

MACHINE LEARNING-BASED EVALUATION OF BINARY SEQUENCE RANDOMNESS FOR SECURE COMMUNICATIONS

Remus-Florin STANCA

“Nicolae Bălcescu” Land Forces Academy, Sibiu, Romania

remus.stanca@campus.utcluj.ro

Abstract: Machine learning is an emerging domain with applications across a wide range of fields. This paper investigates a machine learning-based approach for assigning scores to digital sequences based on their level of randomness, using statistical indicators commonly employed in classical randomness evaluation tests. A total of 15,000 binary sequences were generated for training and testing, covering six classes: Advanced Encryption Standard in Counter Mode (AES-CTR), ChaCha20, Mersenne Twister, Linear Congruential Generator (LCG), biased random sequences, and periodic sequences. For each sequence, several statistical features were computed, including Shannon entropy, Exclusive OR (XOR) relation score, and binary matrix rank. These features were used to train a machine learning model based on the XGBoost algorithm, implemented in Python. The proposed model successfully distinguishes statistically degraded sequences (periodic and biased) from statistically uniform ones. However, high-quality pseudorandom generators exhibit overlapping behavior, indicating similar statistical properties, especially for short sequences. The level of randomness is an important factor in determining whether a cryptographic key is suitable for use in secure encryption algorithms.

Keywords: pseudorandom number generators, randomness evaluation, machine learning

1. Introduction

Cryptographic key generation is an essential process for encryption algorithms, as in modern cryptography the secret is given only by the key itself, since the algorithm is publicly available and any potential attacker can have access to it [1]. Therefore, for secure military transmissions, it is important to use perfectly random keys and a powerful encryption algorithm that induces a high level of diffusion and confusion in the plaintext [2].

In military environments, the secure transmission of information is essential for preserving the confidentiality and integrity of operational data, as well as for safeguarding mission-critical decisions. Any compromise of transmitted data may

lead to the exposure of sensitive plans, undermining operational effectiveness and potentially jeopardizing personnel and strategic objectives.

In the current state of the art, there are statistical test suites used to determine whether certain binary sequences are random, based on characteristics such as Shannon entropy, runs of ones, linear complexity, etc [3], [4].

In [5], it is proposed a machine learning mechanism based on a neural network trained using different statistical metrics in order to predict the results of the NIST STS tests with an accuracy of 95%. Their work remains limited to binary classification.

In [6], the authors of the study implemented a machine learning program by combining different convolutional neural networks

(AlexNet, ResNet) in order to make a binary classification between random and non-random sequences with an accuracy of 87%.

Recent advances in machine learning have enabled the construction of data-driven distinguishers for cryptographic primitives. In the work [7] the authors propose a general methodology for automatically generating distinguishers using machine learning techniques. The problem is formulated as a classification task, where the goal is to determine whether a given sequence originates from one generator or another. The study demonstrates that machine learning models can achieve a measurable advantage in distinguishing between outputs of different pseudorandom generators, including NIST DRBG constructions. However, the results also reveal important limitations, such as the so-called blind spot paradox, where models may appear to perform better than random guessing without truly capturing meaningful structural differences. Moreover, the accuracy gains remain relatively small, suggesting that high-quality generators are inherently difficult to distinguish using such approaches.

A complementary direction is explored in [8], where deep learning is applied to distinguish outputs of lightweight block ciphers from truly random sequences. The authors employ a convolutional Siamese neural network to learn similarity relationships between sequences, reframing the problem beyond simple classification. Their results show that deep learning models can achieve very high accuracy (over 99%) in distinguishing cipher outputs from random data in controlled scenarios. This indicates that neural networks are capable of identifying subtle statistical patterns not captured by traditional tests. Nevertheless, the approach relies on large datasets and complex architectures, and its effectiveness is demonstrated primarily on specific ciphers and experimental setups, raising concerns about generalization and

scalability.

Despite these advances, all of these papers rely on binary classifications used to distinguish between random and non-random sequences and often rely on complex machine learning models such as neural networks. While they can successfully detect statistical irregularities, they still do not provide a continuous evaluation of pseudorandomness quality and are limited when it comes to short binary sequences. Moreover, distinguishing between high-quality random generators remains an inherent limitation, as these generators are constructed in this way to remain immune to statistical distinguishers. In this context, this paper proposes a feature-based machine learning approach for evaluating the statistical randomness of binary sequences. Unlike existing methods, the proposed model provides a continuous randomness score rather than a binary classification and is specifically analysed on short sequence lengths ranging from 64 to 1024 bits. The experimental results demonstrate that the model effectively separates statistically degraded sequences from statistically uniform ones, while also highlighting the limitations of feature-based approaches in distinguishing high-quality pseudorandom generators. Additionally, the study shows that the scoring behaviour becomes more stable as the sequence length increases, providing insight into the relationship between sequence size and statistical reliability.

2. Materials and methods

The proposed approach consists in generating a feature based machine learning model which evaluates the randomness quality of data. The methodology consists in three stages: data generation, feature extraction and data training.

2.1. Data generation

The dataset is composed of binary sequences generated using different sources with different statistical properties. The high quality pseudo-random sequences

were generated using cryptographic primitives such as AES in CTR mode and ChaCha20 as well as Mersenne Twister algorithm. In addition, low quality sequences were generated using LCG, biased random generators and periodic patterns.

For each type of generator, different lengths sequences were produced (64, 128, 256, 512, and 1024 bits) in order to produce a better evaluation of the behaviour of the model on different sizes.

2.1.1. Generation of AES in CTR mode and ChaCha20 binary sequences

Both AES and ChaCha20 sequences were generated using the Python PyCryptodome library. For each generated sequence, a random nonce and a random cryptographic key were generated using a secure random source. The resulting ciphers were applied to a zero-valued plaintext block, and the output bit-stream was truncated to its desired length.

2.1.2. Generation of LCG binary sequences

The Linear Congruential Generator (LCG) sequences were generated using the following classical implementation:

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

Where the parameters were set to $m=2^{31}$, $a=1103515245$, $b=12345$ and the initial seed X_0 was randomly generated for each individual sequence using a secure random generator implemented and standardized in Python.

Each resulting value was converted into its corresponding binary representation and concatenated until the desired sequence length was obtained.

2.1.3. Generation of Mersenne Twister sequences

Mersenne Twister sequences were generated using the built-in “random” Python module, which implements the MT19937 pseudorandom number generator. The generator was initialized using a system-derived random seed. At

each iteration, 32-bit pseudorandom integers were generated, converted into their binary representation, and concatenated until the desired sequence length was obtained.

2.1.4. Biased random sequence generation process

Biased random sequences were generated using Python’s built-in pseudorandom number generator. For each generated sequence, a randomly selected probability for the occurrence of the bit “1” was chosen within the interval [0.55, 0.9]. The binary values were then generated by taking into account the specific probability selected for the occurrence of “1”, producing statistically non-uniform random sequences with different levels of imbalance between the values of “1” and the values of “0”

2.1.5. Periodic binary sequence generation process

Periodic sequences were generated using Python’s built-in pseudorandom number generator. For each generated sequence, a random period length between 2 and 16 bits was selected. A random binary pattern of the corresponding length was then generated and repeatedly concatenated until the desired sequence length was obtained.

2.2. Score assignment process

Since no standardized continuous score assignment tool implemented using machine learning exists for evaluating the randomness level of short binary sequences, the specific scale for each category of sequence was manually defined. High-performance cryptographic primitives such as AES-CTR and ChaCha20 were assigned high target score values, while statistically degraded sequences were assigned significantly lower values.

For each class, the target score was sampled from a Gaussian normal distribution using a specific mean value and a fixed standard deviation. This approach introduces relative variability into each class of generated sequences, preventing the model from learning deterministic labels. The selected score distributions were chosen to reflect

the expected statistical behavior of the generators rather than their cryptographic security.

AES-CTR and ChaCha20 sequences were assigned a mean score value of 96 with a standard deviation of 1, Mersenne Twister sequences received a mean value of 77 with a standard deviation of 4, LCG sequences were assigned a mean value of 55 with a standard deviation of 7, biased sequences received a mean value of 40 with a standard deviation of 10, while periodic sequences were assigned a mean value of 8 with a standard deviation of 4.

2.3. Feature extraction

Each binary sequence is characterised using a set of statistical and structural features

designed to quantify different aspects of randomness. These features include:

- Shannon entropy
- Bit balance (proportion of ones)
- Runs count and maximum run length
- Autocorrelation (lags 1, 2 and 4)
- Bit pattern frequencies (00, 11)
- XOR- based correlation metrics
- Low-bit transition statistics
- Binary matrix rank
- Lempel-Ziv complexity
- Compression ratio

These features provide a compact representation of the statistical behavior of each sequence.

Table 1. Feature description table

Feature	Description
Shannon entropy	Measures the uncertainty and information content of the sequence
Bit balance	Ratio between the number of ones and zeros
Runs count	Number of transitions between consecutive bits, normalized to sequence length
Maximum run length	Longest sequence of consecutive identical bits
Autocorrelation (lags 1,2,4)	Measures similarity between the initial sequence and the shifted versions of the sequence
Frequency of 00 and 11 patterns	Measures the occurrence frequency of repeated adjacent bits.
XOR relation score	Measures statistical dependence between bits separated by multiple lag values using XOR operations.
Low-bit transition statistics	Evaluates transition uniformity between consecutive low-order bit groups using a chi-square-based metric.
Binary matrix rank	Estimates linear dependence by computing the rank of binary matrices extracted from the sequence.
Lempel-Ziv complexity	Measures sequence complexity based on the number of distinct encountered patterns.
Compression ratio	Ratio between compressed and original sequence size, used as an indicator of compressibility.

2.3.1. XOR relation score

The XOR relation score measures the statistical dependencies between bits separated by a distance k within the binary

sequence, which is computed as:

$$p_k = \frac{1}{N-k} \cdot \sum_{i=0}^{N-k} (x_i \oplus x_{i+k} = 0) \quad (2)$$

Where N represents the sequence length and “ $\oplus$ ” represents the exclusive or operation (XOR). The final value is computed as it follows:

$$S = \max_k |p_k - 0.5| \quad (3)$$

Higher values indicate the existence of stronger dependencies and possible correlations induced by the statistical properties associated with statistically degraded sequences.

The proposed XOR relation score was inspired by XOR bias and correlation analysis techniques used in randomness evaluation literature [9].

2.3.2. Binary matrix rank

The binary matrix rank quantifies the degree of linear independence related to a binary sequence. The binary sequence is shaped into a matrix M , and its rank is computed over the $GF(2)$ field using the Gaussian elimination technique:

$$R = \frac{\text{rank}(M)}{\min(m,n)} \quad (4)$$

[3]

m and n represent the dimensions of the derived matrix, and values closer to 1 are associated with lower linear dependencies, which is one of the characteristics related to pseudorandom binary sequences.

2.3.3. Lempel-Ziv complexity

Lempel-Ziv complexity measures the number of distinct substrings encountered while parsing the sequence from left to right. If $C(n)$ denotes the number of discovered patterns for a sequence of length n , the normalized complexity is defined as:

$$LZ = \frac{C(n)}{n} \quad (5) [10]$$

Higher values indicate increased structural complexity and lower repetitiveness.

2.4. Model training

The extracted statistical features previously explained in this paper were used to train a machine learning model based on XGBoost regression using the “XGBoost” Python library. The dataset was divided into training and testing subsets using an 80/20 split ratio with a fixed random state value of 42 in order to ensure reproducibility.

The XGBoost model was configured using 300 estimators, a maximum tree depth of 6, a learning rate of 0.05, a subsampling ratio of 0.9, and a column subsampling ratio of 0.9. The model performance was evaluated using Mean Absolute Error value (MAE) and the coefficient of determination (R^2).

2.5. Evaluation Procedure

The already trained model was applied onto a dataset of 30,000 of binary sequences of different lengths (64, 128, 256, 512 and 1024 bits) and the resulting scores were exported into Comma separated vector files (CSV). For each length and each class of binary sequences were calculated the mean value and the standard deviation value. This allowed the assessment of both the separation capability of the model and the stability of the scoring mechanism across different sequence lengths.

3. Results and discussions

The proposed model was evaluated on binary sequences generated using multiple sources with different statistical properties, including AES, ChaCha20, Mersenne Twister, LCG, biased generators, and periodic sequences. The evaluation was performed for sequence lengths ranging from 64 to 1024 bits.

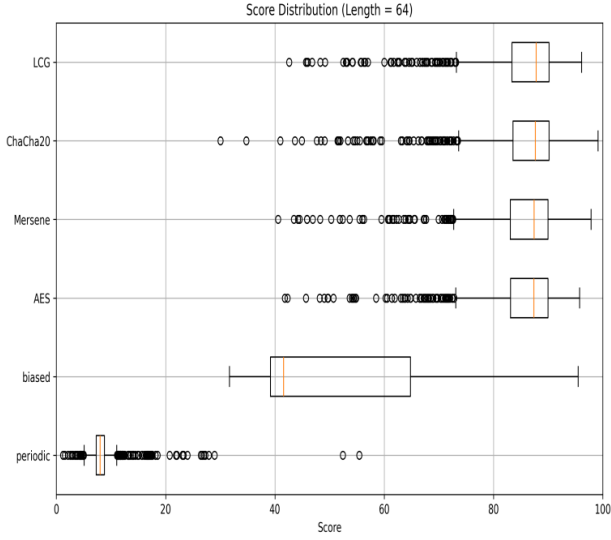


Fig. 1(a). Boxplot representation for score distributions of 64 bit length sequences

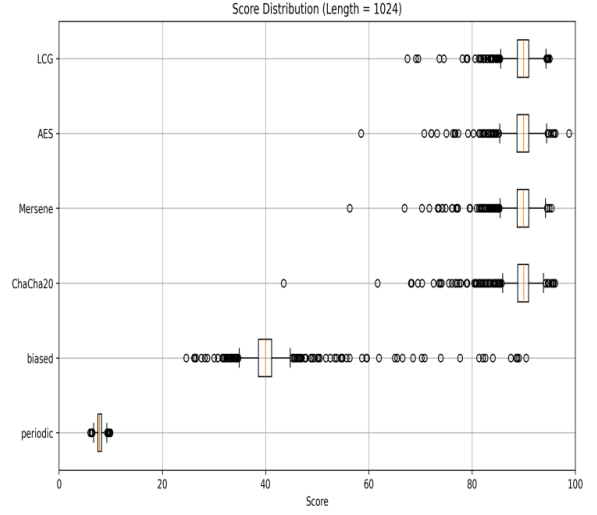


Fig. 1(b). Boxplot representation for score distributions of 1024 bit length sequences

The performance of the proposed model was evaluated using boxplot representations of the score distributions for binary sequences of different lengths. Fig. 1(a) and Fig. 1(b) present the results for 64-bit and 1024-bit sequences, respectively.

The results show a clear separation between statistically degraded sequences and those exhibiting strong randomness properties. Periodic sequences consistently obtain very low scores, with narrow distributions, indicating both poor randomness and high stability. In contrast, biased generators produce higher scores than periodic sequences, but still remain significantly below the rest, with a noticeably larger spread.

All pseudorandom generators (AES, ChaCha20, Mersenne Twister, and LCG) achieve high scores, with median values grouped in the upper range. However, their distributions overlap to a large extent, making it difficult to distinguish between them using the selected features.

A visible effect of sequence length can be observed when comparing the two figures. For 64-bit sequences, the distributions are wider and contain more outliers, suggesting higher variability in the evaluation. For 1024-bit sequences, the distributions become more compact, with reduced interquartile ranges and fewer extreme values. This indicates that the scoring becomes more stable as the sequence length increases.

It is also worth noting that, although LCG is known to have structural weaknesses, its score distribution is very similar to those of AES and ChaCha20. This suggests that the selected statistical features are not sufficient to capture deeper structural differences, especially when working with short sequences.

Overall, the proposed approach is effective in identifying statistically weak sequences, but shows clear limitations when attempting to differentiate between high-quality pseudorandom generators.

Table 2: Mean and standard deviation of randomness scores for different generators and sequence lengths

Generator	64 (mean $\pm$ std)	128	256	512	1024
AES	85.29 $\pm$ 7.53	86.56 $\pm$ 6.82	88.36 $\pm$ 4.39	88.54 $\pm$ 4.39	89.63 $\pm$ 2.77
ChaCha20	85.45 $\pm$ 8.22	86.6 $\pm$ 6.81	88.12 $\pm$ 4.68	88.4 $\pm$ 4.72	89.48 $\pm$ 3.38
Mersenne	85.32 $\pm$ 7.57	87.01 $\pm$ 6.54	88.15 $\pm$ 5.09	88.61 $\pm$ 4.27	89.56 $\pm$ 2.94

LCG	85.46 ± 7.83	87.28 ± 5.88	88.29 ± 4.74	88.95 ± 3.69	89.65 ± 2.46
Biased	51.83 ± 17.9	47.39 ± 14.56	44.13 ± 11.29	42.68 ± 9.47	40.62 ± 6.04
Periodic	8.55 ± 3.48	8.22 ± 1.0	8.02 ± 0.62	7.92 ± 0.69	7.88 ± 0.56

In addition to the visual analysis, Table 1 provides a more precise numerical view of the results, showing the mean scores and standard deviations for each generator and sequence length.

The values confirm the observations from the boxplots. Periodic sequences remain at very low scores (around 8) with extremely small standard deviations, which shows that their behavior is very stable but clearly non-random. Biased generators have higher scores, but still far from the others, and also present larger standard deviations, especially for shorter lengths, indicating more variability.

For the pseudorandom generators, the mean values are consistently high and very close to each other. For example, at 1024 bits, all generators (AES, ChaCha20, Mersenne, and LCG) reach scores around 89–90, with relatively small standard deviations. This again shows that the model treats them in a similar way.

Another aspect that can be observed from the table is the effect of sequence length. As the length increases, the mean scores tend to slightly increase, while the standard deviation decreases. This suggests that the evaluation becomes more stable for longer sequences, which is also consistent with what was seen in the boxplots.

At the same time, the table highlights a limitation: even though LCG is known to be weaker, its numerical results are very similar to those of AES and ChaCha20. This indicates that the selected features are not sufficient to clearly separate high-quality generators.

On the other hand, the performance of the proposed XGBoost regression model was evaluated based on Mean Absolute Error value (MAE) and the coefficient of determination (R^2).

The MAE and R^2 values were computed by comparing the predicted randomness scores

generated by the XGBoost model with the assigned target scores from the testing dataset. The obtained values are MAE=7.85 and $R^2=0.893$. The relatively low MAE value indicates that the predicted scores remain close to the manually assigned values, while the high value related to R^2 parameter reveals that the model captures the relationship between the extracted features and the assigned target scores. However, the overlapping scores obtained for the high performance cryptographic primitives (AES and ChaCha20) and some low quality randomly generated sequences (Mersenne Twister and LCG) indicate that distinguishing between low quality random sequences and high quality primitives remains difficult when the trained model relies on statistical features, especially for short binary sequences.

4. Conclusions

This paper presents a feature-based machine learning approach for evaluating the statistical randomness of binary sequences using a continuous scoring mechanism. The proposed XGBoost model, trained on a set of statistical and structural features, proves effective in separating statistically degraded sequences from statistically uniform ones, while also demonstrating increased stability as sequence length grows.

The experimental results show that periodic and biased sequences consistently receive low scores, whereas pseudorandom generators obtain high and closely grouped values. At the same time, the overlap between high-quality generators such as AES, ChaCha20, and Mersenne Twister highlights the inherent difficulty of distinguishing between them using statistical features, especially for short sequences. A similar behavior is observed for the Linear Congruential Generator

(LCG), which, despite its known structural weaknesses, achieves scores comparable to those of high-quality generators, indicating that the selected features are not sufficient to capture deeper structural deficiencies.

Overall, the proposed approach offers a practical alternative to traditional binary classification methods, providing a more detailed evaluation of randomness quality. However, the proposed model evaluates statistical randomness properties rather than the cryptographic security of the generators

themselves. As demonstrated by the obtained results, generators with known structural weaknesses, such as LCG, may still obtain high scores due to statistical similarities with cryptographically secure generators over short binary sequences. The results confirm both the applicability and the limitations of feature-based machine learning models in this context, supporting further research towards more advanced and discriminative evaluation techniques.

References

- [1] Paar C, Pelzl J. *Understanding Cryptography: A Textbook for Students and Practitioners*. Berlin, Heidelberg: Springer; 2010. DOI: 10.1007/978-3-642-04101-3.
- [2] Shannon CE. *Communication Theory of Secrecy Systems*. Bell System Technical Journal. 1949;28(4):656–715. DOI: 10.1002/j.1538-7305.1949.tb0092.
- [3] Bassham LE, Rukhin AL, Soto J, Nechvatal JR, Smid ME, Leigh SD, et al. *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*. NIST [Internet]. 2010 [cited 2026 May 1]; Available from: <https://www.nist.gov/publications/statistical-test-suite-random-and-pseudorandom-number-generators-cryptographic>.
- [4] L’Ecuyer P, Simard R. TestU01: A C library for empirical testing of random number generators. *ACM Trans Math Softw*. 2007;33(4):1–40. DOI: 10.1145/1268776.1268777.
- [5] Nagy I, Suci A. Randomness Testing with Neural Networks. In: 2021 IEEE 17th International Conference on Intelligent Computer Communication and Processing (ICCP). 2021. p. 431–6.
- [6] Kaner S, Garipcan AM, Erdem E. A novel deep learning-based statistical randomness evaluation test methodology for cryptographic applications. *J King Saud Univ Comput Inf Sci*. 2025;37(9):264. DOI: 10.1007/s44443-025-00271-4.
- [7] Brunetta C, Picazo-Sanchez P. Modelling cryptographic distinguishers using machine learning. *J Cryptogr Eng*. 2022;12(2):123–35. DOI: 10.1007/s13389-021-00262-x.
- [8] Ebadi Ansaroudi Z, Zaccagnino R, D’Arco P. On Pseudorandomness and Deep Learning: A Case Study. 2023; DOI: 10.3390/app13053372.
- [9] Davies RB. Exclusive OR (XOR) and hardware random number generators [Internet]. 2002 [cited 2026 May 18]. Available from: <https://www.robertnz.net/pdf/xor2.pdf>.
- [10] Lempel A, Ziv J. On the Complexity of Finite Sequences. *IEEE Trans Inf Theory*. 1976; 22(1): 75–81.