue-number-system designs approach the desirable high-speed, high-security zone

DISCUSSION

The comparative findings indicate that PRNG performance is application-dependent. Traditional generators provide the best

Table 8 Normalized comparison of traditional, machine-learning-based, and chaos-based PRNGs

Metric	Traditional PRNGs (LCG, LFSR, MT19937)	ML-based PRNGs (DNN, GAN, RNN)	Chaos-based PRNGs (Logistic, Tent, Hybrid)
Typical period	10^6–10^{19}	10^{30}–10^{60}	10^{12}–10^{40}
Throughput (Mbit/s)	500–1500	50–200	200–1000
Entropy (bits/sample)	0.85–0.92	0.97–0.999	0.94–0.995
Autocorrelation	0.05–0.15	0.01–0.04	0.02–0.08
NIST pass rate (%)	85–90	97–99	93–98
Security strength (bits)	≤ 64 (weak)	≥ 128	96–128
Computation cost	Low	High	Medium
Memory requirement	16–256 bytes	5–50 MB	< 1 KB
Scalability	Excellent	Limited (GPU/TPU)	High
Hardware suitability	CPU, FPGA	GPU, TPU	CPU, FPGA, IoT
Determinism	Fully deterministic	Semi-deterministic	Deterministic
Cryptographic applicability	Weak–moderate	High	Moderate–high

Table 9 Security and speed trade-off for selected PRNGs

Type	Speed	Security	Trade-off Summary
LFSR	Very high	Very low	Fast but fully predictable through the Berlekamp–Massey algorithm; therefore, it is unsuitable for cryptographic applications.
MT19937	High	Low	Provides excellent statistical properties, but its internal state can be reconstructed from 624 consecutive outputs, making it unsuitable for cryptographic security.
Xorshift	Very high	Low	Extremely fast and computationally efficient due to linear operations; however, it lacks cryptographic security and is vulnerable to state recovery attacks.
Hybrid PRNG	Moderate–high	Moderate–high	Slightly slower than conventional linear generators, but offers significantly improved entropy, unpredictability, and robustness against statistical and differential attacks.

throughput and lowest implementation cost, making them suitable for simulation, graphics, sampling, and non-security-critical modeling; their weakness is predictability, since linear recurrence structures and recoverable internal states make them unsuitable for cryptographic key generation and secure communication. Chaos-based generators improve unpredictability through nonlinear dynamics, sensitivity to initial conditions, and low correlation, and are especially attractive for lightweight cryptography, image encryption, Internet-of-Things devices, and embedded security. Their main limitation is implementation fragility, since finite precision, poor parameter choices, and inadequate post-processing can reduce entropy or create short cycles.

Machine-learning-based approaches broaden the PRNG landscape by supporting both generation and evaluation: neural and reinforcement-learning methods detect hidden bias, model complex dependencies, and generate high-quality sequences, but they introduce significant computational cost, training dependence, reproducibility challenges, and hardware requirements. They are therefore best viewed as complementary tools rather than drop-in replacements. Hybrid and residue-number-system designs most directly address this three-way trade-off, and Table 8 and Figure 7 show them consistently closer to the jointly fast-and-secure region than either purely traditional or purely chaotic designs in isolation.

Future research on pseudo-random number generators should increasingly focus on integrative architectures that combine the computational efficiency of traditional generators, the nonlinear dynamics of chaotic systems, the parallel processing capabilities of residue-number systems, and the adaptive learning capacity of modern machine learning techniques. Such hybrid designs have the potential to simultaneously improve randomness quality, security, and computational performance. In particular, hybrid PRNG architectures integrating fast linear generators with chaotic or cryptographic post-processing can significantly enhance entropy and resistance against prediction attacks while maintaining high throughput.

Another promising direction involves the hardware acceleration of chaos-based and machine-learning-assisted PRNGs through FPGA, ASIC, GPU, TPU, and low-power embedded implementations, enabling their deployment in real-time and resource-constrained applications (Fang *et al*. 2014). Furthermore, precision-aware chaotic systems capable of preserving long periods, high entropy, and robust chaotic behavior under fixed-point arithmetic remain an important research challenge for practical hardware realization.

Future generators should also incorporate adaptive reseeding strategies using trusted entropy sources to maintain unpredictability even if part of the internal state becomes compromised. Finally, comprehensive evaluation methodologies should evolve beyond existing statistical frameworks by extending standards such as NIST SP 800-22, DIEHARD, and TestU01 to better assess the security and randomness characteristics of nonlinear, hybrid, and machine-learning-based PRNGs (Antunes 2025).

CONCLUSION

This review shows that PRNG design requires a careful balance among speed, randomness quality, predictability, memory cost, scalability, and cryptographic strength. Traditional generators remain valuable for high-speed, non-secure applications. Chaos-

based generators offer stronger nonlinear behavior and higher entropy but require careful parameter tuning and precision-aware implementation. Hybrid and residue-number-system methods provide a promising bridge between these families. Machine-learning-based approaches add value by detecting hidden patterns and supporting adaptive generation, although computational cost and reproducibility remain open challenges. No single PRNG family is optimal for every environment: simulation benefits from fast traditional generators; lightweight cryptography and Internet-of-Things applications from chaos-based or hybrid generators; and high-security systems from machine-learning-enhanced evaluation and adaptive reseeding. The most promising direction is a layered architecture combining speed, nonlinearity, entropy management, hardware efficiency, and rigorous testing.

Funding

This research received no specific grant from any funding agency, commercial entity, or not-for-profit organization. The study was conducted independently and fully supported by the authors.

Conflict of Interest

The authors declare that there is no conflict of interest regarding the publication of this research.

Availability of data and material

The data supporting the findings of this study are available within the manuscript. No external datasets were generated or analyzed. Additional information is available from the corresponding author upon reasonable request.

Ethical standard

The authors have no relevant financial or non-financial interests to disclose.

LITERATURE CITED

Al-Daraiseh, A., Y. Sanjalawe, S. Al-E'mari, S. Fraihat, M. Bany Taha, *et al.*, 2023 Cryptographic grade chaotic RNG based on tent-map. Journal of Sensor and Actuator Networks **12**: 73.

Aljohani, M. A. and H. Al-Barhamtoshy, 2025 A method for assessing the performance of cryptographic PRNGs utilizing ANNs. In *Proceedings of ICAISC 2025*, pp. 1–5.

Almaraz Luengo, E. and J. Román Villaizán, 2023 Cryptographically secured pseudo-random number generators. Mathematics **11**: 4812.

Almardeny, Y., A. Benavoli, N. Boujnah, and E. Naredo, 2023 A reinforcement learning system for generating instantaneous quality random sequences. IEEE Transactions on Artificial Intelligence **4**: 402–415.

Aniszewska, D., 2018 New discrete chaotic multiplicative maps based on the logistic map. International Journal of Bifurcation and Chaos **28**: 1850118.

Antunes, B. and D. R. C. Hill, 2024 Random numbers for machine learning. Journal of Data Science and Intelligent Systems .

Antunes, B. A., 2025 Statistical quality and reproducibility of PRNGs in ML. arXiv preprint arXiv:2507.03007.

Avaroglu, E., 2017 Pseudorandom number generator based on arnold cat map. Turkish Journal of Electrical Engineering and Computer Sciences **25**: 633–643.

Boanca, S., 2024 Exploring patterns and assessing the security of PRNGs with ML. In *Proceedings of ICAART 2024*, pp. 186–193.

Boanca, S., 2025 PRNGs, perfect learning and model visualization with neural networks. In *Proceedings of IoTBDS 2025*, pp. 117–128.

Crespo, J. L., J. González-Villa, J. Gutiérrez, and A. Valle, 2024 Assessing the quality of RNGs through neural networks. Machine Learning: Science and Technology **5**: 025072.

Dahiya, P., I. Shumailov, and R. Anderson, 2024 Machine learning needs better randomness standards. In *Proceedings of the 33rd USENIX Security Symposium*.

De Micco, L., M. Antonelli, and O. A. Rosso, 2021 From continuous-time chaotic systems to PRNGs. Entropy **23**: 671.

Egan, S., 2024 High-speed secure RNG co-processors for privacy-preserving ML. Unpublished manuscript.

Fang, X., Q. Wang, C. Guyeux, and J. M. Bahi, 2014 FPGA acceleration of a PRNG based on chaotic iterations. Journal of Information Security and Applications **19**: 78–87.

Fischer, S., 2018 Testing CSPRNGs with ANNs. In *Proceedings of SecITC 2018*.

Guisande, N., M. P. Di Nunzio, N. Martinez, O. A. Rosso, and F. Montani, 2023 Chaotic dynamics of the hénon map. Chaos **33**: 043111.

Irfan, M., A. Ali, M. A. Khan, M. Ehatisham-ul Haq, S. N. M. Shah, *et al.*, 2020 PRNG design using HC-MRLM. IEEE Access .

Ismail, S. M., M. N. Mohammed, M. A. Ameedeen, and H. A. S. Ahmed, 2018 A new trend of PRNG using multiple chaotic systems. Advanced Science Letters **24**: 7401–7406.

Jiang, H., D. Belkin, S. E. Savel'ev, S. Lin, Z. Wang, *et al.*, 2017 A novel true random number generator based on a stochastic diffusive memristor. Nature Communications **8**: 882.

Kumar, K. J., K. Jairam, and C. Ambedkar, 2023 An in-depth study of ML in artificial intelligence. Educational Administration: Theory and Practice .

Machicao, J. and O. M. Bruno, 2017 Improving pseudo-randomness properties of chaotic maps using deep-zoom. Chaos **27**: 053116.

Maheshwari, R., S. Gupta, V. Sharma, and V. Chauhan, 2014 In *Proceedings of IEEE IACC 2014*.

Martini, G. and F. G. Bruno, 2017 True random numbers generation from stationary stochastic processes. In *Proceedings of ICNF 2017*, pp. 1–4.

Mehrabi, M. A., N. Mukhtar, and A. Jolfaei, 2021 Power side-channel analysis of RNS GLV ECC. ACM Transactions on Internet Technology **21**: 1–20.

Mukhtar, N. *et al.*, 2022 ML-assisted side-channel attacks on RNS ECC using hybrid feature engineering.

Nagy, I. and A. Suciu, 2021 Randomness testing with neural networks. In *Proceedings of IEEE ICCP 2021*, pp. 431–436.

Noibate, S., 2023 Random number generators, challenges and limitations. SSRN Electronic Journal .

Oliveira, D. F. M., 2024 Mapping chaos: Bifurcation patterns and shrimp structures in the ikeda map. Chaos **34**: 123137.

Peng, J., Y. Alkabani, S. Sun, V. J. Sorger, and T. El-Ghazawi, 2020 DNNARA: A DNN accelerator using residue arithmetic and photonics. In *Proceedings of ICPP 2020*.

Saeed, H., W. I. El Sobky, T. O. Diab, and M. A. Elsisi, 2024 Chaotic maps in cryptography. In *Proceedings of ITC-Egypt 2024*, pp. 635–645.

Sathya, K., J. Premalatha, V. Rajasekar, P. Sathiyananthan, S. TharunPrasath, *et al.*, 2021 A cutting-edge approach to generate random bit sequences. AIP Advances **11**: 140006.

Sayed, W. S., A. G. Radwan, and H. A. H. Fahmy, 2016 Double-sided bifurcations in tent maps. In *Proceedings of ACTEA 2016*, pp. 207–210.

Silva, R. D. and S. D. Prado, 2023 Exploring transition from stability

to chaos through random matrices. Dynamics **3**: 777–792.
Stoyanov, B. and K. Kordov, 2014 A novel PRNG based on chirikov standard map. Mathematical Problems in Engineering **2014**: 986174.
Stoyanov, B. and K. Kordov, 2015 Novel secure PRNG based on two tinkerbell maps. Advanced Studies in Theoretical Physics **9**: 411–421.
Truong, N. D., J. Y. Haw, S. M. Assad, P. K. Lam, and O. Kavehei, 2019 Machine learning cryptanalysis of a quantum random number generator. IEEE Transactions on Information Forensics and Security **14**: 403–414.
Valueva, M. V., N. N. Nagornov, P. A. Lyakhov, G. V. Valuev, and N. I. Chervyakov, 2020 Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. Mathematics and Computers in Simulation **177**: 232–243.
Wang, L. and H. Cheng, 2019 Pseudo-random number generator based on logistic chaotic system. Entropy **21**: 960.
Wen, Y. and W. Yu, 2019 Machine learning-resistant PRNG. Electronics Letters **55**: 515–517.
Yu, F., Q. Wan, J. Jin, L. Li, B. He, *et al.*, 2019 Design and FPGA implementation of a PRNG based on a four-wing memristive hyperchaotic system. IEEE Access **7**: 181884–181898.

How to cite this article: Karim, M. A., Agbedemnab, P. A., and Alhassan, S. From Deterministic Chaos to Machine Learning: A Comparative Review of Pseudo-Random Number Generators for Modern Computing and Cryptography. *Chaos and Fractals*, 3(2), 125-136, 2026.