LWE Based Quantum-Resistant Pseudo-Random Number Generator

Atul Kumar
SOCEMS, Defence Institute of Advanced Technology, India
atulkumar.diat@gmail.com

Arun Mishra
SOCEMS, Defence Institute of Advanced Technology, India
arunmishra@diat.ac.in

Abstract: *In the realm of cryptography, computational statistics, gaming, simulation processes, gambling, and other related fields, the design of Cryptographically Secure Pseudo-Random Number Generators (CSPRNGs) poses a significant challenge. With the rapid advancement of quantum computing, the imminent "quantum-threat" looms closer, posing a risk to our current cryptographically secure PRNGs. Consequently, it becomes crucial to address these threats seriously and develop diverse tools and techniques to ensure that cryptographically secure Pseudo-Random Number Generators (PRNGs) remain unbreakable by both classical and quantum computers. this paper presents a novel approach to constructing an effective Quantum-Resistant Pseudo-Random Number Generator (QRPRNG) using the principles of lattice-based Learning with Errors (LWE). LWE is considered quantum-resistant due to its reliance on the hardness of problems like the Shortest Vector Problem and Closest Vector Problem. Our work focuses on developing a QRPRNG that utilizes a Linear Feedback Shift Register (LFSR) to generate a stream of pseudo-random bits. To construct a secure seed for the QRPRNG, LWE is employed. The proposed QRPRNG incorporates a secure seed input to the LFSR, and employs a Homomorphic function to protect the security of the finite states within the LFSR. NIST statistical tests are conducted to evaluate the randomness of the generated output by the constructed QRPRNG. The proposed QRPRNG achieves a throughput of 35.172 Mbit/s.*

Keywords: *Learning with errors, homomorphic function, pseudo-random number generator, homomorphic function, linear feedback shift register, NIST statistical test suite.*

Received January 25, 2021; accepted October 11, 2022
https://doi.org/10.34028/iajit/20/6/8

1. Introduction

A Pseudo-Random Number Generator *(*PRNG*)* is a function especially mathematical which is based on some deterministic procedures to produce a stream of random numbers. To create nonces, blinding values, challenges, and padding bytes, random numbers are necessary. The creation of private/public key pairs for asymmetric cryptographic algorithms like RSA, Diffie-Hellman, and Digital Signature Algorithm (DSA) is also accomplished using random numbers. The most important element is that neither attackers nor intruders, nor people who are familiar with the design, should be able to predict the results of the random number generator with any degree of accuracy. The security of random numbers is a prerequisite for the security of the various protocols. The entire system based on that protocol would be vulnerable if the random number is unreliable. This is accomplished by using a Cryptographically Secure Pseudo-Random Number Generator (CSPRNG) like Fortuna [10], Yarrow-160 [1], Counter-Based Random Number Generator (CBRNG) [13], and Blum-Blum Shub [25] in classical computer.

However, as soon as quantum algorithms are developed, a difficulty appears. Since Grover's search method [26] employs quantum computers, the underlying structure of the Fortuna, and Yarrow 160 PRNG which uses SHA-256 and SHA-1 hash functions, respectively, can be broken. CBRNG's security depends on the key (seed) value and encryption technique. Since Advanced Encryption Standard (AES) is typically employed, Grover's search algorithm may be used to recover the seed value, making CBRNG vulnerable to quantum attacks. Blum-Blum Shub, which is predicated on the difficulty of the factorization issue, the seed might be retrieved via factorization using Shor's method [28]. Because these Pseudo-Random Number Generators (PRNGs) are not quantum-resistant, a quantum-resistant PRNG is necessary to ensure that it cannot be compromised by a quantum attack.

In this paper, we developed a plan for a Quantum-Resistant PRNG (QR-PRNG) to address the aforementioned CSPRNG concerns. The proposed plan is divided into two main sections:

1) The building of secure seed.
2) Quantum-Resistant method for generating random bits.

The lattice-based cryptographic method Learning With Errors (LWE) [2] is resistant against quantum computers, according to research by theoretical computer scientist Oded Regev [2] in 2005. The lattice-based LWE technique is used since the primary

objective is to create the quantum-resistant seed as stated above. A polynomial-time solution that might defeat the lattice-based LWE algorithm currently does not exist, neither in classical nor quantum realms [3]. The quantum-resistant method that generates the random bit sequence uses the constructed secure, quantum resistant, seed as an input. The proposed method makes use of distributed Pseudo-Random Functions (PRFs) that may perform any binary operation. The circuit accepts the secure seed as an input (Refer Figure 1). The input is divided into three segments in a way that preserves the lattice features of LWE. Since maintaining the secrecy of the seed is a goal of the homomorphic function, operations on encrypted data are permitted. Because homomorphic function is utilized to construct circuits in a way that allows it to compute on encrypted data (a secure seed) to produce bitstream without losing the lattice features and secure the internal states of the circuit, the suggested technique is quantum-resistant.

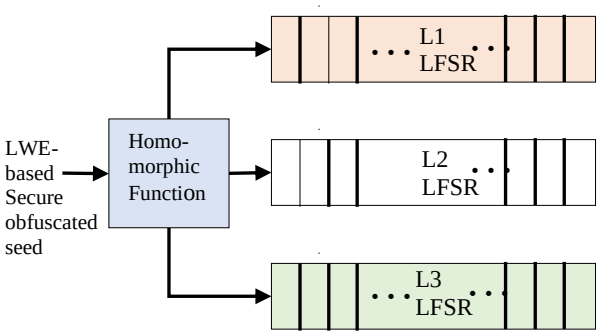

Figure 1. Initial representation of LFSR circuit.

2. Related Work

Through Goldreich, Goldwasser and Micali (GGM) Construction [6], Goldreich, Goldwasser and Micali formalized the idea of PRFs and PRNGs. Another fundamental method of building PRFs known as NR constructs [16] was developed by Naor and Reingold (NR). Theoretically, PRFs and PRNGs are built from the GGM and NR construction, which were derived with particular hardness functions, such as Number Theoretic assumptions and Lattice-based hardness assumptions, leading to effective PRFs. Although there are generic ways to create PRFs, such as the GGM and NR constructions, in practice it is difficult to create effective PRFs using these methods, hence realistic implementation uses the idea of PRNGs. Until the polynomial's period is achieved, these effective PRNGs continue to run. The contemporary construction of PRFs using homomorphic functions is one of the paradigms in seed preservation. The homomorphic function [30], which permits computation on encrypted data, is employed in a variety of cloud storage and Zero Knowledge privacy applications. Since Pseudo-Random Functions (PRFs) are a fundamental component of PRNGs, they could be cracked in polynomial time by quantum algorithms. Consequently, the Lattice based LWE technique could be useful to build an effective and unpredictable PRF in order to secure the PRFs from such types of attacks.

The Linear Feedback Shift Register (LFSR) is one of the most basic and extensively used methods for generating cryptographic pseudo-random number stream. Traditionally, LFSRs have been intended to operate on the binary Galois field GF(2) [12]. This technique is suitable for hardware implementations; however, it has low software efficiency due to two major drawbacks:

1. To accomplish the register shifting or output generating operations required to change the state of an LFSR, the processor must spend several clock cycles.
2. Because binary LFSRs only deliver one output bit per clock pulse, software implementations are expensive and squander contemporary processor capabilities.

Delgado-Mohatar and Fúster-Sabater [8] constructed the LFSR over the extended field GF(2^n) rather than the binary field GF(2). The goal of this kind of construction was to efficiently utilize the modern processor with 2n elements where n is size of the register in the processor [27]. When extended fields are used instead of finite fields, speed factors of up to 10.15 are achieved [8]. However, LFSR construction was not practicable for large n (commended over the extension field GF(2^{16}) as the internal operation cost was so high.

Espinosa Garcia *et al*. [9] proposed two methods to improve the execution time and primitive polynomial searching time in the extended field GF(2^n). The basic polynomial over GF(2^n) would be represented using binary LFSRs. The identical sequences will be created using solely binary operations (XOR) in this manner. The LFSR in GF(2^n) and n binary LFSRs would have a same sequence and would share the same feedback polynomial in GF(2). Thus, the n binary LFSRs' bit operations may be calculated together, resulting in XOR operations across n-bit words and reducing the inefficiency caused by single-bit operations in this sort of processor. Since execution time is getting reduced there probability of a cryptanalysis attack would be reduced therefore security would be improved. Furthermore, the suggested technique allows any primitive polynomial in GF(2^n) to be utilised as an LFSR's feedback polynomial, circumventing the present limits. However, LFSR construction will not be practicable for large n as the internal operation cost would be so high.

Chowdhury *et al*. [7] also proposed a fast correlation attack-resistant block oriented software implementation method of LFSR. The bit stream is formed by aggregating the produced sequences of several distinct LFSRs using the nonlinear Boolean function. However, the actual implementation results were not favorable.

3. Preliminaries

3.1. Lattice-Based Cryptography

The *RSA* algorithm is the most widely used algorithm in a classical computer. Its security hinges on the large integer factoring problem. Using a classical computer, it is challenging to factor a huge composite number with exactly two large prime numbers as its factors in polynomial time. However, Shor's technique may be used to the quantum computer to address this issue in polynomial time [28]. As a result, a quantum computer can easily solve the issue of prime factorization. Despite this, some cryptography methods, such as lattice-based cryptography [23], are secure in the presence of a quantum computer. Such type of cryptography is known as post-quantum cryptography.

One of the main competitors in the area of post-quantum cryptography that establishes the framework of cryptographic primitives involving lattices is lattice-based cryptography. A group of neatly arranged points that are uniformly spaced in an n-dimensional space make up a lattice [17]. On graph paper, points might be seen as a grid. The hardness of the lattice problem, increases with the increase in dimension, determines how secure a cryptosystem is, and the majority of cryptosystems are entirely based on their hardness only. Hard lattice problems include the Closest Vector Problem (CVP) [14] and the Shortest Vector Problem (SVP) [20]. SVP Determine the shortest vector from the origin for the given lattice. CVP for a given point that is not in the lattice, locate the point that is closest in the provided lattice. Lattice-based cryptography's worst-case hardness assumption is the source of its security. It is also protected from quantum computers [22]. Many cryptographic techniques, including digital signatures, public-key encryption, identity-based encryption, encryption resistant to key leakage attacks, and homomorphic encryption, utilise lattices. One of the most important algorithms used in lattice-based cryptography is LWE.

- Learning with Errors (LWE)

The LWE [2] problem is thought to be as hard as the worst-case lattice problem [22]. As a result, the development of such a cryptographic system would be resistant to quantum attacks. Finding the secret vector from any number of noisy linear Equations is the goal of this problem.

$$a_{11}s_1 + a_{12}s_2 + \ \dots\ + a_{1n}s_n + e_1 = b_1 \bmod q$$
$$a_{21}s_1 + a_{22}s_2 + \ \dots\ + a_{2n}s_n + e_2 = b_2 \bmod q$$
$$\dots$$
$$\dots$$
$$\dots$$
$$a_{m1}s_1 + a_{m2}s_2 + \dots + a_{mn}s_n + e_m = b_m \bmod q$$

Let the elements of a matrix A are $(a_{11}, a_{12}, \dots, a_{mn})$, whereas the elements of the vectors s, e, and b are $(s_1, s_2, \dots, s_n)$, $(e_1, e_2, \dots, e_m)$, $(b_1, b_2, \dots, b_m)$ respectively. The aforementioned linear equations can be expressed in abbreviated form as $A.s + e = b \bmod q$. Where m and n denote the rows and columns of the matrix, which contains values that are uniformly and randomly selected, and $s \in \mathbb{Z}_q^m$, $A \in \mathbb{Z}_q^{n\times m}$, $e(error) \in \chi^n$, $q \in$ prime number. The error is chosen at random from a Gaussian distribution with a mean of zero and a small standard deviation. The secret value s must be found in great difficulty since otherwise, these equations can be solved by the Gaussian Elimination technique in polynomial time. The *LWE* based encryption/decryption process may be broken into three sections:

Key Generation:

$$pubKey = \{A, b = A.s + e\}\ and\ secKey = s$$

Encryption:

$$Enc(pubKey, bit) = \{cipher\ text\ preamble\ (u) = A.x,$$
$$cipher\ text\ (u') = b.x + bit.\left\lfloor\frac{q}{2}\right\rfloor\}, where\ x \in \{0,1\}^n$$

Decryption:

$$Dec\big(secKey, (u, u')\big) = \begin{cases} 0\ if\ u' - s.u\ mod\ q \in [\frac{-q}{4}, \frac{q}{4}] \\ 1\ if\ u' - s.u\ mod\ q \in (\frac{q}{4}, \frac{3q}{4}] \end{cases}$$

As $SVP \leq CVP \leq LWE$ [4] describes the SVP, which is indirectly reducible to the Learning with Errors (LWE) problem, has not yet been solved by a quantum computer, the LWE approach would likewise be secured from a quantum attacker.

3.2. Random Number Generator

There are two broad categories:

1. True random number generator [15].
2. Pseudo-random number generator.

While PRNG takes the seed value or initial value from the user or from an external entity like another process and applies mathematical operations over that to produce the stream of random numbers, TRNG takes the input from external natural sources in an analogue form through sensors and then converts it into the digital form and produces the sequence of random number streams (Refer Figure 2.)

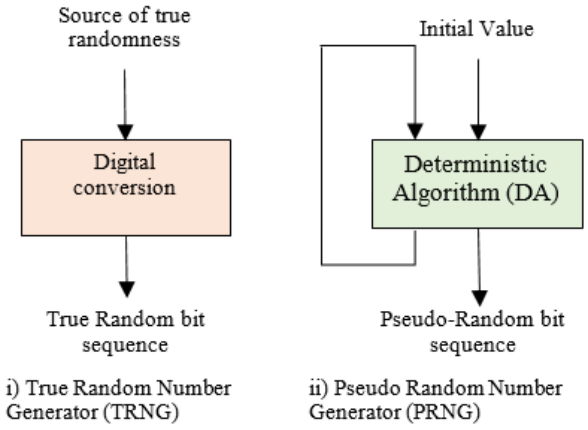

Figure 2. Random number generators.

3.3. Linear Feedback Shift Register (LFSR)

An LFSR [11] is a combination of shift registers in a sequential manner of bits using combinational logic. The output from each shift register is carefully staged and fed back into its input in an LFSR, which enables the function to endlessly cycle through a variety of patterns. The feedback shift register may be broken down into two main components: the shift registers and the feedback function. A group of bits make up the shift register. When the length of a shift register is n bits, the shift register is referred to as an n-bit shift register. If a bit is necessary, all of the bits are typically moved one bit to the right, starting with the least significant bit. Based on other bits in the register, the next leftmost bit is determined. The output of the shift register is often the least significant bit. The period of a shift register refers to the length of the output sequence prior to its repetition. A linear function, often XOR, is used as the input in the simplest type of feedback shift register, known as an LFSR. Tap bits are the bits that modify the LFSR's state. Both the Galois approach and the Fibonacci method can be used to link tap bits. Unlike the Galois approach, which XORs each tap bits with the output stream, the Fibonacci method cascades tap bits and feeds them into the LFSR's leftmost shift register. In cryptography, LFSRs might be used to produce pseudo-random numbers, nonces, etc.

3.4. Quantum Cryptography

In a classical computer, cryptography [5] is a method for carrying out secure communication and protecting information between two parties in the public domain by using mathematical concepts and algorithms. In a quantum computer, however, cryptography is a method that employs laws of quantum mechanics to secure communication and protection of information even when the intruders have access to quantum computing.

4. Proposed PRNG based on LWE

In the current work, a method for generating pseudorandom numbers is provided that takes use of the adaptability of lattice-based cryptography. For the purpose of securing the seed against cryptanalysis, the proposed method opts for Learning with Errors (LWE) implementation of the Lightweight Block Cipher (LBC) [23]. The seed bits are obfuscated by LWE into a vector in the lattice, which is then given to the Linear Feedback Shift Registers (LFSR), where a circuit (composed of three LFSRs) is computed to produce the random sequence. Keeping the lattice vector in the lattice after the circuit generates the feedback sequence for it is one of the more difficult side of this approach. Additionally, we must make sure that the original seed or secret used for LWE encryption is not compromised. To address this, the current scheme computes the value for the circuit using a homomorphic function. On encrypted data, homomorphic functions are performed without disclosing the information of the secret seed.

The scheme is methodically divided into three sections:

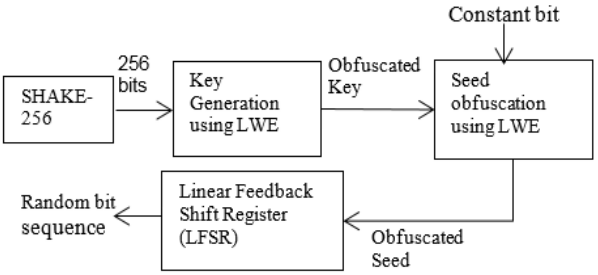

Figure 3. Block diagram of proposed PRNG.

4.1. Seed Obfuscation using LWE

In the current work, we develop the LWE Obfuscation method, which consists of two stages: Key Generation and Seed Obfuscation (Refer Figure 3. *Block diagram of proposed PRNG*), for obscuring the input seed. Key Generation step grabs the input seed to produce the matrix $A \in R_q^{k \times k}$ and a secret key $s \in R_q^{k \times 1}$ and also to produce obfuscation key $A.s + e$ where e represents the uniform Gaussian error distribution. Once the keys have been obtained, the obfuscation vector c is created by passing the input to the Seed Obfuscation function. This is accomplished by first generating the random vector $x \in R_q^{k \times 1}$ and then $c = b^T.x + const\_bit.q/2$. The const_bit is used to increase the obfuscated seed's entropy. To apply the homomorphic function, the LFSR circuit will receive the obfuscated seed, "c".

4.2. Homomorphic Function

After evaluating obfuscated seed, homomorphic function enables operations on the obfuscated seed with the aim of maintaining privacy. One of the most crucial uses of lattice-based cryptography has proven to be homomorphic encryption [24]. In the current work, homomorphic function is utilized to design circuits that compute on encrypted data to generate sequence without losing the lattice properties. The distributed

PRFs on the $\mathbb{Z}_q$ vector and the ○ 'a binary' operation are both used by the proposed PRNG generator. Input from the LWE obfuscation function "c". is divided into three parts by the circuit such that the operations adhere to the following circuit equality:

$$Homo\_LWE(c)=Homo\_LWE(c_1)\circ Homo\_LWE(c_2)\circ Homo\_LWE(c_3)$$

Where ○ is any binary operation and *Homo_LWE(c)* is the function that LFSR obeys that does not interfere with the lattice properties of LWE. To determine if the circuit retains the LWE structure after homomorphic translation, we would establish the following theorem.

- *Theorem*: If a circuit *Homo_LWE* with operation ○ is used, the evaluation of the LWE obfuscated vector c is performed using the equality shown below:

$$Homo\_LWE(c) = Homo\_LWE(c_1) \circ Homo\_LWE(c_2) \circ Homo\_LWE(c_3)$$

Where the distributed seeds produced from the LWE obfuscation function c_1, c_2, and c_3.

- *Proof*: Seed obfuscation function is defined as $c = b.x + const\_bit.q/2$. Let c_1, c_2 and c_3 represent the divided outputs of the *Homo_LWE(c)* with ○ operation, respectively, where c_1, c_2, and c_3 serve as inputs to each distinct circuit function. The following equations are obtained by dividing the c results into three parts:

$$c_1 = b_1.x_1 + const\_bit_1.q/2 \quad (1)$$

$$c_2 = b_2.x_2 + const\_bit_2.q/2 \quad (2)$$

$$c_3 = b_3.x_3 + const\_bit_3.q/2 \quad (3)$$

Given that $b = A.s + e$, the function $Homo\_LWE(c)$ is as follows:

$$\begin{aligned}(A.s+e)\,x \;+\; const\_bit.q/2 = \\ (A_1s_1+e_1)x_1 + \; const\_bit_1.q/2 + \\ (A_2s_2+e_2)x_2 \; + const\_bit_2.q/2 \; + \\ (A_3s_3+e_3)x_3 + const\_bit_3.q/2\end{aligned} \quad (4)$$

$$\begin{aligned}A.s.x + e.x \; + \; const\_bit.q/2 \; = \\ A_1s_1x_1 \; + \; A_2s_2x_2 \\ + \; A_3s_3x_3 \; + e_1x_1 + e_2x_2 \\ + e_3x_3 + \; const\_bit_1.q/2 \\ +const\_bit_2.q/2 + const\_bit_3.q/2\end{aligned} \quad (5)$$

It is proved that the circuit keeps the structure of LWE after homomorphic translation since neither side of the equation has yet leaked any type of secret.

4.3. Stream Generation using LFSRs

The above discussed LWE based homomorphic function is used to produce the pseudorandom sequences.

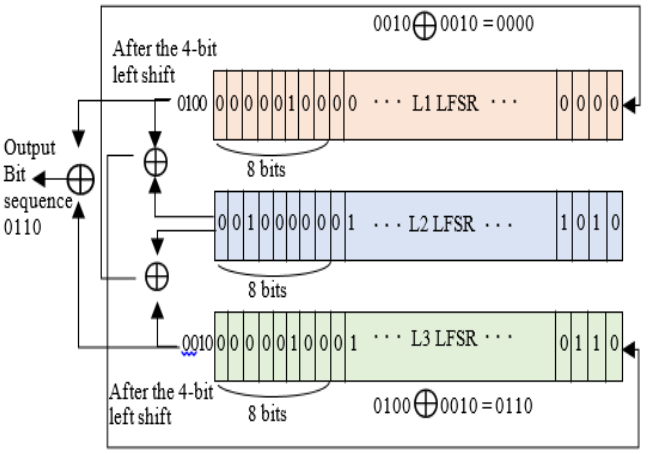

Figure 4. Linear feedback shift register.

The distributed homomorphic circuit is implemented using three LFSRs and an obfuscated seed in the proposed technique. There are three phases in the LFSR sequence generator:

1) Initializing LFSRs.
2) Generating bit output sequences.
3) Generating feedback.

In the first phase, three LFSRs, L1, L2, and L3, each with 2000 bits, are taken into consideration during design. To initialize all bits, the LWE obfuscation function creates a 6000-bit seed that is obfuscated. The non-sequential initialization of each LFSR uses the obfuscated seed. An iterative technique is utilized to initialize the LFSRs. Eight bits are successively filled in each LFSR in the first iteration. Following that, the eight bits of each LFSR are XORed to determine which pair of LFSRs has the highest XOR value. In the following iteration, a pair with the highest XOR value would be filled. Until the two LFSRs are entirely filled, the same procedure would be repeated. The remaining obfuscated seed bits would then be sequentially fed into the pending third LFSR. In this way, the initialization process of LFSRs is not sequential. In the second phase, following initialization of all three LFSRs, the output sequence would be produced in accordance with the state of the Master-Slave LFSRs. In the beginning, the scheme selects one LFSR, let's say L2, as the Master LFSR, and the other two, let's say L1 and L3, as the Slaves. Every LFSR splits its bits into eight-bit blocks at the beginning. First, the Master LFSR L2's first block is chosen, and the positions of the ones in that block are determined. For slave LFSR L1 and L3, a left shift would be carried out based on where the ones were located in a chosen block of L2. For instance, if position two in a chosen block of the Master LFSR L2 contains one, then the number of shifting bits would be calculated as 2^2=4. L1 and L3 would thus each have a four-bit left shift. The output would be the result of further XORing the four bits from the LFSR L1 and L3 together. Slave LFSRs' shifted values would be compared to one another in order to choose the master

LFSR for the following iteration. The LFSR with the highest value would take over as master for the following iteration, while the other two would act as slaves. However, the present Master LFSR would continue to function as the Master LFSR if the values of the two slave LFSRs were equal. The cycle continues eternally. In the last phase, the LFSR sequence must be fed continually to prevent exhaustion in order to generate outputs for any number of bits. The current byte and the output bits are used in this technique to give the LFSRs feedback. The two slave LFSRs' left-shifted bits from the bit output sequence generation are XORed with the same number of bits from a chosen block of the master LFSR. These XORed bits are cross-fed into the slave LFSRs. For instance, in the LFSR illustrated in Figure 4, if L2 is the master and L1, and L3 are the slaves, then the XORed bits of L1 and L2 would be sent into L3, and the XORed bits of L2 and L3 would be fed into L1.

5. Result and Analysis

The NIST STS (Statistical Test Suite) [23] is a collection of empirical tests designed to analyse bitstream randomness in accordance with a wide range of bit or block statistics. The suggested Pseudo-Random Number (0/1) Generator which is implemented in 'C' language has undergone each of the 12 NIST tests. The CDAC-PARAM Shavak machine is used to conduct these tests in the Secure Systems Lab of the SOCEMS at DIAT, Pune. To verify the generator's randomness, we buffered the 1.1 GB output bits into a file, which was then used as input to the NIST test suite. This provided 10^6 bits for every 50 samples. The selected parameters for the respective tests are mentioned in Table 1.

Table 1. Parameters selected for the respective tests.

Sl. No.	Test	Block length
1.	Block Frequency Test	128
2.	Non-Overlapping Template Test	8
3.	Overlapping Template Test	8
4.	Approximate Entropy Test	7
5.	Serial Test	10
6.	Linear Complexity Test	500

The minimum pass rate for each statistical test with the exception of the random excursion (variant) test is approximately=47 for a sample size=50 binary sequences.

The minimum pass rate for the random excursion (variant) test is approximately=21 for a sample size= 23 binary sequences.

As the proportion value for each statistical test meets the minimal criterion limit to pass the statistical tests, Table 2 clearly shows that the proposed Pseudo-Random Number Generator has passed all NIST tests.

Table 2. Testing results for the generator.

Sl. No.	Test	Statistic and reference distribution	Proportion	Accepted/ Rejected
1.	Frequency (Mono-bit) test	Half normal distribution	47/50	Accepted
2.	Frequency test within a Block	χ^2 distribution	48/50	Accepted
3.	Runs test	χ^2 distribution	48/50	Accepted
4.	Tests for the longest-run-of ones in a block	χ^2 distribution	47/50	Accepted
5.	Binary matrix rank test	χ^2 distribution	48/50	Accepted
6.	Discrete fourier transform (Spectral) Test	Normal distribution	49/50	Accepted
7.	Non-overlapping template matching test	χ^2 distribution	47/50	Accepted
8.	Overlapping template matching test	χ^2 distribution	47/50	Accepted
9.	Linear complexity test	χ^2 distribution	50/50	Accepted
10.	Cumulative sums test	Normal distribution	49/50	Accepted
11.	Random excursions test	χ^2 distribution	23/23	Accepted
12.	Random excursions variant test	Half normal distribution	23/23	Accepted

The speed of suggested QRPRNG was tested, and 35.172 Mbit/s were obtained. A software-based methodology, time stamping, is used to gauge the speed of the current QRPRNG.

6. Key Space Analysis

The collection of potential keys that can be used to generate random bits using a certain PRNG strategy is known as the key-space of a PRNG technique. The size of the key space is determined by the total amount of bits in the key, also referred to as the key length. The number of available bits is multiplied by the length of the key at each location to get the size of the key space. The proposed QRPRNG generates the LFSR's initial seed via a lattice-based hard problem. Shake256 is used as the input for the LWE encryption. Key space for the output produced by SHAKE256 is 2^{256}. As a result, the key space for the proposed generator is 2^{256}, a large number.

7. Comparison with other PRNGs

This section provides a comparative analysis of the proposed PRNG in relation to other approaches, considering randomness, quantum security, key space analysis, and performance. The proposed approach is based on a lattice-based hard problem and LFSR, enabling the generation of a random bit sequence. One significant advantage of the proposed scheme is its satisfactory statistical properties and completely random behavior. Table 3 employs criteria such as statistical test suite, speed analysis, and quantum safety

to assess PRNGs. The data clearly demonstrates that the proposed PRNG underwent the most extensive testing.

Table 3. Comparison of proposed PRNG with other PRNGs.

Features	[19]	[29]	[21]	[18]	**Proposed PRNG**
Implementation based on	PCG32 PRNG-linear congruential generator	Xoroshiro128+	Memristor + LFSR	Lattice based	Lattice based LWE +LFSR
NIST test suite	Pass	–	Pass	Pass	Pass
Speed (Mbit/s)	4-5	8-9	1.5	35.089	35.172
Quantum safe	No	No	No	✓	✓
Key space analysis	–	–	–	✓	✓
Comparison analysis	–	–	–	✓	✓

- Note: '✔'states achieved and '–' states no result reported yet.

8. Conclusions

Lattice-based primitives, notably the LWE problem, are used in this approach to attempt to secure the seed. Through the use of LFSR and a homomorphic function, it generates an indefinite long random sequence, satisfying the LWE problem's theoretical security requirement. The suggested PRNG is subjected to NIST statistical testing. As it is based on a lattice-based LWE problem, the proposed system is also resistant to quantum attacks.

Present work is focused on generators to ensure their viability as quantum resistant solution. In future, the code can be parallelized to speed up the QRPRNG. Further research is needed to optimize the efficiency and analyze the security of LWE-based PRNGs.

References

[1] Abdulsalam S., Olaniyi M., and Ahmed A., "Enhanced Tiny Encryption Algorithm for Secure Electronic Health Authentication System," *International Journal of Information Privacy, Security and Integrity*, vol. 3, no. 3, 2018. DOI:10.1504/IJIPSI.2018.10013222

[2] Albrecht M., Player R., and Scott S., "on the Concrete Hardness of Learning with Errors," *Journal of Mathematical Cryptology*, vol. 9, no. 3, pp. 169-203, 2015. DOI:10.1515/jmc-2015-0016

[3] Banerjee A., Peikert C., and Rosen A., "Pseudorandom Functions and Lattices," *in Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, UK, pp. 719-737, 2012. https://doi.org/10.1007/978-3-642-29011-4_42

[4] Becker A., Gama N., and Joux A., "Solving Shortest and Closest Vector Problems: The Decomposition Approach," *Cryptology ePrint Archive*, 2013.

[5] Bennett C. and Brassard G., "Quantum Cryptography: Public Key Distribution and Con Tos5," *Theoretical Computer Science*, vol. 560, pp. 7-11, 2014. https://doi.org/10.48550/arXiv.2003.06557

[6] Blum M. and Micali S., *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, Association for Computing Machinery, 2019. DOI:10.1145/3335741.3335751

[7] Chowdhury S. and Maitra S., "Efficient Software Implementation of Linear Feedback Shift Registers.," *in Proceedings of the 2nd International Conference on Cryptology*, Chennai, pp. 297-307, 2001. https://doi.org/10.1007/3-540-45311-3_28.

[8] Delgado-Mohatar O. and Fúster-Sabater A., "Software Implementation of Linear Feedback Shift Registers over Extended Fields," *in Proceedings of the International Joint Conference CISIS'12-ICEUTE´ 12-SOCO´ 12 Special Sessions*, Ostrava, pp. 117-126, 2013. https://doi.org/10.1007/978-3-642-33018-6_12

[9] Espinosa García J., Cotrina G., Peinado A., and Ortiz A., "Security and Efficiency of Linear Feedback Shift Registers in GF(2^n) Using n-Bit Grouped Operations," *Mathematics*, vol. 10, no. 6, 2022. https://www.mdpi.com/2227-7390/10/6/996#

[10] Ferguson N., Schneier B., and Kohno T., *Cryptography Engineering: Design Principles and Practical Applications*, Wily, 2015. https://doi.org/10.1002/9781118722367.ch9

[11] Hassan S. and Bokhari M., "Design of Pseudo Random Number Generator using Linear Feedback Shift Register," *International Journal of Engineering and Advanced Technology*, vol. 9, no. 2, 2019. DOI: 10.35940/ijeat.B2912.129219

[12] Krivenko S. and Krivenko S., "Many-to-Many Linear-Feedback Shift Register," *in Proceedings of the IEEE 34th International Scientific Conference on Electronics and Nanotechnology*, Kyiv, pp. 176-178, 2014. doi: 10.1109/ELNANO.2014.6873939

[13] Kumar A. and Mishra A., "Evaluation of Cryptographically Secure Pseudo Random Number Generators for Post Quantum Era," *in Proceedings of the IEEE 7th International Conference for Convergence in Technology*, Mumbai, pp. 1-5, 2022. doi: 10.1109/I2CT54291.2022.9824543.

[14] Laarhoven T., "Sieving for Closest Lattice Vectors (With Preprocessing)," *in Proceedings of the International Conference on Selected Areas in Cryptography*, Canada, pp. 523-542, 2017. https://doi.org/10.48550/arXiv.1607.04789

[15] L'Ecuyer P., *Random Number Generation*, Springer Berlin Heidelberg, 2012.

[16] Naor M. and Reingold O., "Constructing Pseudo-Random Permutations with A Prescribed Structure," *Journal of Cryptology*, vol. 15, pp. 97-

102, 2002. https://doi.org/10.1007/s00145-001-0008-5

[17] Nejatollahi H., Dutt N., Ray S., Regazzoni F., Banerjee I., and Cammarota R., "Post-quantum Lattice-Based Cryptography Implementations: A Survey," *ACM Computing Surveys*, vol. 51 no. 6, pp. 1-41, 2019. https://doi.org/10.1145/3292548

[18] Pandit A., Kumar A., and Mishra A., "Lwr-based Quantum-Safe Pseudo-Random Number Generator," *Journal of Information Security and Applications*, vol. 73, 2023. https://doi.org/10.1016/j.jisa.2023.103431

[19] *PCG, A Family of Better Random Number Generators | PCG, A Better Random Number Generator*. https://www.pcg-random.org/index.html.

[20] Peikert C., SVP, Gram-Schmidt, LLL. 1, pp. 1-5, 2013.

[21] Rai V., Tripathy S., and Mathew J., "Memristor Based Random Number Generator: Architectures and Evaluation," *Procedia Computer Science*, vol. 125, pp. 576-583, 2018. https://doi.org/10.1016/j.procs.2017.12.074

[22] Regev O., "CS 294.1 The Learning with Errors Problem: Introduction and Basic Cryptography Solving Systems of Linear Equations," 2018.

[23] Rukhin A., Soto J., Nechvatal J., Smid M., Barker E., and Leigh S., *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, National Institute of Standards and Technology, 2010.

[24] Shaheen S., Yousaf M., and Jalil M., "A Smart Card Oriented Secure Electronic Voting Machine Built on NTRU," *The International Arab Journal of Information Technology*, vol. 17, no. 3, pp. 386-393, 2020. doi: 10.34028/iajit/17/3/12

[25] Shrestha B., Multiprime Blum-Blum-Shub Pseudorandom Number Generator, Master Thesis, Naval Postgraduate School Monterey United States, 2016.

[26] Tang X., Xu J., and Duan B., "A Memory-efficient Simulation Method of Grover's Search Algorithm," *Computers, Materials and Continua*, vol. 57, no. 2, 2018. https://doi.org/10.32604/cmc.2018.03693

[27] Tsaban B. and Vishne U., "Efficient Linear Feedback Shift Registers with Maximal Period," *Finite Fields and Their Applications*, vol. 8, no. 2, pp. 256-267, 2002. https://doi.org/10.1006/ffta.2001.0339

[28] Ugwuishiwu C., Orji U., Ugwu C, and Asogwa, C., "An Overview of Quantum Cryptography and Shor's Algorithm," *International Journal of Advanced Trends in Computer Science* and *Engineering*, vol. 9, no. 5, 2020. https://doi.org/10.30534/ijatcse/2020/214952020

[29] *Xoroshiro128+ — RandomGen 1.23.1 Documentation*.https://bashtage.github.io/random gen/devel/bit_generators/xoroshiro128.html.

[30] Yi X., Paulet R., and Bertino E., *Homomorphic Encryption and Applications*, Springer, 2014.

Atul Kumar is working in R&D organization in India. He completed his M.Tech. In Cyber Security from Computer Science and Engineering Department at Defence Institute of Advanced Technology, Pune. His area of interest is Post Quantum Cryptography.

Arun Mishra is working as an Associate Professor in Computer Science and Engineering Department at Defence Institute of Advanced Technology, Pune. He completed his Ph.D. from MNNIT, Allahabad in 2012. He has more than 40 research publications in peer-reviewed journals & international conference proceedings. His area of interest is Cryptography, Blockchain Technology, Post Quantum Cryptography (Code-based Cryptography, Lattice-based Cryptography and Hashed based Cryptography), Secure Software Engineering, and Secure Systems Design (isolation/ secure execution/ secure computations). In the current project, he contributed to the development of the concept and technique.