SN Computer Science (2024) 5:140
https://doi.org/10.1007/s42979-023-02323-w

ORIGINAL RESEARCH

Verifying the Reliability of Quantum Random Number Generator: A Comprehensive Testing Approach

Rounak Biswas[1] · **Dhruv Roy Talukdar**[1] · **Utpal Roy**[1]

Received: 26 April 2023 / Accepted: 12 September 2023

Abstract
Computers typically use pseudo-random numbers generated by algorithms that produce a deterministic sequence of numbers that appear random but are predictable if the entropy of the seed is disclosed. On the other hand advantage of quantum random numbers is that they are generated based on the inherent uncertainty of quantum mechanics, which means they are truly random and unpredictable. This makes them ideal for cryptographic purposes, as attackers cannot easily guess or reproduce them. We proposed a test verifying the randomness of classical and quantum random number generators by running the National Institute of Science and Technology (NIST) test suite. Tests intend to draw attention to whether quantum random numbers match or surpass today's classical random numbers.

Keywords Random numbers · Superposition · Hadamard gate · Quantum random numbers · Randomness test suite · Classical versus Quantum Random Number Generator (RNG)

Introduction

Random numbers are widely used in computer science algorithms. However, current random number generation methods rely on deterministic processes and generate pseudorandom numbers. The effectiveness of these algorithms depends on the randomness of the given entropy and seed value [4]. This dependence on randomness is particularly critical in fields such as cryptography since the entire system's security can be compromised if the random numbers are predictable or easily guessed. Quantum computers possess inherent properties of quantum mechanics, such as uncertainty, which can be utilized to generate random sequences [1]. These sequences can provide better results compared to classical deterministic sequences. Studies have been conducted on quantum random number generators (QRNG), but no standardised QRNG can generate random numbers for use in other algorithms. Different research groups have proposed various techniques, such as using single photon-based QRNGs as a prominent example [25]. This article presents a study comparing classical and quantum random numbers to evaluate the randomness of quantum random numbers, which is still in progress. The National Institute of Science and Technology (NIST) test suite is used to verify the randomness of the given sequence. The study aims to determine if quantum random numbers can exceed or at least match the quality of today's pseudorandom numbers.

Related Work

Classical random or pseudo-random numbers have found widespread use owing to their simplicity and efficiency. The necessity for pseudo-random numbers arises in various domains, each with unique requirements. One such area is cryptography, where the security of cryptographic schemes heavily relies on the random nature of the keys, thus for security mandating the use of either random or

Dhruv Roy Talukdar and Utpal Roy contributed equally to this work.

This article is part of the topical collection "SWOT to AI-embraced Communication Systems (SWOT-AI)" guest edited bySomnath Mukhopadhyay, Debashis De, Sunita Sarkar and Celia Shahnaz.

✉ Rounak Biswas
03333342104@visva-bharati.ac.in

Dhruv Roy Talukdar
03313051909@visva-bharati.ac.in

Utpal Roy
utpal.roy@visva-bharati.ac.in

[1] Computer and System Sciences, Visva-Bharati University, Santiniketan, Bolpur 731235, West Bengal, India

Published online: 03 January 2024

pseudo-random numbers mentioned in this paper [32]. Random numbers must satisfy three fundamental criteria: unpredictability, good statistical properties, and non-reproducibility of their number streams. Pseudo-random number generators (PRNGs) are commonly employed to produce numbers that appear random to users without knowledge of the generator function [29]. These generators utilize mathematical functions and a seed value to generate numbers that successfully pass specific tests for randomness [28]. The research found that pseudo-random numbers still face security concerns due to their deterministic nature and reliance on mathematical functions [21]. Attackers may predict these numbers, potentially leading to vulnerabilities in cryptographic protocols [9].

Various works claim the measurement outcomes of a quantum state hold randomness and unpredictability [5, 33]. The Heisenberg–Robertson uncertainty relation provides theoretical limits for the accuracy of measuring two non-commuting observables. Also, a recent discovery has unveiled a quantum uncertainty relation of coherence between two measurement bases that do not commute [26]. All of these define the quantum mechanic's inherent probabilistic nature. We are taking these properties as the key element of generating random numbers.

Quantum random number generators (QRNGs) have emerged as promising tools for generating high-quality random sources. Some works claim that in most cases QRNGs rely on the intrinsic randomness of quantum mechanics [24]. Various researchers have proposed various implementations of QRNGs, such as those based on correlated photon pairs created through spontaneous four-wave mixing in optical fibres [11], the utilization of quantum computers [14], uncharacterized light sources [34], and the orbital angular momentum of light [12]. Experimental results have showcased the generation of true random numbers at high rates. These quantum random number generation advancements hold significant implications for fields that demand robust and secure random sources.

Lastly, the research paper delves into generating true random numbers using quantum computers [10]. It sheds light on the underlying principles of quantum computing, such as superposition, qubits, and quantum gates, which serve as the foundation for achieving true randomness. By employing these quantum phenomena, quantum computers can generate numbers that exhibit genuine randomness, essential in various applications.

Organization of the Article

This section presents an overview of the remaining parts of the article, highlighting the key topics covered.

"Random Numbers", "Requirement of Randomness" section is a concise introduction to random numbers. Emphasizing their significance in various fields, particularly computer science. It outlines the diverse requirements for randomness in these domains.

Next, in "Classical Deterministic Random Number Generator (DRNG)" to "Quantum Hadamard Gate (H-gate)" section the article delves into the two distinct approaches for generating random numbers: classical and quantum. Further commences by elucidating classical deterministic random number generators and exploring quantum random number generators. The focus then shifts to an in-depth examination of quantum random number generation and the utilization of Hadamard gates to achieve quantum superposition.

"Randomness Test" section introduces the statistical test employed to compare the effectiveness of the random number generation process. For this analysis, we rely on the guidelines suggested by the NIST.

In "Discussion of Test Methods" section the core part of the article commences, which involves a comprehensive description of the three methods for generating quantum random numbers. The article provides detailed insights into the algorithms utilized in the generation process, which are separately outlined in an algorithmic format.

Finally, in "Analysis of the Result" section, the article discusses the analysis of the generated quantum random number values, thoroughly examining their properties and implications. The article concludes with a conclusion section and a list of references.

Contributions of the Study

Our research pursued two primary objectives: firstly, to assess the quality and statistical characteristics of classical and quantum methods for generating random numbers, and secondly, to present a comparative analysis of the findings through tabular and graphical representations.

Through an extensive performance analysis and rigorous statistical tests, our study reveals that quantum random number generation matches traditional classical methods. Furthermore, as quantum technologies advance, the disparity between the two approaches widens significantly, paving the way for a potential future where quantum random number generators become the preferred choice.

Random Numbers

Random numbers refer to numbers generated unpredictably, without any discernible pattern, leading to a sequence of numbers that appear to be random. They find wide-ranging applications in computer science, cryptography, statistics, and other domains where unpredictability and randomness are crucial [7, 22]. Random numbers follow these properties...

- *Unpredictability* The numbers generated should be unpredictable and free from any persistent pattern or structure.
- *Independence* The generated numbers should be independent. Numbers generated in time T_x do not contain any information on numbers generated in both T_{x-1} and T_{x+1}.
- *Uniform distribution* The numbers generated should be evenly distributed across possible values.

Requirement of Randomness

Random numbers have a wide range of use cases in various fields. Here are some examples:

- *Cryptography* Random numbers are crucial in generating secret keys for encryption and decryption. Predictable secret keys can be easily deciphered, leading to compromised system security. Generation of One Time Password (OTP) and Captcha code also required randomness.
- *Simulation* Utilized to generate input parameters for simulating real-world scenarios. For instance, random numbers are used in weather simulations to generate temperature and humidity values for various locations.
- *Genetic algorithm* Random numbers are used to initialise the population, where a set of potential solutions are randomly generated to start the optimization process. They are also used in the selection process, where individuals with higher fitness scores are more likely to be selected for the next generation. However, there is still some random chance involved in introducing diversity.
- *Machine learning* Play a significant role in initializing the model's weights. This random initialization helps avoid the risk of the model getting stuck at a local minimum. Random numbers are also used to select a random data subset for training and validation.
- *Quality control* Random numbers are utilized in manufacturing to choose a random sample of products for quality control purposes. This method of random selection guarantees that the selected sample accurately represents the entire population of products.
- *Gaming* Random numbers are used in gaming applications to add an element of unpredictability to the game. For example, a good source of randomness is needed to shuffle the cards in a card-based game.

Classical Deterministic Random Number Generator (DRNG)

A classical random number generator is a system that produces a sequence of random numbers using a deterministic algorithm, either in software or hardware [3]. The algorithm takes an input seed value as entropy to determine the sequence of random numbers. Although the output appears spontaneous, it is not truly random because it is based on a deterministic process. The randomness of the output depends on the seed value's unpredictability and the core deterministic algorithm's complexity. The core process of general DRNG is given in Fig. 1. The given image contains all the components and parameters used in DRNG.

Some important parts of the classical deterministic random bit generator are as follows-

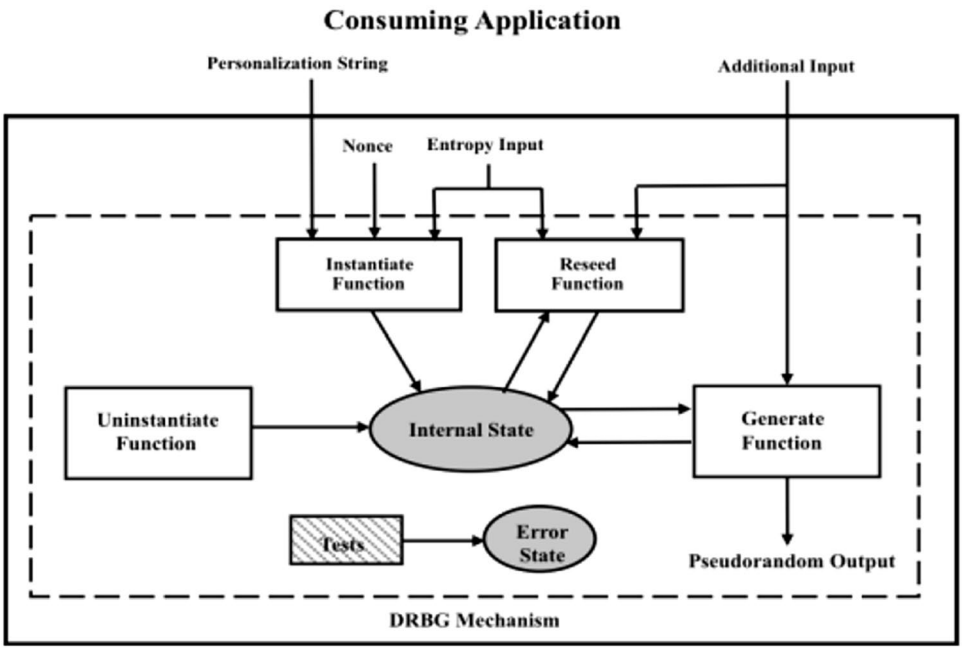

Fig. 1 DRBG Functional Model

- *Entropy Input* Entropy input is a confidential part that needs to be kept secret. Any disclosure of that makes random numbers predictable. The whole security and randomness of the mechanism depend on the entropy value.
- *Instantiating Function* It take Entropy value, nonce (number use only ones), and some additional optional input. Its purpose is to provide the starting internal state value that can provide random bits.
- *Reseed Function* Every generator has some time limit; after that needs to recalculate and create a new seed and internal state. The same starting entropy could not provide any randomness after a specific point and gives a predictable string.
- *Generate Function* The purpose is to generate random bits using the current internal state of the Deterministic Random Bit Generator (DRBG) mechanism.
- *Un-instantiate Function* The role of this function is to clear the whole internal state to all 0s̀ upon the specific requirements.

Quantum Random Number Generator (RNG)

A quantum random number generator utilizes the inherent properties of quantum mechanics to produce unpredictable and truly random numbers. This distinguishes it from classical random number generators that use a deterministic algorithm. Quantum mechanics provides uncertainty that can help to generate more accurate and random numbers. Various methods are available for developing a quantum random number mechanism, among which photon-based QRNG is widespread and easily accessible [30]. It generates random numbers by measuring the inherent randomness of quantum processes involving photons. The fundamental concept behind this type of Quantum Random Number Generator (QRNG) is to utilize a photon source (e.g., a laser) to produce a sequence of photons and then measure the unpredictable arrival times of these photons at a detector. A model of photon-based QRNG is depicted in Fig. 2.

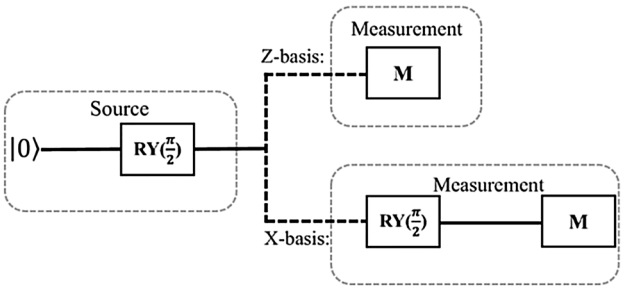

Fig. 2 Single photon based QRNG model

Quantum Superposition and Inherent Randomness

Quantum superposition is a fundamental and captivating property in quantum mechanics, setting it apart distinctly from the classical world. It characterizes a state where a quantum system can exist in multiple states simultaneously. A linear combination of several basis states, each corresponding to the possible outcomes of a specific measurement or observation, represents the state of a quantum system. These basis states' coefficients, known as probability amplitudes, dictate the likelihood of the system being observed in each basis state. The inherent randomness of quantum superposition stems from the system's uncertainty until a measurement is made. Before observation, the system exists simultaneously in all possible states, defying predictability with certainty.

A remarkable example highlighting this phenomenon is the renowned double-slit experiment, where a particle simultaneously passing through two slits exhibits interference patterns, showcasing the essence of superposition. The experiment uses an optical system with liquid crystal spatial light modulators [8].

Quantum superposition carries profound implications for both quantum computing and random number generation. In quantum computing, quantum bits (qubits) can exist in superpositions of 0 and 1. Still, upon measurement, the superposition collapses, yielding a definite state with an outcome determined by the squared magnitude of the probability amplitude.

Quantum processes, like photon polarization or quantum tunnelling, generate random outcomes that inherently elude predictability due to the superposition of quantum states. These outcomes can then be harnessed to obtain genuinely random numbers. In summary, quantum superposition is a foundational principle of quantum mechanics, giving rise to intrinsic randomness in quantum systems, an aspect of utmost importance in quantum computing and random number generation.

Quantum Hadamard Gate (H-gate)

The quantum Hadamard gate is often called "H" in quantum computing. The Hadamard Gate is named after the French mathematician Jacques Hadamard. The Hadamard gate acts on a qubit and transforms its state according to a specific matrix operation [2]. Mathematically, the Hadamard gate can be represented as follows:

$$\mathbf{H}^{\otimes^n}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y\in\{0,1\}^n} (-1)^{x\cdot y}|y\rangle$$

In a computational basis, where $|0\rangle\&|1\rangle$ represent the basis states of a qubit, the Hadamard gate can be represented as a 2x2 matrix:

$$H = \frac{1}{\sqrt{2}} \cdot \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

When applied to a qubit in the $|0\rangle$ state, the Hadamard gate transforms the form as follows:

$$H|0\rangle = \frac{1}{\sqrt{2}} \cdot \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \cdot (|0\rangle + |1\rangle) = |+\rangle$$

Similarly, when applied to a qubit in the $|1\rangle$ state, the Hadamard gate transforms the form as follows:

$$H|1\rangle = \frac{1}{\sqrt{2}} \cdot \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \cdot \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{1}{\sqrt{2}} \cdot (|0\rangle - |1\rangle) = |-\rangle$$

The Hadamard gate is crucial in creating superposition states, essential for generating quantum random numbers using quantum measurements. In the following way, superposition is achieved using H-gate:

- *Initialization* Start with a single qubit in the $|0\rangle$ state, representing the initial state of the quantum system.
- *Apply Hadamard Gate* Apply the Hadamard gate (H) to the qubit. This transforms the qubit into a $|0\rangle\&|1\rangle$ states superposition.
- *Quantum Measurement* Perform a quantum measurement on the qubit. This measurement will yield either $|0\rangle or |1\rangle$ as the outcome. The measurement outcome will be probabilistic since the qubit is superposed after the Hadamard gate. The probability of measuring $|0\rangle or |1\rangle$ will equal, meaning the quantum measurement will produce random outcomes.
- *Repeat for More Bits* To generate multiple quantum random bits, repeat the process by applying the Hadamard gate to additional qubits and performing quantum measurements on each.

This article capitalizes on two distinct sources of quantum randomness, one of them being the H-gate QRNG. The rationale behind selecting the H-gate is grounded in two primary considerations. Firstly, adopting the quantum Hadamard gate within the Qiskit environment facilitates the creation of quantum superposition—a pivotal characteristic of quantum randomness [23, 31]. Secondly, the H-gate exhibits a remarkable overall fidelity exceeding 98%, signifying its efficacy in generating quantum superposition states [18]. The concept of fidelity assumes significance as it gauges the proximity of the actual quantum state of real-world qubits to the ideal scenario. It quantifies the accuracy of the output qubit resulting from the gate's operation.

Randomness Test

Various statistical tests can be used to evaluate the randomness of a given sequence [16]. These tests examine the series's distribution and patterns by measuring bias or correlations to determine its randomness. Statistical tests verify that a given sequence of random numbers is genuinely random. However, it is impossible to provide definitive proof of the randomness of a sequence. Passing statistical tests is not a guarantee of randomness, but failing them can be a strong indication of non-randomness, which can help improve the quality of the random number generator. Several test suites are available to test the randomness of a given sequence [15], including NIST Statistical Test Suite, Diehard Test Suite [20], TestU01 [13], and PractRand. We have used the NIST test suite to evaluate the randomness of classical and quantum random numbers [19].

The National Institute of Standards and Technology (NIST) Test Suite

The NIST provides a statistical test suite for random and pseudo-random number generators to evaluate the quality of a given sequence [6]. The NIST suite includes randomness, distribution, uniformity, independence, and more tests. The NIST tests are widely used in the cryptography community to validate the random number generators used in cryptographic systems and ensure they meet the necessary security standards. There are sixteen statistical tests on which a random sequence is taken as input; the generated result gives information about whether the sequence is random. Some of the most commonly used tests in the NIST suite are:

- *Frequency (Mono-bit) Test* Evaluates the distribution of 0 s and 1 s in a binary sequence.
- *Block Frequency Test* Evaluates the distribution of blocks of consecutive bits in a binary sequence.
- *Run Test* Evaluates the distribution of runs of consecutive 0's or 1's of the same value in a binary sequence.
- *Longest Run of Ones in a Block Test* Evaluates the longest run of 1 s in a block of binary digits.
- *Rank Test* Evaluate the distribution of the number of 0 s and 1 s in a binary sequence.
- *Discrete Fourier Transform (DFT) Test* Evaluates the non-randomness of a binary sequence using the Discrete Fourier Transform.
- *Non-overlapping Template Matching Test* Evaluates the non-randomness of a binary sequence using a sliding window of fixed length.

NIST P-value

The p-value also called the probability value, is a crucial statistical tool in hypothesis testing. In assessing randomness, this measure is pivotal in evaluating the quality of a random number sequence generated by a random number generator (RNG). Emphasized by the National Institute of Standards and Technology (NIST), the significance of the p-value lies in its ability to act as a decisive criterion for evaluating the quality of random number sequences. A p-value is computed for each test when conducting the NIST randomness tests. This computed p-value is then compared against a predefined significance level denoted by α, commonly set at 0.01 or 0.05 [17]. If the calculated p-value exceeds α, the sequence successfully passes the test, indicating satisfactory levels of randomness. Conversely, if the p-value is less than or equal to α, the sequence fails the test, signifying a lack of the desired randomness properties.

Discussion of Test Methods

We have conducted tests on classical and quantum random numbers using the same statistical tests provided by the NIST test suite. We used two sources for quantum random numbers: a local 512-qubit Hadamard gate-based RNG and a stored database containing quantum random numbers. "Local H-gate Based RNG" section is for the 512-qubit RNG sequence test. "Australian National University (ANU) Random Number Database", "Multi-thread Random Numbers Access" and "Continuous Single Process Random Number Access" sections shows test procedure using quantum random number database.

Local H-gate Based RNG

We ran our local quantum random generator based on a 512-qubit Hadamard gate for 10, 100, and 1000 iterations and saved the resulting values in files. The P-values of the generated numbers are obtained and analyzed using the NIST test suite function. Although the quantum circuit used for generating these numbers requires significant time, the simple circuit can still produce considerable randomness. The results of the tests indicate that the uncertainty inherent in quantum mechanics can contribute to producing high-quality random sequences. The results suggest that the 512-qubit Hadamard gate-based QRNG can produce high-quality random sequences. The algorithm for H-gate QRNG is presented in a pseudo-code format as algorithm [1] and the Fig. 4 represents the result of this test.

Australian National University (ANU) Random Number Database

The Australian National University offers two distinct methods to access their quantum random numbers [27]. One option for accessing quantum random numbers is using their API to access live random sequences, while another is accessing their stored random database. We chose to use the stored database as live access has limitations in accessing qubits per session. We modelled our test in two different sections. One involved multiple threads accessing random numbers at a specific time, while the other involved a single process accessing random numbers continuously in a predefined manner.

Algorithm 1: H-gate QRNG (Using IBM qiskit package)

Require: $Packages - QuantumCircuit, QuantumRegister, etc\ldots$
$q \leftarrow QuantumRegister(n\_qbits)$ ▷ Create specified qubit circuit
$c \leftarrow ClassicalRegister(n\_bits)$
$circuit \leftarrow QuantumCircuit(q, c)$
$circuit.h(q)$ ▷ Apply Hadamard gate
$circuit.measure(q, c)$
$backend \leftarrow provider.get\_backend(`simulator\_stabilizer')$
$nth \leftarrow runs$
$measure\_values \leftarrow []$
for $i \leftarrow 1$ to nth **do** ▷ Run circuit multiple times
  $job \leftarrow$ execute($circuit$, $backend$, shots=1024)
  $counts \leftarrow job.result().get\_counts()$
  $measure\_values \leftarrow measure\_values + (counts.keys())$
end for
$f \leftarrow$ open( ) ▷ Create a file for storing values
for $i \leftarrow 1$ to nth **do**
  f.write($measure\_values[i]$)
end for

Multi-thread Random Numbers Access

We have created a multi-threaded test where 50 threads generate concurrent random numbers for a specified period. The equivalent algorithm in pseudo-code format is presented in the algorithm [3]. We then run the NIST test suite on each sequence of random numbers for both classical and quantum. Finally, we use the Python matplotlib package to plot p-values. The results are presented in Fig. 5.

Continuous Single Process Random Number Access

This test done for a single process generate 512-bit random numbers 1000 times over a given time limit continuously. The equivalent algorithm in pseudo-code format is presented in the algorithm [4]. The NIST test suite is used to obtain p-values for each sequence, and these p-values are plotted using a Python matplotlib package. The results are

Algorithm 2: Common Algorithm for Random number generation

Require: (M, X)
Ensure: List of M random strings
  $RandomStrings \leftarrow [\,]$
  for i from $1 \ldots M$ **do**
    Initialize $RandomStr$ as an empty string
    for j from $1 \ldots X$ **do**
      $ch \leftarrow$ randomly assign either 0 or 1
      Append ch with $RandomStr$
    end for
    Add $RandomStr$ with $RandomStrings$
  end for
  return $RandomStrings$

Algorithm 3: Algorithm for Multi-thread Access

Require: (N, M, X)
Ensure: List of M random strings each having X bits generated by N processes
  $P \leftarrow [\,]$
  $RandomStrings \leftarrow []$
  for i from $1 \ldots N$ **do**
    $process_i \leftarrow$ spawn a new process that will run $GenerateRandomStrings(M, X)$ and store the result in $RandomStrings[i]$
    Add $process_i$ to P
  end for
  for i from $1 \ldots N$ **do**
    $P[i].wait()$ // wait for i^{th} process to finish
  end for
  Run NIST test-suite for each of the M strings for N processes

Require: (M, X, T)
Ensure: $p - values$ associated with M random numbers (each having X bits) generated over time interval of T.
  $P\_values \leftarrow [\,]$
  for i from $1 \ldots M$ **do**
    $R \leftarrow$ GenerateRandomStrings(1,X)
    $P\_val \leftarrow p - value$ for R calculated using NIST test suite
    Add P_val to P_values
    SLEEP(T) ▷ to pause the execution for T time
  end for
  return P_values

Algorithm 4: Algorithm for continuous single process access

presented in Fig. 6 and equivalent pseudo-code displayed using algorithm [4].

Analysis of the Result

The analysis section of this study is bifurcated into two distinct segments. Initially, *p*-values are visualized through a tabular representation and a time/*p*-value chart, offering a comprehensive perspective on their distribution and variation. Subsequently, the subsequent section delves into a thorough exposition of the insights gleaned from the presented *p*-value tables and charts. This section enhances the understanding of the analytical outcomes and underscores the pivotal aspects that arise from the *p*-value analysis.

P-value Table and Time/P-value Chart

Figure [3] provided compiles the p-values resulting from 15 distinct statistical tests outlined in the NIST documentation. It is important to note that the table selectively presents data from extensive test datasets. Precisely, the focus is placed on a designated subset of random number values corresponding to the time interval spanning from t_{96} to t_{100}. In this context, t_1 signifies the instance at which the initial execution of the mentioned random number generation method, as detailed in the algorithm [2], concludes. As per the documentation's standards, values falling within ranges between 0.0 and 1.0 are deemed to possess a non-random.

Figure 4, illustrates the outcomes in X–Y plots (time vs. *p*-value), where the *x*-axis depicts time in an increasing sequence, while the *y*-axis represents the *p*-value. Specifically, Fig. 4 focuses on the H-gate QRNG. The observations indicate that there is no prominent decline in *p*-values over time. This finding establishes that a simple quantum circuit integrating quantum superposition constitutes a robust source of high-quality randomness.

Moving on to Fig. 5, we delve into multi-thread access. This chart features X–Y plots where the *x* and *y*-axis correspond to time. Both classical and quantum random numbers are integrated here. At each time instance t_i, 50 threads are concurrently generated, producing random

Exe. No.	Freq Mono	Freq Block	Runs Test	Longest Run	Binary Mat	DFT	Non-overlap	Overlap	Maurer	Linear Com	Serial	Appr Entro	Cusums	Random Excur	Random Var
Hadamard gate QRNG															
t96	0.79088	0.87500	0.59799	0.23776	-1.00000	0.74560	0.99925	nan	-1.00000	-1.00000	0.49896	1.00000	0.81877	0.81358	0.28884
t97	0.07710	0.05658	0.88949	0.04110	-1.00000	0.08851	0.50922	nan	-1.00000	-1.00000	0.49896	1.00000	0.09346	0.50906	0.70546
t98	0.07710	0.42319	0.09878	0.14417	-1.00000	0.37228	0.99925	nan	-1.00000	-1.00000	0.49896	1.00000	0.15420	0.50249	0.91605
t99	0.47950	0.72426	0.46544	0.23495	-1.00000	0.74560	0.99925	nan	-1.00000	-1.00000	0.69077	1.00000	0.77869	0.46662	0.31187
t100	0.59588	0.65586	0.84985	0.21493	-1.00000	0.25614	0.50922	nan	-1.00000	-1.00000	0.69077	1.00000	0.61423	0.04880	0.36131
Multi-thread Access (Classical)															
t96	0.01701	0.00562	0.00313	0.00824	-1.00000	0.00088	0.00001	nan	-1.00000	-1.00000	0.00000	1.00000	0.02209	0.00000	0.00000
t97	0.01701	0.04428	0.01964	0.00919	-1.00000	0.01866	0.00019	nan	-1.00000	-1.00000	0.06722	1.00000	0.01914	0.00000	0.00000
t98	0.00468	0.00511	0.03418	0.00198	-1.00000	0.01192	0.00000	nan	-1.00000	-1.00000	0.00000	1.00000	0.00499	0.00000	0.00000
t99	0.03389	0.02206	0.09129	0.00083	-1.00000	0.01866	0.00019	nan	-1.00000	-1.00000	0.00004	1.00000	0.02957	0.00000	0.00000
t100	0.01701	0.02838	0.03101	0.03289	-1.00000	0.03496	0.00001	nan	-1.00000	-1.00000	0.00000	1.00000	0.02684	0.00000	0.00000
Multi-thread Access (Quantum)															
t96	0.03389	0.00945	0.00777	0.01247	-1.00000	0.00582	0.08933	nan	-1.00000	-1.00000	0.00000	1.00000	0.03857	0.00000	0.00175
t97	0.02156	0.00740	0.01007	0.03892	-1.00000	0.00156	0.00966	nan	-1.00000	-1.00000	0.00645	1.00000	0.03857	0.00000	0.00000
t98	0.00468	0.01880	0.04965	0.00302	-1.00000	0.00582	0.00000	nan	-1.00000	-1.00000	0.00004	1.00000	0.00636	0.00000	0.00000
t99	0.03389	0.04663	0.02220	0.00244	-1.00000	0.03496	0.00000	nan	-1.00000	-1.00000	0.00000	1.00000	0.04414	0.00000	0.00000
t100	0.00614	0.00883	0.02244	0.02235	-1.00000	0.03496	0.00001	nan	-1.00000	-1.00000	0.00000	1.00000	0.00683	0.00000	0.00004
Continuous Single Process (Classical)															
t96	0.83660	0.50925	0.61717	0.86033	0.17811	0.75242	0.13720	0.18851	-1.00000	0.14979	0.93087	0.78636	0.70183	0.89864	0.96440
t97	0.01612	0.35453	0.07703	0.01054	0.88822	0.93144	0.34108	0.08269	-1.00000	0.77130	0.82822	0.08914	0.01523	0.28265	0.05183
t98	0.74046	0.60939	0.63009	0.70053	0.30983	0.68807	0.46075	0.65762	-1.00000	0.45250	0.02729	0.06513	0.78070	0.52387	0.39542
t99	0.13200	0.19568	0.44999	0.88376	0.15681	0.19689	0.31617	0.89020	-1.00000	0.93009	0.41738	0.77670	0.19282	0.73868	0.14896
t100	0.02693	0.03496	0.71955	0.04618	0.77662	0.60572	0.65731	0.03327	-1.00000	0.73465	0.44863	0.58533	0.02596	0.03187	0.41228
Continuous Single Process (Quantum)															
t96	0.83660	0.55473	0.47607	0.71015	0.29097	0.52811	0.32398	0.83999	-1.00000	0.92838	0.47036	0.62238	0.71651	0.25255	0.78286
t97	0.18517	0.87979	0.62202	0.41903	0.44627	0.66708	0.11467	0.02232	-1.00000	0.83791	0.51122	0.48527	0.25922	0.27675	0.25228
t98	0.86599	0.98222	0.77845	0.90918	0.66760	0.28867	0.72057	0.44140	-1.00000	0.05569	0.09795	0.09017	0.77494	0.46693	0.35513
t99	0.67997	0.07919	0.00500	0.26025	0.04054	0.81855	0.17892	0.03844	-1.00000	0.72711	0.70052	0.14665	0.79502	0.83594	0.78353
t100	0.43099	0.16637	0.08760	0.71842	0.65623	0.95426	0.16818	0.71443	-1.00000	0.43750	0.44793	0.48389	0.77204	0.76225	1.00000

Fig. 3 P-values of a specific time period of all NIST test

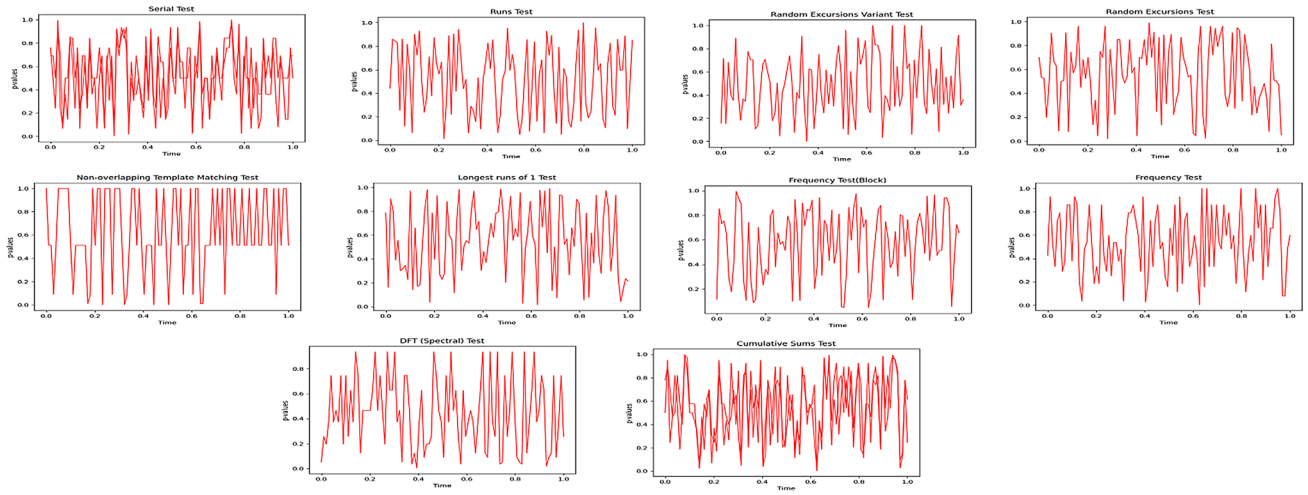

Fig. 4 Hadamard gates based QRNG

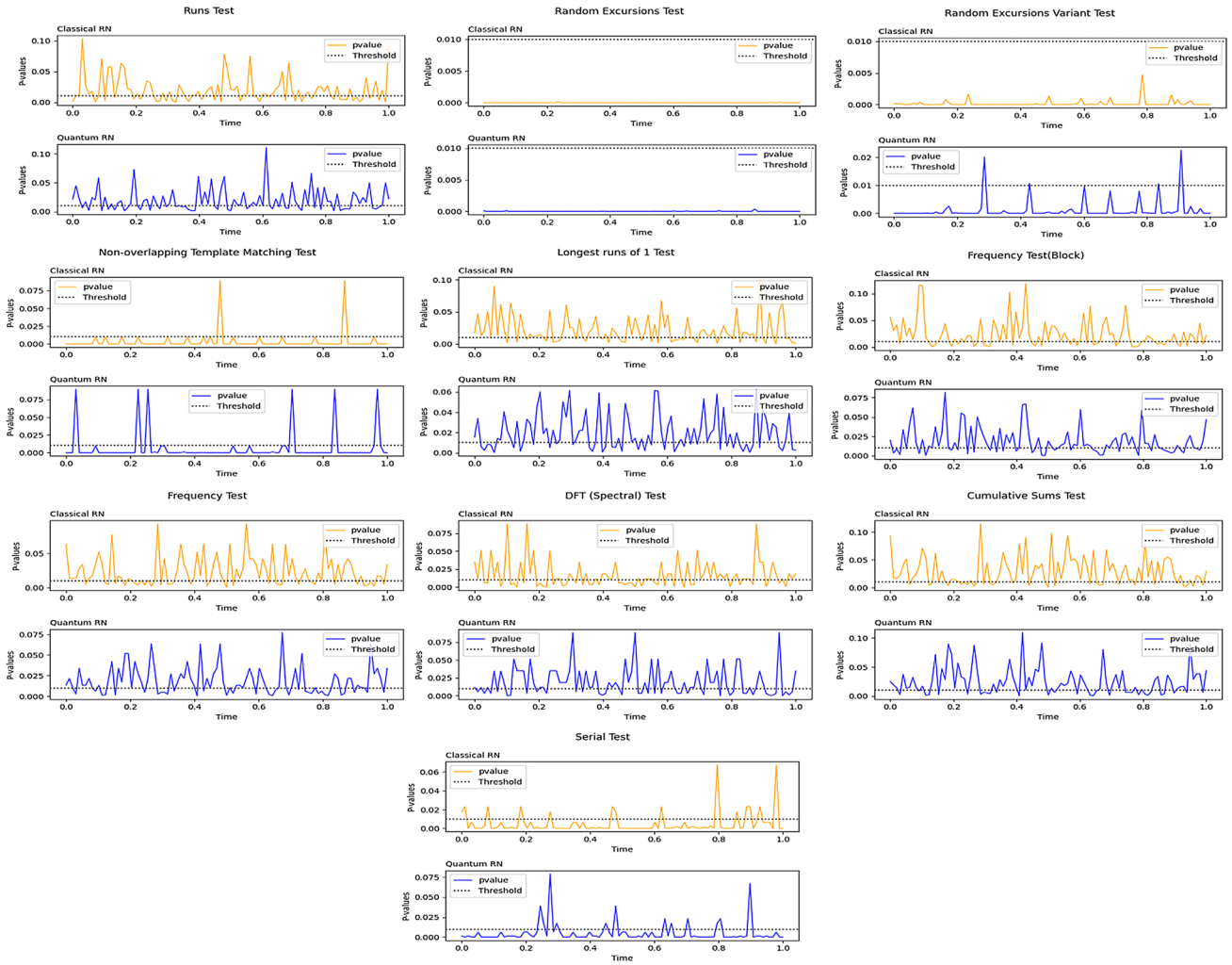

Fig. 5 Multi-thread test using ANU test database

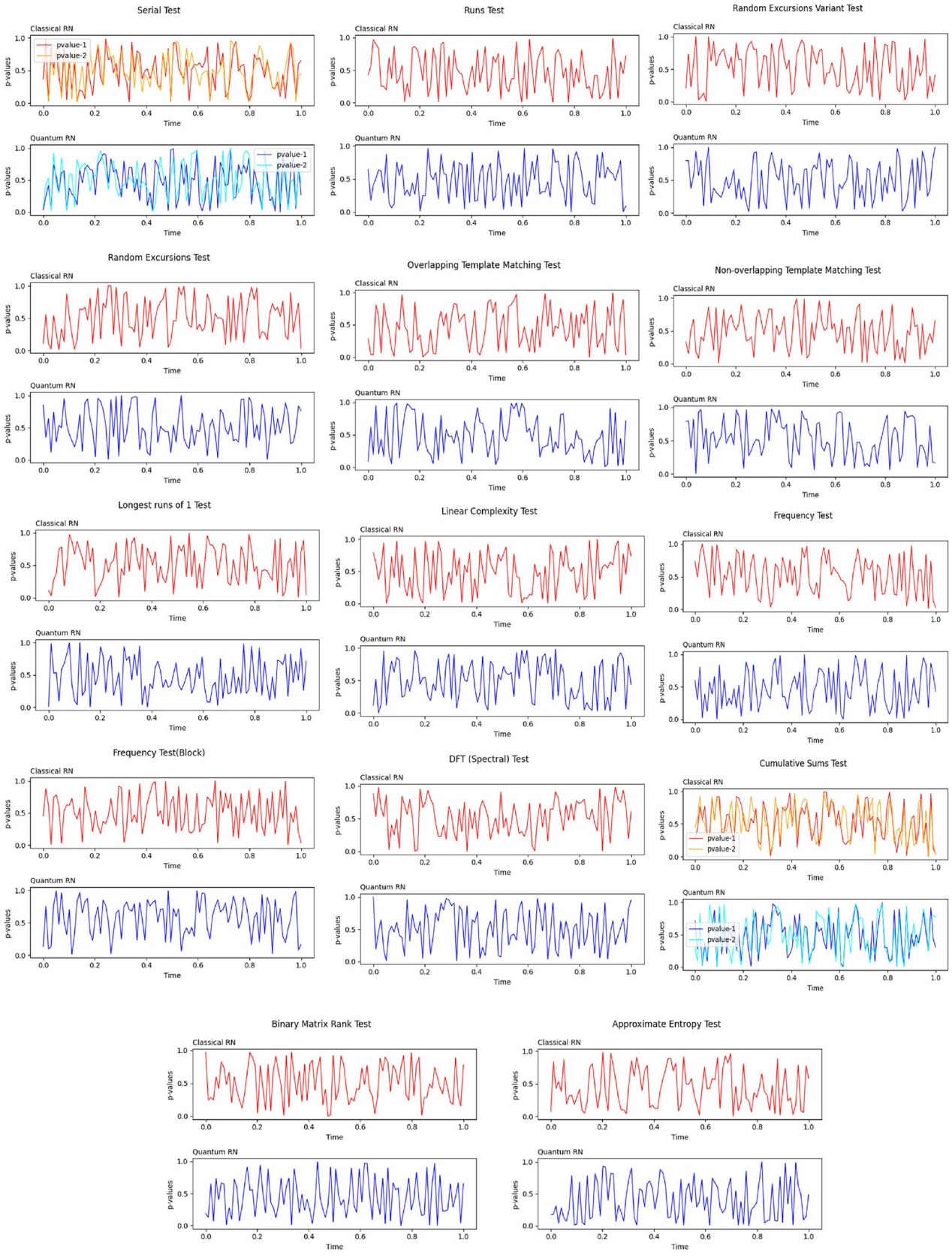

Fig. 6 Consecutive test using ANU test database

numbers. After the NIST tests, each of the 50 threads yields 15 distinct p-values. The plotted data showcases the minimum p-value among the 50 threads for each statistical test at time instance t_i. It is worth noting that selecting the minimum p-value is crucial for assessing randomness.

Figure 6, introduces the results from the single process continuous access test. Similar to the previous figures, this chart comprises X-Y plots where the x and y-axes convey analogous meanings. Both classical and quantum random numbers are considered once again. This time, the chart focuses on the continuous access scenario, demonstrating the behaviour of both types of random numbers.

Discussion of Obtained Result

All three tests aim to show that quantum random numbers are on par with the current pseudo-random numbers. By examining the p-value table and equivalent plots, some crucial observations have been made regarding the changes in p-values over time.

- *General observation* The fact that the p-values of quantum random numbers do not decrease over time implies that their performance remains consistent even after generating a large number of random numbers. This suggests that prolonged use does not affect the quality of quantum random numbers.
- *H-gate QRNG* Figure 4 illustrates that the H-gate QRNG yields favourable outcomes when executed on a simulator. However, if the same circuit is run on a physical machine, the results are expected to exhibit greater randomness.
- *Stored Database* Regarding the tests conducted with stored data values shown in Figs. 5 and 6, it is observed that some of the results obtained from quantum random numbers exhibit an upward trend over time.

This study contributes to our comprehension of quantum random numbers by demonstrating that they can compete with contemporary classical pseudo-random numbers. This indicates that quantum random numbers have the potential to be a viable alternative in various applications that require secure and unpredictable random numbers. Also worth noting is that the results would be more significant if further testing is conducted using a more accurate and live quantum generator. Ignoring the limitations of quantum hardware, the results displayed in the plotted diagrams indicate that quantum random numbers are a promising area of research. With further advancements in the future, quantum random number generators will likely outperform deterministic random number generators.

Conclusion

In the article's concluding section, we discuss the potential of using quantum random numbers to replace classical DRNGs. Additionally, we highlight how the inherent uncertainty of quantum computing can be leveraged to produce a reliable source of randomness

- *Summary of main results* Two quantum data sources were subject to three tests to determine their randomness. The p-value plot obtained from classical and quantum sources showed that quantum random numbers provide a reliable source of randomness.
- *Implications* With the quantum computer, we can include the uncertainty principle in the generation process of random sequences. Now that quantum mechanics is random and probabilistic, we hope it helps create accurate or near-true random sequences.
- *Limitations* The limitations of quantum computers include limited hardware resources, high cost, lack of skilled technicians etc. Additionally, the 512-qubit Hadamard circuit used in this study can capture even more probabilistic outcomes if executed on physical hardware.
- *Recommendations* Despite the hardware limitations, quantum computers can provide better random number generation than available deterministic random number generators. Therefore, it is recommended to conduct stress tests on classical versus quantum random number generators using different methods, such as running the test algorithm for a long time and measuring the points where either fails. To further advance the study of quantum random numbers, it is recommended to test classical versus quantum random number generators using other parameters. This article only tests two types of quantum random number generator methods, and in the future, exploring other sources of quantum random numbers would be interesting.

Funding Doesn't use any funding sources for the study.

Data and Materials Availability Our used QRNG data is from a stored database by the ANU organization. Publicly accessible at https://cloudstor.aarnet.edu.au/plus/s/9Ik6roa7ACFyWL4.

Declarations

Conflict of Interest All Author declares that he or she has no conflict of interest.

Informed Consent and Animal Welfare This article contains no studies with human participants or animals performed by any of the authors.

References

1. Abdelgaber N, Nikolopoulos C. Overview on quantum computing and its applications in artificial intelligence. In: 2020 IEEE Third International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), IEEE. 2020; pp 198–199.
2. Aharonov D. A simple proof that toffoli and hadamard are quantum universal. arXiv preprint quant-ph/0301040. 2003.
3. Barker E, Kelsey J. Recommendation for random number generation using deterministic random bit generators. 2015. https://doi.org/10.6028/NIST.SP.800-90Ar1.
4. Barker EB, Kelsey JM. Sp 800-90a. recommendation for random number generation using deterministic random bit generators. 2012.
5. Barrett JA, Huttegger SM. Quantum randomness and underdetermination. Philos Sci. 2020;87 (3):391–408.
6. Bassham L, Rukhin A, Soto J, et al. A statistical test suite for random and pseudorandom number generators for cryptographic applications. 2010. https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=906762.
7. Chlumeckỳ M, Buchtele J, Richta K. Application of random number generators in genetic algorithms to improve rainfall-runoff modelling. J Hydrol. 2017;553:350–5.
8. Cofré A, Vargas A, Torres-Ruiz FA, et al. Dual polarization split lenses. Optics Expr. 2017;25 (20):23773–83.
9. Cremers C, Garratt L, Smyshlyaev S, et al. Randomness improvements for security protocols. Internet Requests for Comments, RFC Editor, RFC 8937. 2020.
10. Deepak Chahal PK. Analysis of quantum computers and randomness. Int J Innov Res Technol. 2020;6 (11):531–4.
11. Huo M. Deterministic quantum teleportation through fiber channels. Sci Advan. 2018. https://doi.org/10.1126/sciadv.aas9401.
12. Hurley-Smith D, Hernandez-Castro J. Quantum leap and crash: Searching and finding bias in quantum random number generators. ACM Trans Privacy Security (TOPS). 2020;23 (3):1–25.
13. L'ecuyer P, Simard R. Testu01: Ac library for empirical testing of random number generators. ACM Transactions on Mathematical Software (TOMS). 2007;33 (4):1–40.
14. Li YH, Han X, Cao Y, et al. Quantum random number generation with uncharacterized laser and sunlight. Quantum Inform. 2019;5 (1):97.
15. Mannalath V, Mishra S, Pathak A. A comprehensive review of quantum random number generators: Concepts, classification and the origin of randomness. arXiv preprint arXiv:2203.00261. 2022.
16. Martin-Löf P. The definition of random sequences. Inform Control. 1966;9 (6):602–19.
17. Marton K, Suciu A. On the interpretation of results from the nist statistical test suite. Sci Technol. 2015;18 (1):18–32.
18. Muhonen JT, Laucht A, Simmons S, et al. Quantifying the quantum gate fidelity of single-atom spin qubits in silicon by randomized benchmarking. J Phys Condensed Matter. 2015;27 (15): 154205.
19. Paul R, Dey H, Chakrabarti A, et al. Accelerating more secure rc4 : Implementation of seven fpga designs in stages upto 8 byte per clock. arXiv:1609.01389. 2016.
20. Rütti M, Troyer M, Petersen WP. A generic random number generator test suite. arXiv preprint math/0410385. 2004.
21. Sathya K, Premalatha J, Rajasekar V. Investigation of strength and security of pseudo random number generators. In: IOP Conference Series: materials Science and Engineering, IOP Publishing, p 012076. 2021.
22. Schindler W. Random number generators for cryptographic applications. Cryptographic Engineering pp 5–23. 2009.
23. haq Shaik E, Rangaswamy N. Implementation of quantum gates based logic circuits using ibm qiskit. In: 2020 5th International conference on computing, communication and security (ICCCS), IEEE, pp 1–6. 2020.
24. Smirnov M, Petrovnin K, Fedotov I, et al. Quantum random numbers from a fiber-optic photon-pair source. Laser Phys Lett. 2019;16 (11): 115402.
25. Stefanov A, Gisin N, Guinnard O, et al. Optical quantum random number generator. J Modern Optics. 2000;47 (4):595–8.
26. Svozil K. Quantum randomness is chimeric. Entropy. 2021;23 (5):519.
27. Symul T, Assad SM, Lam PK. Real time demonstration of high bitrate quantum random number generation with coherent laser light. Appl Phys Lett. 2011;98 (23): 231103. https://doi.org/10.1063/1.3597793.
28. Thottempudi P, Thottempudi N, Bhushan K, et al. Generation of cryptographically secured pseudo random numbers using fpga. Int J Electr Commun Eng Technol. 2014;5:2.
29. Tuncer T, Avaroğlu E. Random number generation with lfsr based stream cipher algorithms. In: 2017 40th International Convention on Information and Communication Technology. IEEE: Electronics and Microelectronics (MIPRO); 2017. p. 171–5.
30. White SJ, Klauck F, Tran TT, et al. Quantum random number generation using a hexagonal boron nitride single photon emitter. J Optics. 2020;23 (1):01TL01.
31. Wille R, Van Meter R, Naveh Y. Ibm's qiskit tool chain: Working with and developing for real quantum computers. In: 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, pp 1234–1240. 2019.
32. Xw Yang, Xq Zhan, Hj Kang, et al. Fast software implementation of serial test and approximate entropy test of binary sequence. Secur Commun Netw. 2021;2021:1–10.
33. Yuan X, Bai G, Peng T, et al. Quantum uncertainty relation using coherence. Phys Rev A. 2017;96 (3): 032313.
34. Zahidy M, Tebyanian H, Cozzolino D, et al. Quantum randomness generation via orbital angular momentum modes crosstalk in a ring-core fiber. AVS Quant Sci. 2022;4 (1):89.