Научная статья

УДК 004.421.5
DOI 10.25205/1818-7900-2025-23-1-33-45

Сравнительный анализ статистических тестов на основе теста сжатия данных и стандартных тестов для оценки случайности генераторов случайных чисел

Йешевас Гетачеу Лулу

Новосибирский государственный университет
Новосибирск, Россия

j.lulu@g.nsu.ru, https://orcid.org/0009-0006-8054-9846

Аннотация

В работе представлен подробный сравнительный анализ статистических тестов, использующих как современные компрессоры данных, так и стандартные статистические методы для оценки случайности генераторов случайных чисел (ГСЧ). Цель исследования – всесторонне оценить эффективность и результативность этих тестов при определении качества выходных последовательностей ГСЧ.

Методы сжатия данных давно признаны эффективными статистическими тестами, причем некоторые из них являются асимптотически оптимальными. В статье сравнивается эффективность этих тестов, основанных на сжатии данных, с традиционными статистическими тестами при оценке свойств случайности ГПЗУ.

Результаты исследования показывают, что эффективность тестов на основе компрессора данных и стандартных статистических тестов практически одинакова. Проведенные эксперименты и анализ, показали, что оба подхода дают сопоставимые результаты при оценке случайности выходных последовательностей ГПЗУ.

Ключевые слова

статистические тесты сжатия данных, генератор случайных чисел, статистический тест Национального института стандартов и технологий, тест TESU01

Благодарности

Автор выражает искреннюю благодарность научному руководителю, профессору Б. Я. Рябко за неоценимую помощь в критическом обзоре эксперимента и содержательные комментарии к рукописи.

Для цитирования

Лулу Й. Г. Сравнительный анализ статистических тестов на основе теста сжатия данных и стандартных тестов для оценки случайности генераторов случайных чисел // Вестник НГУ. Серия: Информационные технологии. 2025. Т. 23, № 1. С. 33–45. DOI 10.25205/1818-7900-2025-23-1-33-45

Comparative Analysis of Statistical Test based onData compression methods and standard tests for assessing Randomness of Random Number Generators

Yeshewas Getachew Lulu

Faculty of Information TechnologyNovosibirsk State University
Novosibirsk, Russian Federation

j.lulu@g.nsu.ru

Abstract

This paper presents a detailed comparative analysis of statistical tests utilizing both modern data compressors and standard statistical methods for assessing the randomness of Random number generators (RNG). Our study aims to thoroughly evaluate the efficiency and performance of these tests in determining the quality of RNG output sequences. Data compression techniques have long been recognized as effective statistical tests, with some being asymptotically optimal. We compare the effectiveness of these data compressor-based tests with traditional statistical tests in assessing the randomness properties of RNG.

Our results demonstrate that the efficiency of data compressor tests and standard statistical tests is closely similar. Through rigorous experimentation and analysis, we show that both approaches yield comparable results in evaluating the randomness of RNG output sequences.

Keywords

Data compression statistical tests, random number generator, National Institute of Standards and Technology statistical test, TESU01 test

Acknowledgement

I would like to express my sincere gratitude to my supervisor, Professor Ryabko Boris Yakovlevich, for his invaluable assistance with the critical review of the experiment and insightful comments on the manuscript.

For citation

Yeshewas Getachew Lulu. Comparative Analysis of Statistical Test based onData compression methods and standard tests for assessing Randomness of Random Number Generators. *Vestnik NSU. Series: Information Technologies*, 2025, vol. 23, no. 1, pp. 33–45 (in Russ.) DOI 10.25205/1818-7900-2025-23-1-33-45

1. Introduction

Data compression techniques have been recognized as effective statistical tests since paper [1] was published. It has been demonstrated in the literature that data compression tests are asymptotically optimal [2]. We compare the effectiveness of these data compressor-based tests with other well-known test suites such as NIST and TestU01 in assessing the randomness properties of RNG.

Random number generators stand as fundamental components across a wide spectrum of computational domains, ranging from simulations and cryptography to statistical sampling [3–5]. These methods play a critical role in generating sequences of numbers that mimic randomness, a crucial requirement in various applications. Selecting the appropriate RNG is vital to uphold the integrity and dependability of the generated sequences, as biased or predictable outputs can undermine the validity of computational results and compromise security measures [5].

In response to the imperative need for robust RNG, extensive research efforts have been directed towards developing and evaluating these algorithms. A key aspect of this evaluation involves subjecting RNGs to rigorous testing procedures to ascertain their randomness properties. Numerous statistical test suites have emerged to facilitate this evaluation process, enabling researchers and practitioners to measure the performance of RNGs under different testing methodologies [6].

Among the prominent test suites employed for assessing the randomness of RNGs, the National Institute of Standards and Technology (NIST) test suite [7], TestU01 [8], and data compressor tests [9], DIEHARD [10], CRYPT-X [11], a test based on diffusion characteristic of a block cipher [12],

topological binary test defined by Alcover et al [13]. hold significant prominence. These suites offer a comprehensive battery of statistical tests designed to probe various aspects of randomness, including uniformity, independence, and unpredictability. By subjecting RNG outputs to these tests, researchers can gain insights into the strengths and weaknesses of different algorithms, thereby informing their selection for specific applications [14].

Through rigorous experimentation and analysis conducted in this study, four weak RNG (RANDU, linear congruential generator, linear-feedback shift register, Middle Square Generator) and three strong generators (Mersenne Twister, permuted congruential generator, Blum Blum Shub Generator) were examined. Various file lengths (1 KB, 10 KB, 100 KB,500KB,1 MB) with 100 sequences each were utilized.

Therefore, this study aims a comparative analysis of RNGs using standard batteries of tests and test based on modern data compression.

The rest of the paper is organized as follows. In section 2 contains methodology used for evaluation of RNG, In Sections 3 we investigate the experimental setup and testing environment, in section 4 experimental results will be shown, in section 5 discussion and interpretation of experimental results are shown and conclusion will be drawn in final sections.

2. Methodology

In this section, we detail the methodologies employed to evaluate the performance of Random number generators (RNGs) through three distinct approaches: The Data Compression Test, the NIST Test Suite, and the TestU01 (Big-Crush) Statistical Test Suite.

2.1. Data Compression Test

To assess the randomness of RNG output, we examine its compressibility using modern compression algorithms such as Gzip [15], LZMA2[16], Brotli [18], Zstd [19], and LZFSE [20]. In this context, we consider an alphabet size of $|A| = 256$, corresponding to 2^8, with $n \log |A|$ representing the length of the file in bits before compression (where n denotes the length in bytes) [1] [21]. For a significance level of $\alpha = 0.01$, the hypothesis regarding randomness (H0) is rejected if the length of the compressed file is equal to or less than $n \log |A| - 8$ bits [21]. To simplify, considering $\alpha \leq 2^{-7} = 1/128$, if the n-byte file is compressed (meaning the length of the encoded file is $n - 1$ bytes or fewer), it is deemed non-random, leading to the rejection of H0.

Flow Chart Approach: The Data Compression Test approach typically involves the following steps:

1. ***Generation of RNG Sequences:*** *First, RNG sequences are generated using selected algorithms.*
2. ***File Generation:*** *These sequences are saved into files with a specified length.*
3. ***Compression:*** *The files are then compressed using modern compression algorithms.*
4. ***Length Comparison:*** *The length of the compressed file is compared against the critical value based on the alphabet size and original file length.*
5. ***Hypothesis Testing:*** *Finally, hypothesis testing is performed based on the comparison results.*

2.2. NIST Test Suite

The NIST Test Suite is a battery of statistical tests developed by the National Institute of Standards and Technology (NIST) to assess the randomness of sequences [7].

Flow Chart Approach: The evaluation process using the NIST Test Suite typically follows these steps:

1. ***Sequence Generation:*** *RNG sequences are generated.*
2. ***Test Application:*** *The generated sequences are subjected to various statistical tests provided by the NIST Test Suite.*
3. ***Result Analysis:*** *The results obtained from the tests are analyzed to determine the randomness and quality of the sequences.*

2.3. TestU01 (Big-Crush) Statistical Test Suite

TestU01, particularly the Big-Crush suite, is a comprehensive set of statistical tests designed to evaluate the randomness and quality of random number generators. It includes a wide range of tests covering different aspects of randomness, such as uniformity, independence, and unpredictability [8].

Flow Chart Approach: The methodology utilizing the TestU01 suite generally involves the following steps:

1. ***Random Number Generation:*** *RNG sequences are generated using selected algorithms.*
2. ***Test Suite Application:*** *The sequences are subjected to various statistical tests within the TestU01 suite, covering different aspects of randomness.*
3. ***Evaluation:*** *The results obtained from the tests are evaluated to determine the suitability and quality of the RNGs*

3. Experimental Setup

The experimental setup involved comprehensive testing of RNGs using three distinct methodologies: NIST, TestU01, and data compression tests. The RNGs were evaluated across varying input sizes to ensure a thorough assessment of their performance. For the data compression tests, a range of compressors was employed, including Gzip [15], LZMA2[16], Brotli [18], Zstd [19], and LZFSE [20] These compressors were utilized to analyze the randomness of the generated sequences, with each RNG subjected to compression using different algorithms. Subsequently, the results obtained from the data compression tests, NIST tests, and L'Ecuyer-TESTU01 (BIG-CRUSH) tests were collected for analysis. This analysis aimed to evaluate the performance of each RNG under various testing conditions, including data compression, NIST, and L'Ecuyer-TESTU01 (BIGCRUSH) tests, providing insights into their effectiveness and reliability for generating random sequences.

3.1. Characteristics of Random number generators used in the experiment

The experiment involved the use of various Random number generators (RNGs), each with distinct characteristics. The Linear Congruential Generator [22], RANDU [23], LSFR [24], Middle Square Generator [25], Mersenne Twister data generator [26], PCG32[27], and Blum Blum Shub (B.B.S) [28] were employed. These RNGs varied in their methods and properties, ranging from simple equations to sophisticated algorithms. While some RNGs offered computational efficiency, others prioritized randomness or cryptographic security [17].

3.2. Characteristics of data compressors used in the experiment

It should be noted that data compression methods have been recognized in the literature as a foundation for randomness testing. For instance, the Lempel-Ziv Compression Test, and the Approximate Entropy Test are associated with universal codes, as discussed in [9]. Unlike traditional

methods, our approach offers the opportunity to conduct a randomness test based on any lossless data compression method, even when the distribution law of the code word lengths is unknown.

In our investigation of the randomness of Random number generators (RNGs), we utilized the following modern data compressors: Gzip [15], LZMA2[16], Brotli [18], Zstd [19], and LZFSE [20]. The detailed characteristics of each compressor are shown below.

Table 1

Characteristics of Data compressor

Data Compressors	Compression Ratio	Compression speed	Decompression Speed	Memory Usage	References
Gzip	Moderate	Fast	Fast	Low	[15]
LZMA2	High	Moderate to Slow	Moderate to Fast	High	[16]
Brotli	Very High	Moderate	Moderate	Moderate	[18]
Zstd	Very High	Fast	Very Fast	Moderate	[19]
LZFSE	Medium	Very Fast	Very Fast	Low	[20]

3.2.1. Gzip

Gzip, developed by Jean-loup Gailly and Mark Adler in 1992, is one of the most popular and widely supported compression algorithms. It utilizes the DEFLATE algorithm, which combines LZ77 compression and Huffman coding. Gzip offers moderate compression ratios and relatively fast compression and decompression speeds, making it suitable for various applications, including web servers, file compression, and transmission over networks. It is a standard tool in Unix-like systems and is commonly used to compress individual files or collections of files into a single archive [15].

3.2.2. LZMA2

LZMA2 is a compression algorithm developed by Igor Pavlov for use in the 7-Zip archiver. It is based on the LZMA (Lempel-Ziv-Markov Chain-Algorithm) compression method, which combines LZ77-style sliding window compression with a range coder. LZMA2 offers high compression ratios and supports multi-threaded compression and decompression, enabling faster processing speeds on modern multi-core systems. It is often used in situations where maximum compression is desired, such as archiving files or distributing software packages. However, the trade-off for its high compression ratios is increased computational overhead during compression and decompression [16].

3.2.3. Brotli

Developed by Google and released in 2015, Brotli is recognized for its exceptional compression ratios compared to many other algorithms. It was created to improve upon existing compression techniques by employing modern compression techniques like context modeling, Huffman coding, and a pre-defined dictionary. This approach allows Brotli to achieve compression ratios typically 20%-30% better than those of similar algorithms like Gzip and Deflate. Brotli also offers fast compression and decompression speeds, making it a popular choice for web servers and browsers to compress HTTP responses, reducing bandwidth usage and improving website loading times [18].

3.2.4. Zstd

Developed by Facebook and released in 2016, Zstandard (Zstd) is a compression algorithm designed for high performance and efficiency across a wide range of compression levels. It was created to strike a balance between compression ratio and speed, making it adaptable for various applications. Zstd can achieve compression ratios comparable to those of algorithms like Brotli while offering significantly faster compression and decompression speeds [19].

3.2.5. LZFSE

Developed by Apple and released in 2015, LZFSE (Lempel-Ziv Finite State Entropy) is optimized for use on iOS and macOS platforms. It was created by combining the Lempel-Ziv compression algorithm with finite state entropy coding to achieve high compression ratios while maintaining low memory and CPU usage. LZFSE was designed with efficiency in mind, particularly for devices with limited computational resources, such as smartphones and tablets. Its fast compression and decompression speeds, along with adaptive compression capabilities, make it an efficient choice for various data compression tasks [20].

4.0. Experimental Results

4.1. Linear Congruential Generator (LCG)

Table 2

LCG Data compression test result

Compressor	10^3	10^4	10^5	$5*10^5$	10^6
Gzip	100	100	100	100	100
LZMA	100	100	100	100	100
Brotli	100	100	100	100	100
Zstd	100	100	100	100	100
LZFSE	100	100	100	100	100

Table 3

LCG: NIST statically test results

Test name	10^3	10^4	10^5	$5*10^5$	10^6
Frequency (Mono bit) Test	0	0	0	0	0
Frequency Test within a Block	0	0	0	0	0
Runs Test	0	0	0	0	0
Test for the Longest Run of Ones in a Block	0	0	0	0	0
Binary Matrix Rank Test	0	0	0	0	0
Discrete Fourier Transform (Spectral) Test	100	100	100	100	100
Non-overlapping Template Matching Test	0	0	0	0	0
Overlapping Template Matching Test	0	0	0	0	0
Maurer's 'Universal Statistical' Test	0	0	0	0	0
Linear Complexity Test	0	0	0	0	0
Serial Test	0	0	0	0	0
Approximate Entropy Test	0	0	0	0	0
Cumulative Sums Test (Forward)	100	100	100	100	100
Cumulative Sums Test (Backward)	100	100	100	100	100
Random Excursions Test	0	0	0	0	0
Random Excursions Variant Test	0	0	0	0	0

Table 4

LCG L'Ecuyer-TESTU01(BIG-CRUSH)

Test Name	10^3	10^4	10^5	$5*10^5$	10^6
Birthday Spacings Test	0	0	0	0	0
Overlapping Permutations Test	0	0	0	0	0
Collision Test	100	100	100	100	100
Close Pairs Test	0	0	0	0	0
Coupon Collector Test	100	100	100	100	100
Gap Test	100	100	100	100	100
Matrix Rank Test	0	0	0	0	0
Linear Complexity Test	100	100	100	100	100
Hamming Weight Test	0	0	0	0	0
Hamming Correlation Test	0	0	0	0	0
Hamming Independence Test	100	100	100	100	100
String Test	100	100	100	100	100
Overlapping Template Matching Test	100	100	100	100	100
Quadrat Test	100	100	100	100	100
Permutation Test	0	0	0	0	0
Run Test	0	0	0	0	0
Poker Test	100	100	100	100	100
Poker Test (Poker 2)	0	0	0	0	0
Gap Test (Gap 16K)	100	100	100	100	100
Gap Test (Gap)	0	0	0	0	0
Weighted Digits Test	100	100	100	100	100
Weight Distribution Test	100	100	100	100	100
Random Spheres Test	0	0	0	0	0
Squeeze Test	0	0	0	0	0
Fourier Transform Test	0	0	0	0	0
Discrete Fourier Transform Test	0	0	0	0	0

4.2. RANDU RNG

Table 5

RANDU Data compression test result

Compressor	10^3	10^4	10^5	$5*10^5$	10^6
Gzip	0	100	100	100	100
LZMA	0	55	100	100	100
Brotli	0	100	100	100	100
Zstd	0	0	1	100	100
LZFSE	0	0	100	100	100

Table 6

RANDU NIST test suit results

Test name	10^3	10^4	10^5	$5*10^5$	10^6
Frequency (Monobit) Test	0	0	0	0	0
Frequency Test within a Block	0	0	0	0	0
Runs Test	0	0	0	0	0
Test for the Longest Run of Ones in a Block	0	0	0	0	0
Binary Matrix Rank Test	0	0	0	0	0
Discrete Fourier Transform (Spectral) Test	100	100	100	100	100
Non-overlapping Template Matching Test	0	0	0	0	0
Overlapping Template Matching Test	0	0	0	0	0
Maurer's 'Universal Statistical' Test	0	0	0	0	0
Linear Complexity Test	0	0	0	0	0
Serial Test	0	0	0	0	0
Approximate Entropy Test	0	0	0	0	0
Cumulative Sums Test (Forward)	100	100	100	100	100
Cumulative Sums Test (Backward)	100	100	100	100	100
Random Excursions Test	0	0	0	0	0
Random Excursions Variant Test	0	0	0	0	0

Table 7

RANDU L'Ecuyer-TESTU01(BIG-CRUSH)

Test Name	10^3	10^4	10^5	$5*10^5$	10^6
Birthday Spacings Test	0	0	0	0	0
Overlapping Permutations Test	0	0	0	0	0
Collision Test	100	100	100	100	100
Close Pairs Test	0	0	0	0	0
Coupon Collector Test	100	100	100	100	100
Gap Test	100	100	100	100	100
Matrix Rank Test	0	0	0	0	0
Linear Complexity Test	100	100	100	100	100
Hamming Weight Test	0	0	0	0	0
Hamming Correlation Test	0	0	0	0	0
Hamming Independence Test	100	100	100	100	100
String Test	100	100	100	100	100
Overlapping Template Matching Test	100	100	100	100	100
Quadrat Test	100	100	100	100	100
Permutation Test	0	0	0	0	0
Run Test	0	0	0	0	0
Poker Test	100	100	100	100	100
Poker Test (Poker 2)	0	0	0	0	0
Gap Test (Gap 16K)	100	100	100	100	100
Gap Test (Gap)	0	0	0	0	0
Weighted Digits Test	100	100	100	100	100
Weight Distribution Test	100	100	100	100	100
Random Spheres Test	0	0	0	0	0
Squeeze Test	0	0	0	0	0
Fourier Transform Test	0	0	0	0	0
Discrete Fourier Transform Test	0	0	0	0	0

4.3. Mersenne Twister data generator (MTRNG)

Table 8

MTRNG -Data compression test

Compressor	10^3	10^4	10^5	$5*10^5$	10^6
Gzip	0	0	0	0	0
LZMA2	0	0	0	0	0
Brotli	0	0	0	0	0
Zstd	0	0	0	0	0
LZFSE	0	0	0	0	0

Table 9

MTRNG NIST test suit results

Test name	10^3	10^4	10^5	$5*10^5$	10^6
Frequency (Mono bit) Test	0	0	0	0	0
Frequency Test within a Block	0	0	0	0	0
Runs Test	0	0	0	0	0
Test for the Longest Run of Ones in a Block	0	0	0	0	0
Binary Matrix Rank Test	0	0	0	0	0
Discrete Fourier Transform (Spectral) Test	100	100	100	100	100
Non-overlapping Template Matching Test	0	0	0	0	0
Overlapping Template Matching Test	0	0	0	0	0
Maurer's 'Universal Statistical' Test	0	0	0	0	0
Linear Complexity Test	0	0	0	0	0
Serial Test	0	0	0	0	0
Approximate Entropy Test	0	0	0	0	0
Cumulative Sums Test (Forward)	100	100	100	100	100
Cumulative Sums Test (Backward)	100	100	100	100	100
Random Excursions Test	0	0	0	0	0
Random Excursions Variant Test	0	0	0	0	0

Table 10

MTRNG L'Ecuyer-TESTU01(BIG-CRUSH)

Test Name	10^3	10^4	10^5	$5*10^5$	10^6
Birthday Spacings Test	0	0	0	0	0
Overlapping Permutations Test	0	0	0	0	0
Collision Test	0	0	2	2	2
Close Pairs Test	0	0	0	0	0
Coupon Collector Test	0	0	2	2	2
Gap Test	100	100	100	100	100
Matrix Rank Test	0	0	0	0	0
Linear Complexity Test	100	100	100	100	100
Hamming Weight Test	1	1	2	2	2
Hamming Correlation Test	0	0	0	0	0
Hamming Independence Test	1	1	3	3	3
String Test	1	1	2	2	2

Overlapping Template Matching Test	100	100	100	100	100
Quadrat Test	100	100	100	100	100
Permutation Test	1	1	2	2	2
Run Test	1	1	2	2	2
Poker Test	1	1	1	1	1
Poker Test (Poker 2)	0	0	1	1	1
Gap Test (Gap 16K)	100	100	100	100	100
Gap Test (Gap)	0	0	0	0	0
Weighted Digits Test	100	100	100	100	100
Weight Distribution Test	0	0	0	0	0
Random Spheres Test	0	0	0	0	0
Squeeze Test	0	0	0	0	0
Fourier Transform Test	0	0	0	0	0
Discrete Fourier Transform Test	0	0	0	0	0

4.4. PCG-32

Table 11

PCG Data compression test results

Compressor	10^3	10^4	10^5	$5*10^5$	10^6
Gzip	0	0	0	0	0
LZMA	0	0	0	0	0
Brotli	0	0	0	0	0
Zstd	0	0	0	0	0
LZFSE	0	0	0	0	0

Table 12

PCG NIST test suit result

Test name	10^3	10^4	10^5	$5*10^5$	10^6
Frequency (Mono bit) Test	0	0	0	0	0
Frequency Test within a Block	0	0	0	0	0
Runs Test	0	0	0	0	0
Test for the Longest Run of Ones in a Block	0	0	0	0	0
Binary Matrix Rank Test	0	0	0	0	0
Discrete Fourier Transform (Spectral) Test	100	100	100	100	100
Non-overlapping Template Matching Test	0	0	0	0	0
Overlapping Template Matching Test	0	0	0	0	0
Maurer’s ‘Universal Statistical’ Test	0	0	0	0	0
Linear Complexity Test	0	0	0	0	0
Serial Test	0	0	0	0	0
Approximate Entropy Test	0	0	0	0	0
Cumulative Sums Test (Forward)	0	0	0	0	0
Cumulative Sums Test (Backward)	0	0	0	0	0
Random Excursions Test	0	0	0	0	0
Random Excursions Variant Test	0	0	0	0	0

Table 13

PCG L'Ecuyer-TESTU01(BIG-CRUSH) result

Test Name	10^3	10^4	10^5	$5*10^6$	10^6
Birthday Spacing's Test	0	0	0	0	0
Overlapping Permutations Test	0	0	0	0	0
Collision Test	0	0	2	2	2
Close Pairs Test	0	0	0	0	0
Coupon Collector Test	0	0	2	2	2
Gap Test	0	0	0	0	0
Matrix Rank Test	0	0	0	0	0
Linear Complexity Test	0	0	0	0	0
Hamming Weight Test	2	2	0	0	0
Hamming Correlation Test	0	0	0	0	0
Hamming Independence Test	5	5	2	2	2
String Test	0	0	0	0	0
Overlapping Template Matching Test	0	0	0	0	0
Quadrat Test	0	0	0	0	0
Permutation Test	0	0	0	0	0
Run Test	0	0	0	0	0
Poker Test	2	2	2	2	2
Poker Test (Poker 2)	1	1	0	0	0
Gap Test (Gap 16K)	0	0	0	0	0
Gap Test (Gap)	0	0	0	0	0
Weighted Digits Test	0	0	0	0	0
Weight Distribution Test	0	0	0	0	0
Random Spheres Test	0	0	0	0	0
Squeeze Test	0	0	0	0	0
Fourier Transform Test	0	0	0	0	0
Discrete Fourier Transform Test	0	0	0	0	0

5.0. Discussion and Interpretation of the experimental results

We investigated the performance of four weak Random number generators (LCG, RANDU, LSFR, Middle Square Generator) and three strong random number generators (MT, PCG32 and Blum Blum shub) The evaluation was conducted using data compression tests, NIST, and L'Ecuyer-TESTU01 (BIG-CRUSH) methodologies across varying file sizes of 10KB, 100KB,500KB and 1MB with each RNG subjected to 100 sequence repetitions.

The results of the experiment revealed that the weak generators, including LCG, RANDU RNG, LSFR, and Middle Square Generator, exhibited a 100% rejection rate in the data compression test, indicating poor randomness. Additionally, mixed results were observed in the NIST Test Suite, with significant rejections across various tests, highlighting weaknesses in the randomness of generated sequences. Similarly, the TestU01 (Big-Crush) analysis demonstrated significant rejections in multiple statistical tests, further indicating inadequacies in the randomness of the sequences generated by weak RNGs.

On the other hand, the strong generators, including MTRNG, PCG32, B.BS showed no significant rejection rates in the data compression test, suggesting better randomness compared to weak generators. Moreover, these strong generators exhibited strong performance in the NIST Test Suite, with a high passing rate across most tests, indicating robust randomness. Similarly, minimal rejections were observed in the TestU01 (Big-Crush) statistical tests for strong generators, further confirming their better performance and robust randomness compared to weak generators.

6.0. Conclusion

In conclusion, the effectiveness of the three tests applied to random number generators reveals important insights into their reliability and randomness. The data compression test, NIST Test Suite, and L'Ecuyer-TESTU01 (BIG-CRUSH) analysis serve as valuable tools for assessing the quality of RNG sequences.

The results indicate that data compressor test batteries such as Gzip, LZMA2, Brotli, Zstd, and LZFSE effectively detect non-randomness in RNGs, while NIST and TestU01 do not.

The research results show that a test based on Modern data compressors has the potential to be as effective as other well-known test suites containing multiple tests.

These findings emphasize the importance of using multiple testing methodologies to assess RNG sequence randomness comprehensively and stress the potential of modern data compression tests as effective test suites.

References

[1] **Ryabko, B. Y.; Monarev, V. A.** Using information theory approach to randomness testing: Journal of Statistical Planning and Inference. 2005, vol. 133, no. 1, pp. 95–110.

[2] **Ryabko, B.** Asymptotically most powerful tests for random number generators. J Stat. Planning Infer.2022, 217, 1–7.

[3] **L'ECUYER, P.** History of uniform random number generation. In 2017 Winter Simulation Conference. 2017, pp. 202–230.

[4] **Mezenes, A.; Van Oorschot, P.; Vanstone, S.** Handbook of Applied Cryptography. CRC Press, Boca Raton, FL. 1996.

[5] **Matsumoto, M.; Takuji, Nishimura.** Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Trans. Model. Comput. Simul. 8. 1998, 3-30.

[6] **Rukhin, L.** Testing randomness: a suite of statistical proce-dures. Theory of Probability & Its Applications. 2001, vol. 45, no. 1, pp.111–132.

[7] **Rukhin, A.; Soto, J.; Nechvatal, J.; Smid, M.; Barker, E.; Leigh, S.; Levenson, M.; Vangel, M.; Banks, D.; Heckert, A.; Dray, J.;Vo,s.** A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. National Institute of Standards and Technology, 2010.

[8] **L'Ecuyer, P.; Simard, R.** TestU01: A C library for empirical testing of random number generators. ACM Trans. Math. Softw. (TOMS) 2007, 33, 22.

[9] **Ryabko, B., Astola, J.; Malyutov, M.** Compression-Based Methods of Statistical Analysis and Prediction of Time Series. Springer International Publishing Switzerland. 2016.

[10] **Marsaglia, G.** The marsaglia random number CDROMincluding the diehard battery of tests of randomness.1995.

[11] **Caelli, W.** Crypt x package documentation. Tech. Rep.,Infor-mation Security Research. 1992.

[12] **Katos, V.** A randomness test for block ciphers. Applied Mathe-matics and Computation. 2005, vol. 162, no. 1, pp. 29–35.

[13] **Alcover, P. M., Guillam´, A., Ruiz , M. D.** A new randomness test for bit sequences. Informatics. 2013, vol. 24, no. 3, pp. 339–356.
[14] **Knuth, Donald. E.** The Art of Computer Programming. Semi numerical Algorithms. 1997, Volume 2, Addison-Wesley.
[15] **Abdelfattah, M. S., Hagiescu, A., Singh, D.** Gzip on a chip: High performance lossless data compression on fpgas using opencl. In *Proceedings of the international workshop on openCL*. 2013 & 2014, pp. 1–9.
[16] **Onuma, Y., Terashima, Y., Kiyohara, R.** Compression method for ECU software updates. IEEE Tenth International Conference on Mobile Computing and Ubiquitous Network (ICMU). 2017 Oct 3, pp. 1-6.
[17] **Andrea, Rock.** Pseudorandom Number Generators for Cryptographic Applications. Universitat Salzburg, Marz. 2005.
[18] **Alakuijala, J., Farruggia, A., Ferragina, P., Kliuchnikov, E., Obryk, R., Szabadka, Z., Vandevenne, L.** Brotli: *A general-purpose data compressor.* ACM Transactions on Information Systems (TOIS).2018, 37, 1–30.
[19] **Collet, Y., Kucherawy, M.** *Zstandard Compression and the application/zstd Media Type*. 2018, No. rfc8478.
[20] **Li, Mingyin., Yue, Liu., Na, Wang.** A Novel ANS Coding with Low Computational Complexity*IEEE/CIC International Conference on Communications in China (ICCC Workshops)*. IEEE, 2023.
[21] **Ryabko, B. Ya., Stognienko, V. S., Shokin, Yu. I.** A new test for randomness and its application to some cryptographic problems. J. Statist. Plann. Inference 123, 365–376.
[22] **Fontaine, C.** LINEAR CONGRUENTIAL GENERATOR. van Tilborg, H.C.A. (eds) Encyclopedia of Cryptography and Security. Springer, Boston, MA.2005.
[23] **Markowsky, G.** The Sad History of Random Bits. *J. Cyber Secur. Mobil*.2014, 31, 1–24.
[24] **Klein, A.** Linear Feedback Shift Registers Stream Ciphers. Springer, London. 2013,17–58.
[25] **Kim, J., Chae, H.** All-Digital True Random-Number Generator using Middle Square Method, *IEEE Asian Solid-State Circuits Conference (A-SSCC)*, Taipei, Taiwan. 2022, pp. 1–3.
[26] **Matsumoto, M., Saito, M., Nishimura, T., Hatagi, M.** Cryptographic Mersenne Twister and Fubuki stream/block cipher. Cryptographic ePrint Archive, June 2005.
[27] **O'Neill,** Melissa E. PCG: A Family of Simple Fast Space-Efficient Statistically Good Algorithms for Random Number Generation. ACM Transactions on Mathematical Software. 2014
[28] **Blum, L., Blum, M., Shub, M.** A Simple Unpredictable Pseudo-Random Number Generator. SIAM Journal on Computing, Society for Industrial & Applied Mathematics (SIAM). 2021, 364–383.

Information about the Author

Yeshewas Getachew Lulu, PhD Student

Информация об авторе

Йешевас Гетачеу Лулу, аспирант

Статья поступила в редакцию 30.01.2025;
одобрена после рецензирования 04.02.2025; принята к публикации 04.02.2025

The article was submitted 30.01.2025;
approved after reviewing 04.02.2025; accepted for publication 04.02.2025