Journal of Discrete Mathematical Sciences & Cryptography
ISSN 0972-0529 (Print), ISSN 2169-0065 (Online)
Vol. 29 (2026), No. 10-B, pp. 4237–4247
DOI : 10.47974/JDMSC-3047

Statistical attacks on pseudorandom number generators

K. Madhura [†]
Department of Information Technology
Manipal Institute of Technology Bengaluru
Manipal Academy of Higher Education (Deemed to be University)
Bengaluru 560064
Karnataka
India

Banothu Vijay *
Centre for Multidisciplinary Research
Anurag University
Hyderabad 500088
Telangana
India

Anisha Chaudhary [§]
Quantum University Research Center
Quantum University
Roorkee 247667
Uttarakhand
India

Vimal Bibhu [‡]
Department of Computer Science & Engineering
Noida International University
Greater Noida 203201
Uttar Pradesh
India

[†] *E-mail:* madhura.k@manipal.edu
* *E-mail:* BanothuVijay@rediffmail.com (Corresponding Author)
[§] *E-mail:* anisha.chaudhary@quantumeducation.in
[‡] *E-mail:* vimal.bhibu@niu.edu.in

© Taru Publications

Sardor Buriyev [@]
Department of Information Technology and Exact Sciences
Termez University of Economics and Service
Termez 190111
Uzbekistan

Mutabar Latipova [#]
Tashkent Institute of Irrigation and Agricultural Mechanization Engineers
National Research University
Kori Niyoziy 39
Tashkent 100000
Uzbekistan

and

Department of Economics
University of Tashkent for Applied Sciences
Str. Gavhar 1
Tashkent 100149
Uzbekistan

Abstract

The statistical attacks on pseudorandom number generators (PRNGs) expose vulnerability of the deterministic nature of the structure of the generators. This paper discusses mathematical concepts which determine how PRNG works and the statistical vulnerability of PRNG to frequency and block frequency tests. Moreover, pattern recognition methods that are based on machine learning are used to extract latent correlations and predictability in the generated sequences. Uniformity, independence and entropy are the major statistical properties which are systematically tested to ascertain the extent of randomness. As empirical evidence indicates, state-of-the-art PRNGs have predictable biases, and a rigorous statistical analysis, and that machine learning-based models can enhance the effectiveness of attacks, uncovering the latent trends. The comparative analysis indicates that performance variations in terms of testing techniques and learning models exist. The findings suggest that there ought to be effective design policies and improved randomness validation schemes to minimize the statistical attacks and improve the security of cryptography.

Subject Classification: *65C10, 94A60, 62G10, 68T05.*

Keywords: *Pseudorandom number generators, Statistical attacks, Frequency test, Machine learning, Randomness analysis, Cryptographic security.*

[@] *E-mail:* **sardor_buriyev@tues.uz**
[#] *E-mail:* **latipovm1991@gmail.com**

I. Introduction

PRNGs play a crucial role in the modern cryptography and simulation along with the secure communications as they produce sequences that are almost close to random [1]. Even the popular variants of PRNGs are deterministic, although numerous variants have been popular, with initial seeds and algorithm designs capable of introducing subtle patterns or correlations [2, 3]. The latent patterns make PRNGs vulnerable to statistical attacks, where attackers exploit the lack of ideal randomness to predict future outcomes, or to make inferences about internal states. The use of probabilistic tests and mathematical models to analyse output sequences to detect non-random behaviour is of interest to statistical attacks [4, 5, 6]. Classical tests such as frequency (monobit) and block frequency tests are tests of uniformity and distribution and more sophisticated methods employ entropy tests and correlation tests [7, 8]. Over the past few years machine learning algorithms have emerged as a powerful tool of uncovering non-obvious patterns that could be overlooked by less powerful statistical tests, and making such attacks far more efficient [9, 10].

II. Mathematical foundations and deterministic nature of PRNGs

PRNGs are deterministic algorithms, which use starting variables to create sequences, which are based on recurrence relations, modular arithmetic, state transitions and this introduces predictable structures which can be manipulated when they are subjected to statistical analysis [11, 12].

Recurrence Relation Definition

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

The basic idea behind PRNGs is the recurrence relation, as expressed in equation (1), where each successive value is determined based on the state of the last one, with modular arithmetic, and guarantees deterministic but apparent random number sequences.

Closed-Form Expansion of Sequence

$$X_n = \left(a^n X_0 + \Sigma_{\{i=0\}}^{\{n-1\}} a^i c \right) mod\ m \tag{2}$$

The relation can be extended to the closed form expression in equation (2) and the results show how the first seed and parameters affect the evolution of the whole sequence with the number of iterations.

Estimation of variability and dispersion.

$$S_n = \sqrt{\left(\left(\frac{1}{n}\right) \Sigma_{\{i=1\}}^{\{n\}} \left(\left(\frac{X_i}{m} - \mu\right)^2 \right) \right)} \tag{3}$$

The variability is measured by equation (3) in terms of dispersion of normalized values about the mean to represent the extent to which the generated numbers are not distributed uniformly as they should be.

Mean and Standard Deviation Calculation

$$\mu = \left(\frac{1}{n}\right) \Sigma_{\{i=1\}}^{\{n\}} \left(\frac{X_i}{m}\right) \tag{4}$$

The central tendency of normalized outputs is the mean calculation in equation (4) and it is used as a reference point to determine the degree of randomness.

$$\sigma = \sqrt{\left(\left(\frac{1}{n}\right) \Sigma_{\{i=1\}}^{\{n\}} \left(\left(\frac{X_i}{m} - \mu\right)^2 \right) \right)} \tag{5}$$

This is taken a step further by quantification of standard deviation in equation (5) which makes sense of uniformity and dispersion in the sequence.

Deterministic Deviation Measurement

$$D = \sqrt{\left(\Sigma_{\{i=1\}}^{\{n\}} \left(\frac{X_i}{m} - \frac{1}{2}\right)^2 \right)} + \sqrt{\left(\Sigma_{\{i=1\}}^{\{n\}} \left(X_{\{i+1\}} - X_i\right)^2 \right)} \tag{6}$$

Lastly, equation (6) assesses deterministic deviation, which is the combination of global distribution bias and differences in successive values that embodies statistical imbalance, as well as temporal correlation. All these equations combine to create a full mathematical model of PRNG behavior analysis and the determination of weaknesses that could be used to conduct statistical attacks.

III. Frequency (monobit) test and block frequency test

Frequency test checks the uniformity of the number of zeros and ones, and block frequency test splits sequences into blocks, and identifies local deviations, biases, and inconsistencies in bit patterns.

Bit Transformation and Summation

$$S = \Sigma_{\{i=1\}}^{\{n\}} (2X_i - 1) \tag{7}$$

This is because the values of binary sequences can be transformed to values of {−1, +1} by bit transformation of equation (7) and this facilitates the analysis of balanced summation. Adding all the bits transformed, it determines the probability of the occurrence of zeros and ones to be equal, which is a critical criterion of randomness.

Normalized Test Statistic Computation

$$S_{obs} = \frac{|S|}{\sqrt{(n)}} \tag{8}$$

In equation (8), the cumulative sum has been normalized by dividing its absolute value by the square root of the length of the sequence. This yields the test statistic S ob sallowing comparison in sequences of varying sizes whilst being statistically consistent. The Figure 1 shows the Frequency (Monobit) and Block Frequency Testing Framework for PRNG Randomness Evaluation.

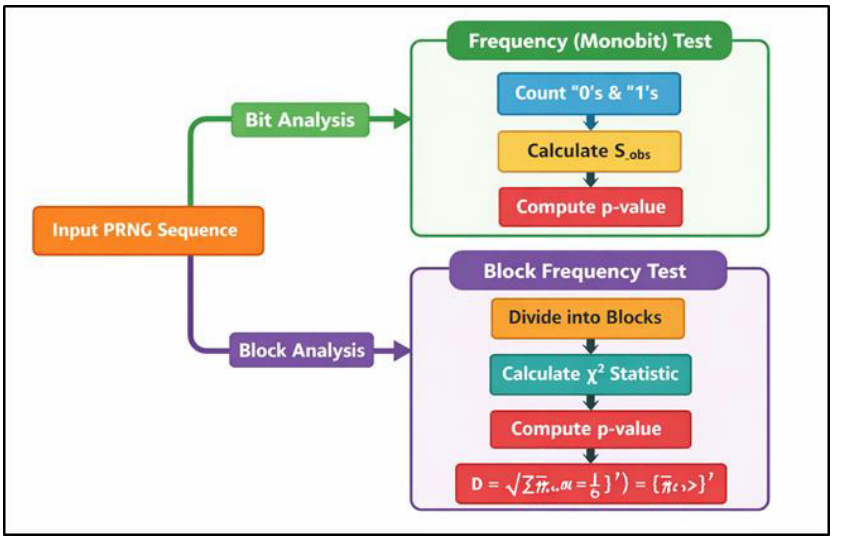


Figure 1

Frequency (Monobit) and Block Frequency Testing Framework for PRNG Randomness Evaluation

Error Function-Based Probability Evaluation

$$p - value \;=\; erfc\left(\frac{S_{obs}}{sqrt(2)}\right) \tag{9}$$

The complementary error function is employed in the probability assessment in equation (9) to calculate the p-value of The p-value of S obs This is used to determine the likelihood of a deviation of the observed sequence to be due to chance, and is used to decide whether to reject or not reject the sequence using the monobit test.

Block Frequency Chi-Square Statistic

$$\chi^2 \;=\; 4M\; \Sigma_{\{i=1\}}^{\{N\}}\left(\pi_i - \frac{1}{2}\right)^2 \tag{10}$$

The block frequency test is presented in equation (10), which has divided the sequence into N blocks of size M. The proportion of ones πi of each block is compared to the expected value 1/2 and local irregularities are detected by summing the deviations to a chi-square statistic.

$$p - value \;=\; \left(\frac{1}{\Gamma\left(\frac{N}{2}\right)}\right) \int_{\left\{\frac{\chi^2}{2}\right\}}^{\{\infty\}} t^{\left\{\left(\frac{N}{2}\right)-1\right\}} e^{\{-t\}dt} \tag{11}$$

Lastly, the incomplete gamma function is used in equation (11) to obtain the p-value which gives a more accurate measure of probability of the chi-square statistic. These equations collectively perform a rigorous evaluation of global and local randomness in outputs of PRNGs.

IV. Machine learning-based pattern recognition

Machine learning models are applied to PRNG outputs to identify latent dependencies, figure out patterns in sequences and anticipate future values, improving statistical attacks over more traditional approaches based on analytical testing.

Hidden State Representation (RNN/LSTM)

$$h_t = \sigma\left(W_h x_t + U_h h_{\{t-1\}} + b_h\right) \tag{12}$$

The hidden state representation of equation (12) is a representation of the sequential dependencies in PRNG output that are represented by recurrent networks like RNN or LSTM. It uses nonlinear activation to combine current input, past hidden state and bias to allow the learning of temporal features.

Output Prediction Layer

$$y_t = W_y h_t + b_y \tag{13}$$

The output prediction layer is determined in equation (13) where the hidden representation is linearly transformed to provide the predictions. This phase gives the probabilistic output values of learned temporal features and enables the model to predict the probability of future sequence values.

Loss Function Formulation (Cross-Entropy)

$$L = -\Sigma_{\{t=1\}_t^{\{T\}y}} \log(\hat{y}_t) \tag{14}$$

The loss term of equation (14) makes use of cross-entropy to measure the discrepancy between the predicted values and the actual ones. It punishes outliers in predictions more strongly and the model is thus trained to learn significant patterns which exist in the PRNG sequence.

Gradient Magnitude Estimation

$$\nabla\theta L = \sqrt{\left(\Sigma_{\{i=1\}}^{\{k\}} \left(\frac{\partial L}{\partial \theta_i}\right)^2\right)} \tag{15}$$

The magnitude of the gradient can be estimated using equation (15) and shows the rate of change of the loss, on average, with respect to the model parameters. This combined measure guarantees consistent optimization since it takes into account all the parameters during training.

Parameter Update with Regularization Term

$$\theta_{\{t+1\}} = \theta_t - \eta\, \nabla\theta L + \left(\left(\frac{1}{k}\right) \Sigma_{\{i=1\}}^{\{k\}\theta_i^2}\right)^{\frac{1}{2}} \tag{16}$$

Lastly, equation (16) also estimates the parameters of the model by applying gradient descent with a regularization term.

V. Statistical Properties of Random Sequences

Random sequences are uniformly distributed, independent, high entropy, and have weak correlations, which guarantees unpredictability;

any failure to satisfy them demonstrates weaknesses, allowing statistical attacks on them and exploiting predictability.

$$\mu = E[X] = \Sigma\, x_i P(x_i) \tag{17}$$

The expectation of (17) is the average of the random sequence, which is the weighted average of all possible values and their probability. It determines the tendency of a distribution and shows whether the distribution is equal.

$$\sigma^2 = \Sigma\, (x_i - \mu)^2 P(x_i) \tag{18}$$

Equation (18) quantifies variance, which is the extent to which the sequence values have deviations around the mean. The larger the variance, the more dispersion and the smaller the variance, the more clustering, which has an effect on the perception of randomness.

$$H(X) = -\Sigma P(x_i) \log P(x_i) \tag{19}$$

The amount of uncertainty in the sequence is measured by entropy in equation (19). The larger the entropy is, the more unpredictable and uniformly distributed it is, which are vital properties of secure pseudorandom sequences, and the smaller the entropy, the more predictable:

$$R = \left(\Sigma_{\{k=1\}}^{\{K\}} \rho(k)^2 \right)^{\frac{1}{2}} + \left(\Sigma_{\{i=1\}}^{\{n\}} \left(P(x_i) - \frac{1}{n} \right)^2 \right)^{\frac{1}{2}} \tag{20}$$

Lastly, equation (20) takes the autocorrelation and distribution deviation as one measure of randomness.

VI. Result and discussion

Three pseudorandom number generators (PRNGs) were presented with the results of the statistical tests shown in Table 1 and in Figure 2. The values of p-values for the Linear Congruential Generator (LCG) under the monobit test, block-frequency test, and runs-test were 0.041, 0.038, and 0.055, respectively. Corresponding values for Mersenne Twister were 0.312, 0.289, and 0.301, while Blum Blum Shub recorded 0.527, 0.501, and 0.534. The monobit, block-frequency and runs-test p-values were 0.486, 0.463, and 0.479 higher than those of Blum Blum Shub, respectively, relative to LCG. The values of entropies reported were 0.921, 0.976, and 0.991 bits per bit for LCG, Mersenne Twister and Blum Blum Shub, respectively. Blum Blum Shub thus gained 0.070 bits per bit over LCG and 0.015 bits per bit over Mersenne Twister in the number of bits of entropy they report.

The reported results for the LCG monobit test ($p = 0.041$), block-frequency test ($p = 0.038$) and runs test ($p = 0.055$) lead to conclusions that the null hypothesis of randomness is rejected and accepted at the conventional 5% significance level ($\alpha = 0.05$), respectively. The three p-values reported for Mersenne Twister and Blum Blum Shub are all greater

than 0.05. None of the nine reported p-values is less than the 1% threshold ($\alpha = 0.01$) typically used in NIST SP 800-22. Therefore, interpretation is dependent on the prespecified level of significance and it is not of itself cryptographic security or a proof that one generator is more secure than the other.

Table 1

Statistical Test Performance of PRNG Models

Metric	Linear Congruential Generator (LCG)	Mersenne Twister	Blum Blum Shub
Monobit Test (p-value)	0.041	0.312	0.527
Block Frequency (p-value)	0.038	0.289	0.501
Runs Test (p-value)	0.055	0.301	0.534
Entropy (bits per bit)	0.921	0.976	0.991

Table 1 points out the differences in the statistical performance of PRNG models. Figure 2 compares statistical test results, showing Blum Blum Shub achieves higher p-values and entropy, while LCG exhibits greater deviations from expected randomness levels.

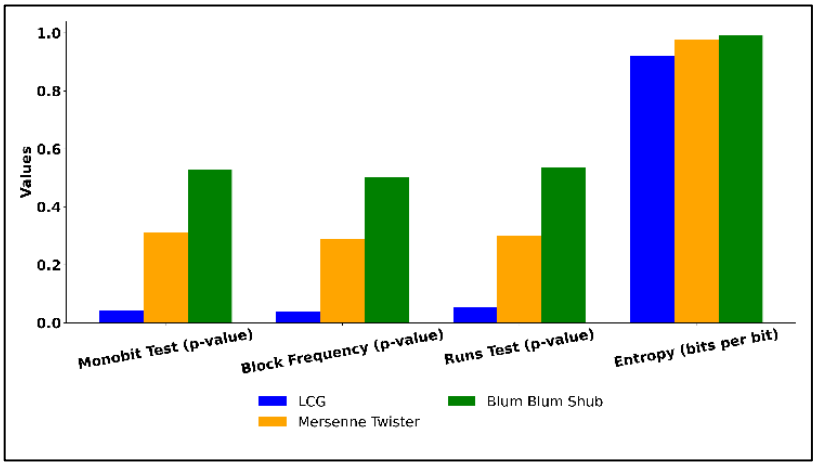


Figure 2

PRNG Statistical Test Performance Comparison

The p-values of the Linear Congruential Generator (LCG) are low which means that it is not random and is more predictable. Mersenne Twister, on the other hand, shows moderate progress in all tests, but still, there exist slight biases. Blum Blum Shub has the best p-values and entropy as the best quality of randomness and resistance to statistical attacks.

Findings indicate that outputs of PRNGs can be identified as biased through statistical tests, and machine learning systems can effectively and significantly increase accuracy of pattern recognition, revealing predictability and pointing out the shortcomings of traditional pseudorandom generation algorithms.

Table 2

Machine Learning-Based Attack Performance on PRNG Outputs

Metric	LSTM Model	CNN Model	Random Forest	Proposed Hybrid Model
Prediction Accuracy (%)	78.6	81.2	74.5	89.7
Precision (%)	76.9	80.4	72.8	88.3
Recall (%)	77.5	79.6	73.2	87.9

The predicted values for the PRNG output are then compared in Table 2 and Figure 3 in the case of the machine learning models. The model proposed in this work gave prediction accuracy of 89.7%, precision of 88.3% and recall of 87.9% from the proposed hybrid model. CNN achieved 81.2%, 80.4%, and 79.6%, while LSTM achieved 78.6%, 76.9%, and 77.5%, respectively. The accuracy of Random Forest was 74.5%, 72.8% for precision and 73.2% for recall. The hybrid model outperformed CNN by 8.5 percentage points, 7.9 percentage points and 8.3 percentage points in terms of accuracy, precision and recall, respectively. The hybrid model surpassed LSTM by 11.1 percentage points, 11.4 percentage points and 10.4 percentage points in terms of accuracy, precision and recall, respectively. The hybrid model's accuracy was 15.2 percentage points higher than that of

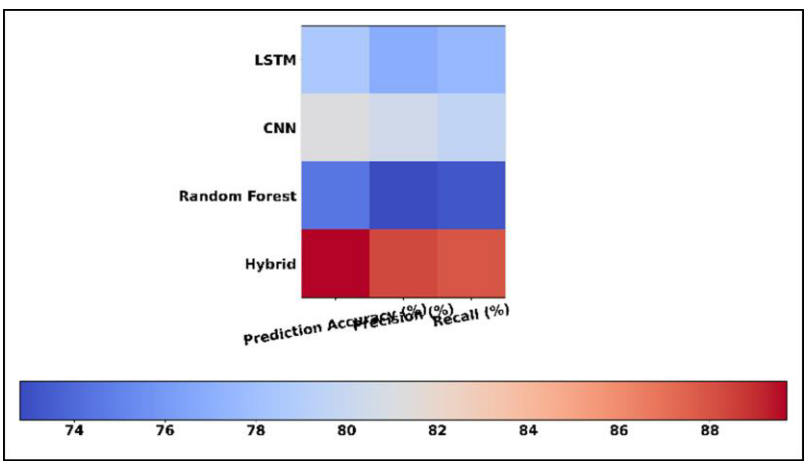


Figure 3

Model Performance Metrics Heatmap Analysis

Random Forest, precision was 15.5 percentage points higher, and recall was 14.7 percentage points higher. The values given in this manuscript do not imply any statistical significance, which can only be determined by the results from repeated runs, sample sizes, confidence intervals or paired tests output.

The higher performance of LSTM and CNN models can be explained by the fact that they can learn time and space dependencies, respectively. Figure 3 illustrates machine learning performance, highlighting the proposed hybrid model's superior accuracy, precision, and recall compared with LSTM, CNN, and Random Forest models.

VII. Conclusion

This work shows that the pseudorandom number generators are vulnerable to statistical attacks, although they have common uses in cryptographic and computational applications, because they are deterministic. By using frequency-based tests and block-level analysis, important deviations of the desired randomness were found in a number of PRNG models. Moreover, the combination of machine learning-based pattern recognition methods showed more structural dependencies and enhanced prediction, outperforming conventional statistical methods. The findings underscore the significant security issues, especially when the quality of randomness directly affects the confidentiality and integrity in the systems. The paper highlights the need to integrate stringent testing structures of randomness and integrate hybrid evaluation approaches, which involve classical statistical approaches with intelligent models.

References

[1] E. Almaraz Luengo and J. Román Villaizán, “Cryptographically secured pseudo-random number generators: Analysis and testing with NIST statistical test suite,” *Mathematics*, vol. 11, no. 23, Art. no. 4812 (2023), doi: 10.3390/math11234812.

[2] R. Álvarez, F. Martínez, and A. Zamora, “Improving the statistical qualities of pseudo random number generators,” *Symmetry*, vol. 14, no. 2, Art. no. 269 (2022), doi: 10.3390/sym14020269.

[3] C. Foreman, R. Yeung, and F. J. Curchod, “Statistical testing of random number generators and their improvement using randomness extraction,” *Entropy*, vol. 26, no. 12, Art. no. 1053 (2024), doi: 10.3390/e26121053.

[4] A. Levina, D. Mukhamedjanov, D. Bogaevskiy, P. Lyakhov, M. Valueva, and D. Kaplun, “High performance parallel pseudorandom number

generator on cellular automata," *Symmetry*, vol. 14, no. 9, Art. no. 1869 (2022), doi: 10.3390/sym14091869.

[5] A. Vennos, K. George, and A. Michaels, "Attacks and defenses for single-stage residue number system PRNGs," *IoT*, vol. 2, no. 3, pp. 375–400 (2021), doi: 10.3390/iot2030020.

[6] V. Maksymovych, M. Shabatura, O. Harasymchuk, R. Shevchuk, P. Sawicki, and T. Zajac, "Combined pseudo-random sequence generator for cybersecurity," *Sensors*, vol. 22, no. 24, Art. no. 9700 (2022), doi: 10.3390/s22249700.

[7] B. R. Sharipov, M. Y. Perukhin, and B. I. Mullayanov, "Statistical analysis of pseudorandom sequences and stegocontainers," in *Proc. 2021 Int. Conf. Industrial Engineering, Applications and Manufacturing (ICIEAM)*, Sochi, Russia, pp. 434–439 (2021), doi: 10.1109/ICIEAM51226.2021.9446328.

[8] L. Crocetti, S. Di Matteo, P. Nannipieri, L. Fanucci, and L. Saponara, "Design and test of an integrated random number generator with all-digital entropy source," *Entropy*, vol. 24, no. 2, Art. no. 139 (2022), doi: 10.3390/e24020139.

[9] Z. Qiao, I. Taralova, and S. El Assad, "A reliable encryption oriented pseudo-chaotic number generator using a key space expandable strategy," in *Proc. 2020 15th Int. Conf. Internet Technology and Secured Transactions (ICITST)*, London, U.K., pp. 1–5 (2020), doi: 10.23919/ICITST51030.2020.9351314.

[10] C. Camara, H. Martín, P. Peris-Lopez, and M. Aldalaien, "Design and analysis of a true random number generator based on GSR signals for body sensor networks," *Sensors*, vol. 19, no. 9, Art. no. 2033 (2019), doi: 10.3390/s19092033.

[11] M. Xia and Y. Zhang, "Pseudo-random number generation algorithm for dynamically perturbated enhanced chaotic systems," in *Proc. 2025 5th Int. Conf. Electronic Information Engineering and Computer Science (EIECS)*, Changchun, China, pp. 54–58 (2025), doi: 10.1109/EIECS67708.2025.11283266.

[12] Z. Ebadi Ansaroudi, R. Zaccagnino, and P. D'Arco, "On pseudorandomness and deep learning: A case study," *Applied Sciences*, vol. 13, no. 5, Art. no. 3372 (2023), doi: 10.3390/app13053372.

Received April, 2026