JOURNAL OF THEORETICAL PHYSICS & CRYPTOGRAPHY

www.IJTPC.org/Publications

Better Evaluation of PRNGs by Reducing Type II Error and Experiment on Merssene Twister

Ahmad Gaini[1], Meysam Tasallazadeh Khemes[2], Behbod Keshavarzi[3]

[1,3]Department of statistic, Faculty of Mathematic science, university of Imam hosein, Tehran, Iran.
[2] Department of Statistics, Faculty of Mathematic science, university of Tarbiat Modares, Tehran, Iran.

ABSTRACT

PRNGs(pseudo random number generator) play a crucial role in Cryptography and statistics and Mersenne twister is a useful pseudorandom number generator . The various statistical methods have introduced in the last twenty years and statistical tests are the main step in determining the quality of PRNGs.NIST(2001) introduced the 16 statistical tests for evaluating pseudorandom random number generators that are applied in the cryptography field. One of the limitations NIST tests is about type II error and this limitation is highlighted in Mersenne twister PRNG. To construct GOOD PRNG by the NIST test suite is easy, though the output of the PRNG is distinguishable from the uniform distribution. On the other hand NIST has restrictions regarding to the central limit theorem and the law of iterated logarithm(LIL). Some of the NIST tests related to frequency tests in NIST tests accommodate the central limit theorem while no tests in NIST test suite cover the law of iterated logarithm. in this paper proposed a novel statistical test with a motivation to understand limitations of NIST tests then have produced the set of zero-one bit sequences in the binary format by one of the most famous PRNGs as named mersenne twister then transformed the output of this PRNG for the length of the zero numbers less than the length of one numbers necessarily and in the final carry out a simulation study and has been compared performance of NIST and LIL tests.

Keywords
Mersenne Twister, LIL test, Randomness,Type II eror, NIST test.

1. INTRODUCTION

PRNGs (Pseudo random number generators) are the main part of cryptographic applications where in the last twenty years widely studied. the idea behind a PRNG is to generate a sequence of bits $(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n)$ using by recursive relationship of the form :

$$\varepsilon_i = f(\varepsilon_{i-1}, \varepsilon_{i-2}, \ldots, \varepsilon_{i-n})$$

Where n initial numbers are needed to start the recurrence. many models have been proposed to generate random numbers but a few of them are useful for cryptography.thus it is important to have efficient random number generators that have huge capabilities in cryptography .Gayakwad and Ranbir[2012] proposed a PRNG based on chaos maps for image processing. note that evaluating the high quality PRNGs is a key step in achieving secure cryptographic systems . for this purpose many Statistical hypothesis testing have been proposed to acquire high quality generators that produce output zero-one bit strings randomly. Marsaglia[1996] introduced the battery of tests as named DIEHARD and also NIST SP800-22 [2010] has introduced the battery of statistical tests for determining randomness in bit string sequence. patidar et al[2009]generated data based on formed chaos standard mappings and run NIST and DIEHARD test on it. Also Stoyanov[2014] introduced pseudorandom bit generation based on chebyshev polynomial and tinkerbell map, stoyanov and kordov [2014] proposed zaslavsky map based PRNG and are evaluated these PRNGs with NIST and DIEHARD packages .one of the most limitation of NIST statistical tests suite is in related to Type II error. for example if output bit string was biased (e.g. sequence of bits consist mainly of 1's), NIST test show that this sequence is Good Pseudorandom and so the output of pseudorandom generator will considered follow as uniform distribution. wang[2002] introduced two approaches (ontological and behavioristic) for definition concept of randomness and described how to evaluate quality of PRNG with the law of iterated logarithm. Wang and Nicol [2015] generated data by Standard C LCG, MT19937, PHP LCG, PHP MT19937, flawed Debian open SSL, standard open SSL and described how these data have tested with LIL,then addressed the weak points of NIST test such that indicated nonrandom PRNG as random. This paper is outlined as follows: in section 2 NIST tests are introduced and in section 3 imply requirement notations for LIL tests. section 3 discusses the LIL test, section 4 represent results of a simulation study for $n=2^{22},2^{23},2^{24},2^{25},2^{26}$ at the sample size 100 that carried out to examine the performance of the NIST and LIL tests.

2 Description of NIST Tests

Suppose that $\varepsilon=\varepsilon_1,\ldots,\varepsilon_n$ be the sequence of bits as generated by the Pseudorandom number generator being tested. The national institute standard and technology (NIST) introduced the fifteen statistical tests for evaluating a PRNG in cryptographic applications. these fifteen tests can be grouped into four categories:

1)frequency tests include of:

The frequency (monobit) test:

In this test it is intended to make sure that frequencies of 1s and 0s equally distributed across the entire sequence of binary bits.

Frequency test within a block:

The focus of this test is to ensure that frequencies of 0 and 1 are equally distributed across the zero-one bit string.

The runs test:

Through this test it is intended to test the frequencies of run of 0s and 1s of several length are in limit of randomness. In this test the runs of length k means exactly k identical bits bounded by bits of opposite value.

Tests for the longest run of ones in a block:

For this test the purpose is to see if the frequencies of longest run of 1s appearing in the tested bit string are consistent with that expected for a random sequence and for execute this test should the n-bit string is divided in N non overlapping blocks each of L bit where $N=\left\lfloor\frac{n}{L}\right\rfloor$ and L should be taken as reasonably small. The additional bits are ignored.

2) recognize the repetitive patterns:

The binary matrix rank test:

By this test one examine the n bit zero one string has repetitive template across entire sequence.

The discrete Fourier transform test:

Through this test it is intended if the n bit string has periodic features across its entire n bit string.

3)tests for matched patterns:

The non overlapping template matching test:

In this test it is intended to see matched pattern in a nonoverlapping manner so that an m-bit window is considered to search for specific m-bit pattern if the tested sequence was random the central limit theorem is applicable.

The overlapping template matching test:

This test detect template matching in an overlapping manner and this test assumed the poisson asymptotic distribution has been satisfied.

Universal test:

The goal of universal test is to determine the number of bits between matching patterns. The purpose of the universal test is to detect whether or not the bit string can be significantly compressed without loss of knowledge of data. A significantly compressible bit string is considered to be non-random bit string.

The linear complexity test:

The focus of this test is to obtain the length of a linear feedback shift register. Note that The purpose in this is test is to determine whether or not the sequence is complex enough to regards as random. consider that an linear feedback shift register that is too short implies non-randomness.

The serial test:

Through this test it is designed for see the frequency of all possible overlapping m-bit patterns across the entire sequence. The purpose of this test is to determine whether the number of occurrences of the 2^l m-bit overlapping patterns is approximately the same as would be expected for a random sequence.. Note that for l = 1, the Serial test is equivalent to the Frequency test

The approximate entropy test:

For n-bit string the entropy that is a test of randomness based on repeating patterns is measured by comparing the frequency of overlapping patterns of all possible m-bit patterns with that of (m+1)-bit patterns. The comparison between entropies of m and (m+1)-bit patterns is termed as approximate entropy, ApEn(m), which is compared against the expected result of a random sequence.

4)Random walk based tests:

The cumulative sums test:

The purpose of this test is the maximal excursion (from zero) of the random walk defined by the cumulative sum of adjusted (-1, +1) digits in the binary sequence. The goal of the test is to determine whether the cumulative sum of the partial sequences occurring in the tested sequence is too large or too small relative to the expected behavior of that cumulative sum for random binary sequences. For a random sequence, the excursions of the random walk should be near zero. For certain types of non-random sequences, the excursions of this random walk from zero will be large.

The random excursions test:

This test looks whether 1s or 0s are occurring in large numbers at early stages or at later stages or 1s and 0s are intermixed evenly across the entire sequence.

The random excursions variant test:

The focus of this test is the total number of times that a particular state is visited (i.e., occurs) in a cumulative sum random walk. The purpose of this test is to detect deviations from the expected number of visits to various states in the random walk. This test is actually a series of eighteen tests (and conclusions), one test and conclusion for each of the states: -9, -8, …, -1 and +1, +2, …, +9.

To obtain p-value, NIST has adopted two procedure: in the first procedure the p-value is calculated by χ^2 data as test statistics and is computed as:

$$p-value = 1 - erf(x) \qquad x \geq 0 \qquad (1)$$

where $erf(x)$ reports to error function given below:

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_x^{\infty} e^{-t^2} dt \qquad x<\infty \qquad (2)$$

and in second procedure the p value is obtained with considering the K as the number of degrees of freedom data and χ^2 data as test statistics. Now for $a = \frac{k}{2}$ and $x = \frac{\chi^2}{2}$ the value of p-value is calculated as:

$$p-value = 1 - \frac{\Gamma(a,x)}{\Gamma(a,\infty)} \qquad (3)$$

Where $\Gamma(a,x)$ is gamma function and given below:

$$\Gamma(a,x)=\int_0^x t^{a-1}e^{-t}dt \qquad (4)$$

If $p-value<0.01$ then the test conclude that the sequence is non-random otherwise the sequence is random. PRNG is considered GOOD if with probability α the binary sequence $\varepsilon=\varepsilon_1,\ldots,\varepsilon_n$ generated by the PRNG fail the test. In the following, we describe that NIST test has limitations with straightforward Type II errors.

Let $\{g_n\}_{n\in N}$ be a mersenne twister where $g_n:\{0,1\}^n\to\{0,1\}^{l(n)}$ with $l(n)>n$ .then define $\{g'_n(x)\}$ such that this novel pseudorandom generator as follows:

$$g'_n(x)=\begin{cases} g_n(x) & \text{if } g_n(x) \text{ has more } 0's \text{ than } 1's \\ g_n(x)\oplus 1^{l(n)} & O.W. \end{cases} \qquad (5)$$

Then it is trivial to has proved that $\{g_n\}_{n\in N}$ is GOOD pseudorandom generator by NIST test in depth analysis of the p-value distributions.

Theorem 1[wang 2014]: Let $g_n:\{0,1\}^n\to\{0,1\}^{l(n)}$ and $\{g'_n\}_{n\in N}$ be defined by (5). Then for each $x\in\{0,1\}^n$, the p-values for $\{g_n(x)\}$ and $\{g'_n(x)\}$ are the same for all of the 15 NIST testing.

Proof. The theorem can be proved for all of the 15 NIST tests using the symmetric properties (when 0s and1s are flipped) of the corresponding probability distributions.

In the following, we prove the theorem for the monobit testing of NIST. For NIST monobit testing, the p-value on a binary string ε is defined by:

$$p-value(\varepsilon)=erf\left(\frac{\left|\sum_{i=0}^{|\varepsilon|-1}2\varepsilon[i]-|\varepsilon|\right|}{\sqrt{2|\varepsilon|}}\right) \qquad (6)$$

Thus it is straightforward that $p-value(g(x))=p-value(g'(x))$ for all $x\in\{0,1\}^n$ ..For survey details of NIST testing methods see [3] and [7].

feler[1945] described that two essential limit theorems of random binary strings are the central limit theorem and the law of the iterated logarithm. other limitation of NIST tests are in conjunction with the law of the iterated logarithm such that the several frequency tests in the NIST cover the central limit theorem however it does not support the law of the iterated logarithm so we will design LIL based statistical test for evaluate $g_n(x)$ and $g'_n(x)$. The iterated logarithm test is based on the law of the iterated logarithm where first is introduced in 1924 by kinchin[4] and can be better than NIST because that is a tool for decreasing the type II error.

3. requirement notations and description LIL test.

In the lil test procedure for testing randomness, it is intended to derive a measure of probability and then is obtained statistical distances for examining randomness a PRNG. Also we use N and R^+ to denote the set of whole numbers and non-negative real numbers respectively, $\psi=\{0,1\}$ and ψ^* is the set of binary strings, ψ^n and ψ^∞ denote the set of binary strings of length n and the set of infinite binary sequences. The length of string s is represented by $|s|$ where $s\in\psi^*\cup\psi^\infty$ is a binary string and $n\in N$, $s\,|\,n=s[0,\ldots,n-1]$ denotes the initial segment of s while $s[n]$ denotes the n^{th} bit of s whereas $s[n]$ is zero or one. For a set ϑ of infinite sequences $prob[\vartheta]$ represents the probability that $\gamma\in\vartheta$ where γ is chosen by a uniform random experiment. furthermore As a basic first definition of PRNG, will introduce the concept of indistinguishability of two ensembles.

Definition 1: let $\{X_n\}_{n\in N}$ and $\{Y_n\}_{n\in N}$ be two probability ensembles then we say that X and Y are statistically indistinguishable if have been for each feasible algorithm Z and each polynomial p and all sufficiently large n the following inequality hold:

$$\left|P[Z(X_n)=1]-P[Z(Y_n)=1]\right|\le\frac{1}{p(n)} \qquad (7)$$

Definition 2:suppose that $f':N\to N$ with $f'(n)\ge n, n\in N$ and $\{U_n\}_{n\in N}$ be the uniform distribution then a pseudorandom generator is a polynomial time algorithm Ω if the two conditions has been satisfied:

1. $|\Omega(s)|=f(|s|)$ for all ψ^* where s is called seed

2. Ensemble $\{\Omega(U_n)\}_{n\in N}$ and $\{U_n\}_{n\in N}$ are computationally indistinguishable

Definition 3:(Goldreich [2004]): ensemble $\{X_n\}_{n\in N}$ is called unpredictable in polynomial-time, if for each probabilistic polynomial-time algorithm Ω and each polynomial p :

$$P[\Omega(X_n) = next_{\Omega}(X_n)] < \frac{1}{2} + \frac{1}{p(n)} \qquad (8)$$

In the following define p-random (polynomial time random) sequence with using concept of martingale

Definition4:(Ville [1939])

A martingale is a function $F:\Sigma^* \to R^+$ such that:

$$F(x) = \frac{F(x_1)+F(x_0)}{2} \qquad \forall x \in \Sigma^*$$

a martingale F succeeds on a sequence $\gamma \in \vartheta$ if $\limsup_{n\to\infty} F(\gamma[0..n-1]) = \infty$

Definition5: (Schnorr [1971]) an infinite sequence $\gamma \in \vartheta$ is p-random if for any polynomial time computable martingale F, F does not succeed on γ .

Also For a nonempty sequence of zero-ones $s \in \psi^*$ let:

$$\begin{cases} \theta(s) = \sum_{i=0}^{|s|-1} s[i] \\ \theta^*(s) = \dfrac{2\theta(s)-|s|}{\sqrt{|s|}} \end{cases} \qquad (9)$$

In the equation (9) $\theta(s)$ is the number of 1s in s sequence and $\theta^*(s)$ denotes the reduced number of bits 1 in s . in other words the law of the iterated logarithm (LIL) describes the fluctuation scales of a random walk furthermore According to law of large numbers for a pseudorandom sequence γ , the limit of $\dfrac{\theta(\gamma \mid n)}{n}$ is $\dfrac{1}{2}$.

Note that there is no complete algorithm to generate p-random sequences thus pseudorandom generators are commonly used to produce long bit strings for implemention in cryptographic programs.

The following theorem describe that how pseudorandom sequences should satisfy the LIL.

Theorem2 (wang [2000]) : suppose that $\gamma \in \vartheta$ then:

$$\theta_{lil}(\gamma \mid n) = \frac{2\sum_{i=0}^{n-1}\gamma[i]-n}{\sqrt{2n\ln n}} \qquad (10)$$

Then for each p random sequence $\gamma \in \vartheta$ we have:

$$\limsup_{n\to\infty} \theta_{lil}(\gamma \mid n) = 1 \quad \text{and} \quad \liminf_{n\to\infty} \theta_{lil}(\gamma \mid n) = -1$$

To obtain the normal approximation for $\theta_{lil}(\cdot)$ DEMoiver-Laplace theorem will be used to approximate the binomial distribution.

(9)

In the normal approximation of binomial distribution the number of success in n independent unbiased coin flips with head(or tail) probability is 0.5 as approximately a normal distribution with mean μ and variance $\frac{n}{4}$.

Theorem 3 (feler [1945]):for fixed k_1,k_2 we have:

$$\lim_{x\to\infty} prob[k_1 \le \theta^*(\xi \mid n) \le k_2] = \Phi(k_2)-\Phi(k_1) \qquad (11)$$

Where

$\Phi(k) = \int_{-\infty}^{k} f(y)dy$ and $f(y)$ is the cumulative distribution and the standard normal density function respectively.

Lemma1:

For any k_1,k_2 ,we have

$$prob[k_1 < \theta_{lil}(\xi \mid n) < k_2] = prob[k_1\sqrt{2\ln\ln n} < \theta_{lil}^*(\xi \mid n) < k_2\sqrt{2\ln\ln n}] \qquad (12)$$

To introduce the lil test for random sequence let $\alpha \in (0,.25]$ and $\Lambda \subset N$ we say that a sequence γ does not pass the weak $(\alpha,\Lambda)-LIL$ test if $-1+\alpha < \theta_{lil}(\gamma \mid n) < 1-\alpha$ for all $n \in \Lambda$

For executing the law of iterated logarithm consider the following steps:

- Select sequences of length $\Lambda_0 = \{2^0 n_1\}, \Lambda_1 = \{2^1 n_1\}, \Lambda_2 = \{2^2 n_1\}, \Lambda_3 = \{2^3 n_1\}, \Lambda_4 = \{2^4 n_1\}$, that $n_1 = 2^{22}$.
- Simulate $m = 100$ sequence and put them in R set.
- Derive the variation distribution, Hellinger distance and root mean square deviation (RMSD)

$$\begin{cases} \text{Variation distance:} & d(\pi_n^R, \pi_n^U) = \sup_{A \in \beta} \left| \pi_n^R(A) - \pi_n^U(A) \right| \\ \text{Hellinger Distance:} & H(\pi_n^R \parallel \pi_n^U) = \frac{1}{\sqrt{2}} \sqrt{\sum_{A \in \beta} \left(\sqrt{\pi_n^R(A)} - \sqrt{\pi_n^U(A)} \right)^2} \\ \text{RMSD:} & RMSD(\pi_n^R, \pi_n^U) = \sqrt{\frac{\sum_{A \in \beta} \left(\pi_n^R(A) - \pi_n^U(A) \right)^2}{42}} \end{cases} \quad (13)$$

For simplicity β is a partition of the real line R such that

$$\beta = \{(-\infty,1),[1,\infty)\} \cup \{[0.05x-1, 0.05x-0.95) : 0 \le x \le 39\}$$

and

$$\pi_n^R(\omega) = prob[\theta_{lil}(s) \in \omega, s \in \upsilon] \qquad \upsilon \subset \psi^n$$

Where ω is a Lebesgue measurable set on real line R for $U = \psi^n$ we use π_n^U to denote the corresponding probability measure induced by the uniform distribution and is equal to $\pi_n^U((-\infty, x]) = \Phi(x\sqrt{2\ln\ln n})$,If these distances were negligible accept the generator as randomness otherwise test rejected

4 Simulation study

Mersenne twister is a type of PRNGs that first developed by matsumoto and nishimura and available on the web site of the authors and is a version of a generalized feedback shift register PRNG.

This PRNG is based on the Mersenne prime $2^{19937} - 1$ and has a long period of $2^{19937} - 1$. The algorithm operates on a seed value of the length of 19937 bits such that there are 31 bits that is unused. this PRNG has the following advantages[3]:

1.its generation speed is very fast.

2. it has a large period of 2 19937

3. it has high dimensional equidistribution property such that 623 dimensionally equidistributed.

In this paper first has examined the quality of a Mersenne twister and transformed mersenne twister generated in (5) by NIST test with significant level $\alpha = .01$. In the simulation study produced the 100 sequences of length 1000000 by using the Mersenne twister pseudorandom generator in R software. The outout of results are illustrated in tables 1 and 2 that illustrate mentioned PRNG as GOOD PRNG unexpectable. Then with using of LIL test evaluate the quality of this PRNG. The result of LIL test has illustrated in table3.

5. Results and discussion

Increasing the quality of statistical tests for evaluating randomness of output a PRNG is essential in the cryptographic studies. There are set of tests which have been examined in this study. In this paper we evaluated the quality of mersenne twister as a efficient PRNG in cryptography with NIST tests and statistical tests based on the law of the iterated logarithm(LIL). in the simulation study has been generated at sample size 100 that length of each sample was 1000000 then the mersenne twister and transformed mersenne twister PRNG passed incorrectly and it was predictable and its reason is respected to type II error. Also we addressed efficiency LIL tests in the evaluation of the transformed PRNG that rejected by statistical test based on the law of iterated logarithm. For following studies preferred to test the lil test over other distribution functions to obtain the statistical distances.

6. REFERENCES

[1] Gayakwad, R. and P. Ranbir., 2012. A pseudorandom number generator using chaotic image processing. J. Theoretical Phys. Cryptogr., 1: 6-12

[2] Goldreich, O., 2004, Foundations of Cryptography. Cambridge University Press, New York Pareek N. K., Patidar V., Sud K. K., 2006. Image encryption using chaotic logistic map, Image and Vision Computing 24 926–934.

[3] Harase, S., 2014 , ArticleOn the F2-linear relations of Mersenne Twister pseudorandomnumber generators,mathematics and computers in simulation, 100, 103-113

[4] J K M Sadique Uz Zaman, Ranjan Ghosh, 2012 . Review on fifteen Statistical Tests proposed by NIST. Journal of theoretical physics and cryptography vol1

[5] Khinchin A. 1924., U ber einen satz der wahrscheinlichkeitsrechnung. Fund Math; 6:9e20.

[6] Marsaglia, G., 1996. DIEHARD: A battery of tests of randomness.http://www.stat.fsu.edu/pub/diehard/cdrom/linux/diehard.doc.

[7] Patidar, V. and K.K. Sud, 2009. A novel pseudo random bit generator based on chaotic standard map and its testing. Electron. J. Theor. Phys., 6: 327-344.

[8] Rukhin, A., J. Soto, J. Nechvatal, M. Smid and E. Barker et al., 2010. A statistical test suite for random and pseudorandom number generators for cryptographic applications. NIST Special Publication 800-22 Revision 1a, National Institute of Standards and Technology, pp: 1-131.

[9] Schnorr,C.P., Zufälligkeit und Wahrscheinlichkeit.1971, Lecture Notes inMath. 218. Springer Verlag

[10] Stoyanov, B. and K. Kordov, 2014, Novel zaslavsky map based pseudorandom bit generation scheme. Applied Math. Sci., 8: 8883-8887.

[11] Stoyanov,B.,2014, Pseudo-random Bit Generation Algorithm Based on Chebyshev Polynomial and Tinkerbell Map. Applied Math. Sci., 8(125): 6205-6210

[12] Ville,J., 1939,Étude Critique de la Notion de Collectif. Gauthiers-Villars, Paris.

[13] Wang,Y., 2002, A comparison of two approaches to pseudorandomness. Theoretical computer science, 276(1):449–459.

[14] Wang, Y. and T. Nicol, 2015, On statistical distance based testing of pseudo random sequences andexperiments with PHP and Debian OpenSSL. Comput. Secur., 53: 44-64.

Table1. NIST tests results and values of p values for mersenne twister generation

	$n = 2^{22}$		$n = 2^{23}$		$n = 2^{24}$		$n = 2^{25}$		$n = 2^{26}$	
Test name	p-value	result	p-value	Result	p-value	Result	p-value	Result	p-value	result
Frequency	.350	Accepted	.030	Accepted	.637	Accepted	.834	Accepted	.554	Accepted
Block frequency	.366	Accepted	.911	Accepted	.816	Accepted	.350	Accepted	.191	Accepted
Cumulative sums(1)	.508	Accepted	.501	Accepted	.350	Accepted	.571	Accepted	.469	Accepted
Runs	.798	Accepted	.437	Accepted	.319	Accepted	.897	Accepted	.851	Accepted
Longest run	.955	Accepted	.719	Accepted	.494	Accepted	.494	Accepted	.574	Accepted
Rank	.699	Accepted	.249	Accepted	.122	Accepted	.494	Accepted	.275	Accepted
FFT	.816	Accepted	.032	Accepted	.224	Accepted	.978	Accepted	.319	Accepted
Non overlapping	*.486*	Accepted	.506	Accepted	.534	Accepted	.492	Accepted	.507	Accepted
Overlapping	.262	Accepted	.262	Accepted	.030	Accepted	.137	Accepted	.062	Accepted
Universal	.304	Accepted	.851	Accepted	.816	Accepted	.678	Accepted	.798	Accepted
Approximate	.657	Accepted	.574	Accepted	.759	Accepted	.595	Accepted	.554	Accepted

entropy										
Random excursions	.347	Accepted	.274	Accepted	.419	Accepted	.808	Accepted	.530	Accepted
Random excursion variant	.449	Accepted	.322	Accepted	.413	Accepted	.408	Accepted	.421	Accepted
Serial (1)	.580	Accepted	.236	Accepted	.629	Accepted	.498	Accepted	.759	Accepted
Linear complexity	.494	Accepted	.883	Accepted	.678	Accepted	.275	Accepted	.236	Accepted

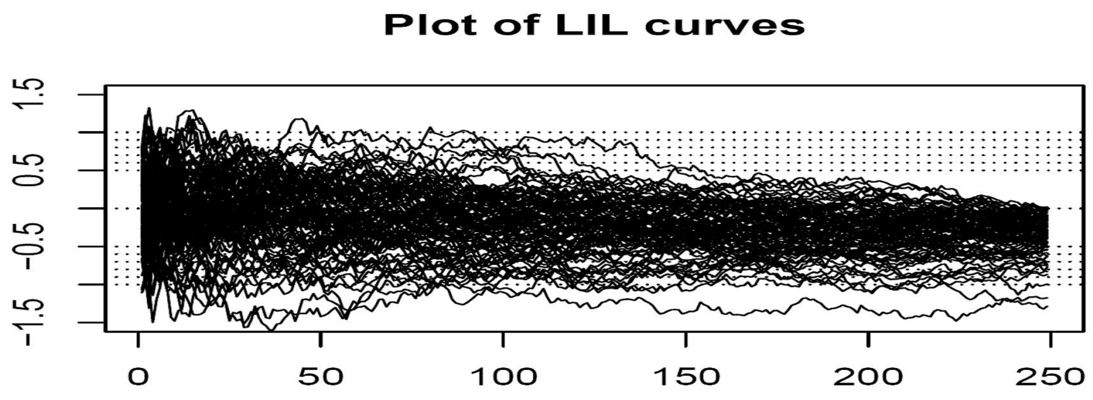
Table2. NIST tests results and values of p values for transformed mersenne twister generation

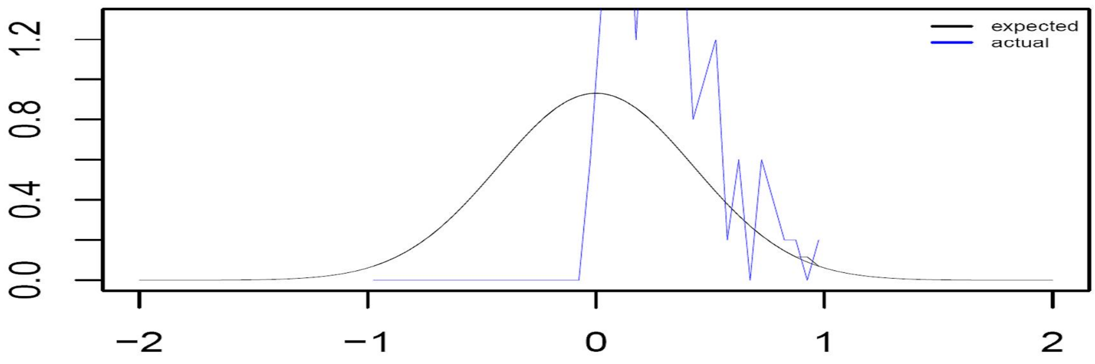
	$n = 2^{22}$		$n = 2^{23}$		$n = 2^{24}$		$n = 2^{25}$		$n = 2^{26}$	
Test name	p-value	Result	p-value	Result	p-value	result	p-value	Result	p-value	result
Frequency	.350	Accepted	.030	Accepted	.637	Accepted	.834	Accepted	.554	Accepted
Block frequency	.366	Accepted	.911	Accepted	.816	Accepted	.350	Accepted	.191	Accepted
Cumulative sums(1)	.508	Accepted	.501	Accepted	.350	Accepted	.571	Accepted	.469	Accepted
Runs	.798	Accepted	.437	Accepted	.319	Accepted	.897	Accepted	.851	Accepted
Longest run	.494	Accepted	.964	Accepted	.437	Accepted	.213	Accepted	.030	Accepted
Rank	.191	Accepted	.129	Accepted	.996	Accepted	.191	Accepted	.779	Accepted
FFT	.816	Accepted	.032	Accepted	.224	Accepted	.978	Accepted	.319	Accepted
Non overlapping	.453	Accepted	.494	Accepted	.464	Accepted	.479	Accepted	.513	Accepted
Overlapping	.055	Accepted	.834	Accepted	.236	Accepted	.191	Accepted	.191	Accepted
Universal	.304	Accepted	.851	Accepted	.816	Accepted	.678	Accepted	.798	Accepted
Approximate entropy	.657	Accepted	.574	Accepted	.759	Accepted	.595	Accepted	.554	Accepted
Random excursions	.577	Accepted	.292	Accepted	.419	Accepted	.572	Accepted	.367	Accepted
Random excursion variant	.427	Accepted	.231	Accepted	.569	Accepted	.364	Accepted	.370	Accepted
Serial (1)	.580	Accepted	.236	Accepted	.629	Accepted	.498	Accepted	.759	Accepted
Linear complexity	.366	Accepted	.897	Accepted	.319	Accepted	.181	Accepted	.419	Accepted

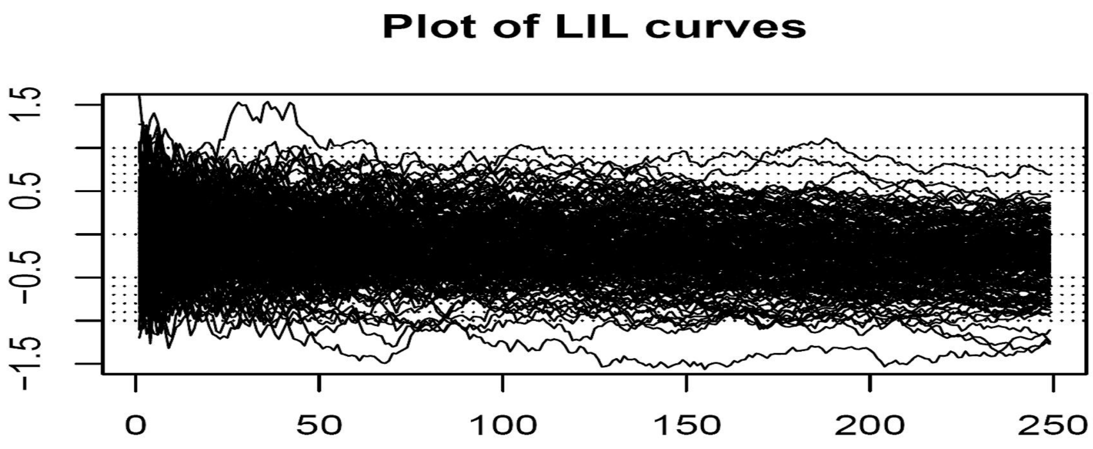
Table3. LIL tests results for transformed mersenne twister generation

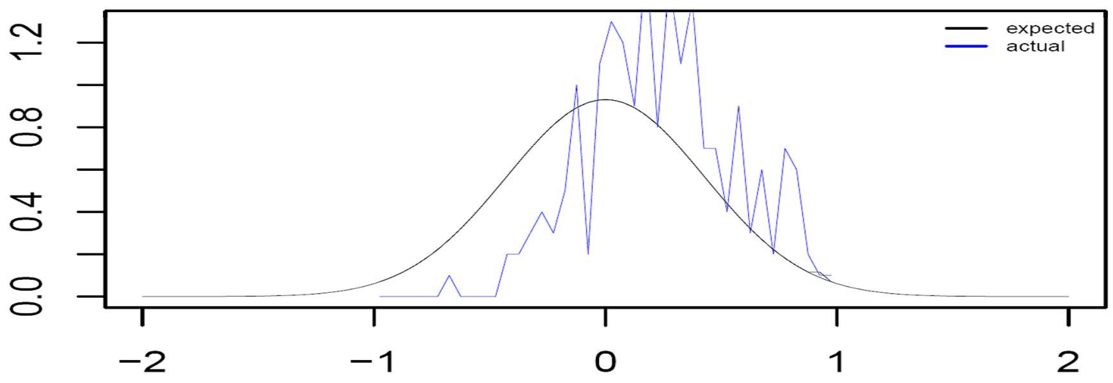
	$n=2^{22}$	$n=2^{23}$	$n=2^{24}$	$n=2^{25}$	$n=2^{26}$	Result
Total variation	0.496	.303	.214	.124	.094	Fail
Hellinger distance	0.537	.337	.214	.123	.103	Fail
RMSD	0.029	.018	.012	.007	.005	Fail

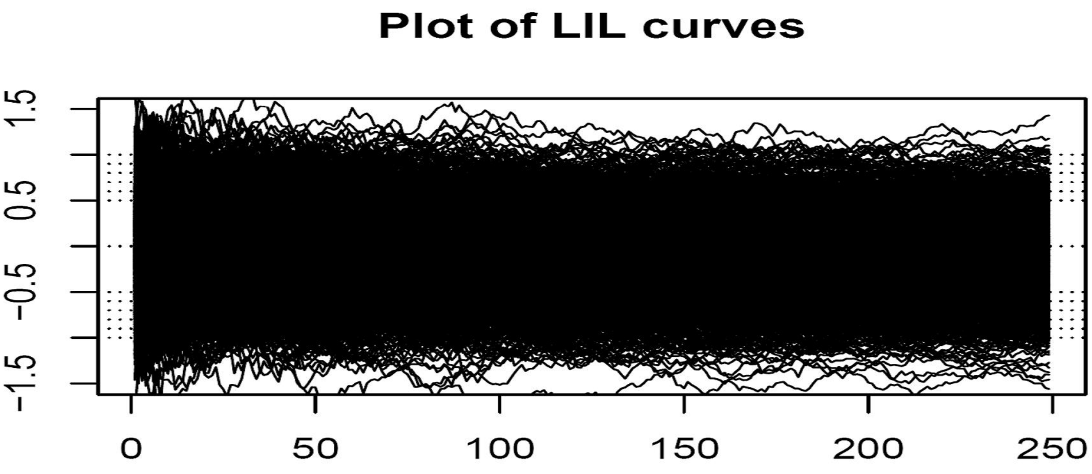
Figure1. Plot for LIL test for the $n=2^{22},2^{23},2^{26}$

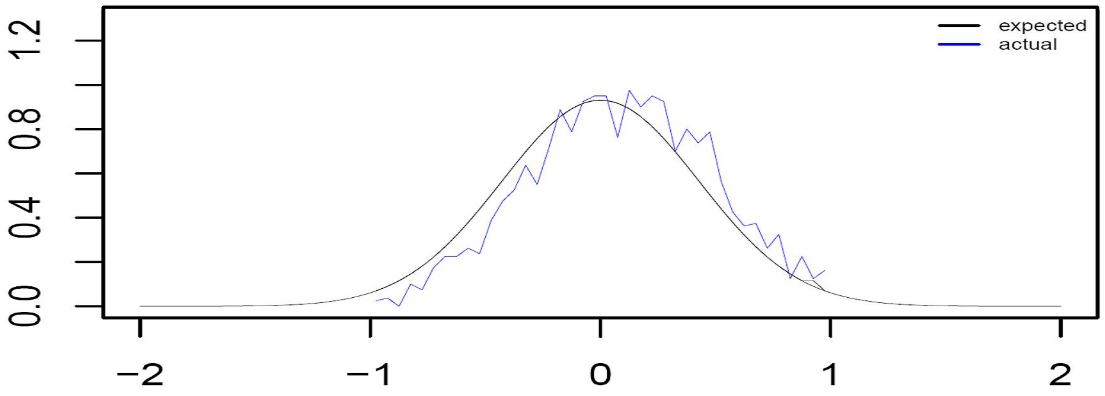







expected
actual


Plot of LIL curves


expected
actual