Efficient Pseudo-Random Number Generator using Number-Theoretic Transform

Anupama Arjun Pandit*, Atul Kumar, and Arun Mishra

Computer Science and Engineering, Defence Institute of Advanced Technology, Pune, India
anupamapandit91@gmail.com, atulkumar.diat@gmail.com, and
arunmishra@diat.ac.in

Abstract. In the finite field of integers, the Number Theoretic Transform (NTT) is a specific variant of the Discrete Fourier Transform (DFT). NTT is the essential method that permits efficient computing. Therefore NTT could be used in the construction of lattice-based pseudo-random number generator. To construct the Lattice based Pseudo-Random Number Generator (PRNG), higher degree of polynomial multiplication is required. Therefore NTT could be used for fast multiplication as it performs point wise multiplication. In this paper, We would examine the algorithmic properties and performance of NTT for fast multiplication by implementing pseudo-random number generator using NTT.

Keywords: Number-Theoretic Transform (NTT), Pseudo-Random Number Generator (PRNG), Discrete Fourier Transform (DFT), Chinese Remainder Theorem (CRT), Lattice-based Cryptography.

1 Introduction

To generate pseudo-random numbers in the era of cutting-edge technology, an efficient pseudo-random number generator [1] is necessary. These numbers can be employed by encryption techniques [2], digital signature methods [3], and as a nonce to guarantee that various runs of the same protocol are distinct [4]. In the near future, a lattice-based [5] pseudo-random number generator will become necessary since it is resistant to quantum attacks. At present no quantum algorithm exist that can break the lattice-based algorithms, hence these generators are secure against quantum attacks [6]. Lattice-based algorithms involve higher degree of polynomial multiplication therefore school-method multiplication is not efficient as it takes $O(n^2)$ time [7]. To mitigate this inefficiency Number-Theoretic Transform (NTT) [8] could be used.

To eliminate arithmetic operations with large numbers, the Chinese Remainder Theorem (CRT) [9] is applied. CRT turns a series of coefficients into numerous series of residues, each with a distinct modulus, when a polynomial is

treated as a series of coefficients. Each series is the same length as the polynomial's degree, and the number of series is the same as the number of primes required to uniquely represent a huge integer.

NTT is a specific variant of the Discrete Fourier Transform (DFT) [11] in the finite field of integers. When converting a series of complex numbers of length n, DFT employs the powers of the n^{th} root of unity as twiddle factors ($e^{-2\pi i/n}$). NTT, on the other hand, performs modular arithmetic operations in an integer space using the powers of the n^{th} root of unity modulo a prime number as twiddle factors (γ_{2n}^{i}).

NTT's goal is to multiply two polynomials so that the coefficients of the resulting polynomials may be computed using a specific modulo. NTT has the advantage of having no precision errors because all computations are done in integers. NTT has a significant limitation in that NTT could be only accomplished with a prime modulo of the form $2^a \cdot b + 1$, where a and b are arbitrary constants. Therefore for a random mod CRT would be used.

In present work, we would compare the performance of Pseudo-Random Number Generator with and without making use of Number Theoretic Transform.

2 Preliminaries

2.1 Pseudo-Random Number Generator

As $fig.$ 1 shows, Pseudo-Random Number Generator (PRNG) [1] grabs a seed from the entity such as any process or user, performs numerous mathematical operations on the seed, and generates a random number that may be used as an initial value / seed for succeeding iteration.

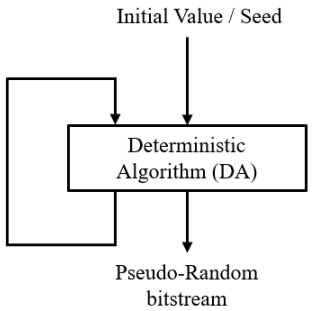

Fig. 1. Pseudo-Random Number Generator

AS PRNG contains a finite number of states, it is almost likely that its value will be mirrored at some point in the future, therefore it is also known as Pseudo RNG or Deterministic RNG.

2.2 Chinese Remainder Theorem

Suppose $q_1, q_2, \ldots, q_k$ are positive integers which are pairwise relatively prime and greater than one. Suppose $z_1, z_2, \ldots, z_k$ are any integers. Then there exists an integer solution z such that $z \equiv z_j (mod\ q_j)$ for each $j = 1, 2, \ldots, k$ [10].

$$z \equiv z_1 (mod\ q_1)$$

$$z \equiv z_2 (mod\ q_2)$$

$$z \equiv z_3 (mod\ q_3)$$

$$\vdots$$

$$z \equiv z_k (mod\ q_k)$$

The ability of Chinese Remainder Theorem (CRT) to build a residue number system is one of its most appealing features. That is, operations on the "small" values z_j are equivalent to operations on the "large" value z. The CRT, for example, could be designed to optimise $N = p \cdot q$ in RSA algorithm. All computations could be performed on the smaller z_j values instead of the large z values specifically by setting $z_1 \equiv z\ mod\ p$ and $z_2 \equiv z\ mod\ q$.

2.3 Discrete Fourier Transform

Any signal is decomposed into a sum of complex sine and cosine waves by using the Fourier Transform. The Discrete Fourier Transform (DFT) [11] is a function that enables switching from one domain to another, and the Fast Fourier Transform (FFT) is an algorithm for computing the DFT in efficient manner. Assume the input polynomial is $Z(x) = z_0 + z_1 x + \cdots + z_{n-1} x^{n-1}$, then evaluate Z at all n^{th} roots of unity. The divide-and-conquer strategy would be employed.

To begin, the polynomial Z may decompose into Z_{even} and Z_{odd}, where $Z_{even}(x) = z_0 + z_2 x + \cdots$, where $Z_{odd}(x) = z_1 + z_3 x + \cdots$. Then $Z(x)$ would be $Z_{even}(x) + Z_{odd}(x)$. Z_{even} and Z_{odd} could be computed recursively at all the $n^{th}/2$ root of unity to evaluate the polynomial Z. The $n^{th}/2$ root of unity are $1, \zeta, \zeta^2, \ldots, \zeta^{\frac{n}{2}-1}$ where $\zeta = \omega^2$. The steps in $FFT(Z, n)$ are as follows:

i) *Calculate* Z_{even}, Z_{odd}.
ii) *Calculate* $FFT(Z_{even}, n/2)$, $FFT(Z_{odd}, n/2)$
iii) *For* $0 \leq k < \frac{n}{2}$: $Z(\omega^k) = Z_{even}(\omega^{2k}) + \omega^k \cdot Z_{odd}(\omega^{2k}) = Z_{even}(\zeta^k) + \omega^k \cdot Z_{odd}(\zeta^k)$

iv) $For\ \frac{n}{2} \leq k < n : Z(\omega^k) = Z_{even}(\omega^{2k}) + \omega^k \cdot Z_{odd}(\omega^{2k}) = Z_{even}(\zeta^{k-\frac{n}{2}}) + \omega^k \cdot Z_{odd}(\zeta^{k-\frac{n}{2}})$

The evaluations of Z at $Z(1), Z(\omega), \cdots, Z(\omega^{n-1})$ are used to compute the unique polynomial Z are as follows in the form of matrix equation:

$$\begin{bmatrix} Z(1) \\ Z(\omega) \\ \vdots \\ Z(\omega^{n-1}) \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \dots & \omega^{(n-1)(n-1)} \end{bmatrix} = \begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ z_{n-1} \end{bmatrix}$$

where the $n \times n$ matrix is signified as the Fourier matrix M. The inverse of the matrix M would be required to compute Z from the evaluation. The following is the inverse of the matrix M.

$$M^{-1} = \frac{1}{n} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \dots & \omega^{-(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega^{-(n-1)} & \omega^{-2(n-1)} & \dots & \omega^{-(n-1)(n-1)} \end{bmatrix}$$

DFT has an order of time complexity of $O(n^2)$, whereas FFT has an order of $O(n \log n)$. The most popular Fast Fourier Transform (FFT) algorithm is the $Cooley–Tukey$ algorithm [12]. It recursively defines the DFT of a random composite size $N = N_1 \cdot N_2$ in terms of N_1 smaller $DFTs$ of sizes N_2 to reduce the time complexity for highly composite N to $O(N \log N)$. Another FFT algorithm is the Gentleman-Sande [13] algorithm.

2.4 Twiddle Factor

Any of the trigonometric constant coefficients multiplied by the data in the Fast Fourier Transform (FFT) process is referred to as a twiddle factor [14]. Twiddle factors are expressed mathematically as: $\omega_N^n = e^{-i2\pi n/N} = cos(2\pi n/N) - i\ sin(2\pi n/N)$, where $n = 0, 1, 2 \cdots, N-1$. The twiddle factor is a rotating vector quantity that rotates in steps of N samples. The twiddle factor is used in DFT and Inverse DFT (IDFT) computations to minimise computational complexity. Because the twiddle factor has cyclic features and is periodic, the twiddle factor's values repeat every $N-$ cycles.

2.5 Lattice-Based Cryptography

The security of lattice based cryptographic systems is derived from gauged contumacy of Lattice problems like Shortest Vector Problem (SVP) [15].
A Lattice [16] is a collection of uniformly spaced grid of points in n-dimensional

space. In other words, lattice is a set of vectors generated by n-linearly independent vectors $b_1, b_2, \cdots, b_n \in R^{\mathrm{n}}$.

$$\mathcal{L}(b_1, b_2, \cdots, b_n) = \left\{ \sum_{i=1}^{n} x_i b_i : x_i \in \mathbb{Z} \right\}$$

Here $b_1, b_2, \cdots, b_n$ are basis vectors of the lattice.

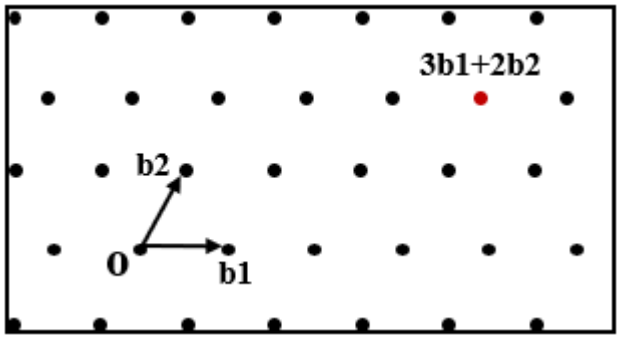

Fig. 2. Lattice representation by two possible bases

In lattice based cryptography, Vector is another decorative name for a point and a tuple of numbers are called the coordinates of the vector whereas a basis is a linearly independent collection of vectors that can be used to generate any point in the uniformly spaced grid of the lattice. Short bases are more practical at the time of solving hard lattice problems such as Shortest Vector Problem (SVP) [15], Closest Vector Problem (CVP)[17] and Shortest Independent Vector Problem (SIVP) [18].

- Shortest Vector Problem: Determine the shortest vector u closest to the origin for a lattice $\mathcal{L}$. In other words, find out a vector u for a given basis $B = b_1, b_2, \cdots, b_n \in Z^{n \times n}$ which satisfies the following condition:

$$||u|| \;=\; \lambda_1(\mathcal{L}(B)) \;\; where \;\; u \in \mathcal{L}(B) \backslash \{0\}$$

- Closest Vector Problem: Determine the vector u closest to the given point t which is not a lattice point for a lattice $\mathcal{L}$. In other words, find out a vector u to a given vector $t \in Z^n$ for a given basis $B = b_1, b_2, \cdots, b_n \in Z^{n \times n}$ which satisfies the following condition:

$$||u - t|| \;=\; dist(\mathcal{L}(B), t) \;\; where \;\; u \in \mathcal{L}(B)$$

- Shortest Independent Vector Problem (SIVP): Determine the set of linearly independent vectors $u_1, u_2, \cdots, u_n$ for a lattice $\mathcal{L}$, and it should also be short vectors. In other words, find out a set of linearly independent vector

$u_1, u_2, \cdots, u_n$ for a given basis $B = b_1, b_2, \cdots, b_n \in Z^{n \times n}$ which satisfies the following condition:

$$||u_i|| \ \leq \ \lambda_n \ \ where \ \ u \in \mathcal{L}(B) \ \ for \ \ i \in [n]$$

Lattice-based Cryptography [19] is assumed to be resistant against quantum attacks as it uses very complex large dimensional geometric structures. The security of a lattice-based cryptosystem is derived from the hardness of lattice problems. Learning with Errors (LWE)[20] and Learning with Rounding (LWR) [21] are the most advanced Lattice-based cryptosystems. The variant of LWE is Ring Learning With Errors (RLWE). LWE could be used for the construction of a Non-Deterministic Pseudo-Random Number Generator. Similarly, LWR could be used for the construction of a Deterministic Pseudo-Random Number Generator.

– Learning with Errors (LWE): LWE [20] problem is to find secret vector s by solving given noisy linear equations. If the error vector denoted as $e = e_1, e_2, \cdots, e_m$ were not introduced in the following system of linear equations then it could be rewritten as $s = A^{-1}b$ and could be easily solved by using Gauss Elimination method where A is a matrix such that $A = a_{11} \cdots, a_{mn}$, secret vector $s = s_1, s_2, \cdots, s_n$, $b = b_1, b_2, \cdots, b_m$, q is prime modulus. Since errors are augmented in the following equations and number of equations are too large therefore it is not possible to solve these equations in polynomial time. This reflects the hardness of this problem. Since there is no quantum algorithm exist which could solve these equations in polynomial time therefore these LWE algorithm is quantum-safe[21].

$$\begin{gathered} a_{11} \cdot s_1 + a_{12} \cdot s_2 + ... \ a_{1n} \cdot s_n + e_1 = b_1 \ mod \ q \\ a_{21} \cdot s_1 + a_{22} \cdot s_2 + ... \ a_{2n} \cdot s_n + e_2 = b_2 \ mod \ q \\ \vdots \\ a_{m1} \cdot s_1 + a_{m2} \cdot s_2 + ... \ a_{mn} \cdot s_n + e_m = b_m \ mod \ q \end{gathered}$$

The steps of LWE algorithm are as follows:

i) Key Generation:

$$public \ key = \{A, b = s \cdot A + e\} \quad \& \quad secret \ key = s$$

ii) Encryption:

$$Enc(public \ key, bit) = \{cipher \ text \ preamble(u) = A \cdot x, \ cipher \ text(u') = b \cdot x + bit \cdot \lfloor \tfrac{q}{2} \rfloor\}, \ where \ x \in \{0,1\}^n$$

iii) Decryption:

$$Dec(secret \ key, (u, u')) = \begin{cases} 0 & if \ u' - s \cdot u \ mod \ q \in [\frac{-q}{4}, \frac{q}{4}] \\ 1 & if \ u' - s \cdot u \ mod \ q \in (\frac{q}{4}, \frac{3q}{4}] \end{cases}$$

3 Proposed Methodology to generate Efficient Pseudo-Random Number Generator

To construct a series of random numbers, a PRNG requires a cryptographically secure initial value or seed. If it is not secure, an attacker may produce the whole random number sequence since the output of the PRNG might be determined only by the seed. It is not required for the seed to be unique, but if it is used again, there is a risk of attack.

The following are two important aspects of seed:

i) Seed should be protected as cryptographic content (eg. key)
ii) Use a cryptographically secure source to generate the seed

A random number generator might be used to distribute the RNG seeds for each operation to eliminate duplicate seeds. The Learning with Rounding (LWR) [21] method could be used for this. Its implementation is also highly efficient, as it is built on lattice-based cryptography, which is effective in the post-quantum era and consists of very solid security proofs based on worst-case hardness. It is also considered to be quantum-resistant because no attack on lattice-based cryptography has been discovered to far.

– Learning with Rounding (LWR): The derandomised version of LWE is called LWR. In LWR, a slight error with $A \cdot s \in Z_q$ may be used to mask their true value, which would then be multiplied by p/q for some $p < q$. If $q > p$ and $A \cdot s$ is a deterministic rounded version, the LWR problem would be hard. As a result, in this example, $b = \lfloor A \cdot s \rceil_p = \lfloor \frac{p}{q} \cdot A \cdot s \rceil_p$ For the implementation aspect, we usually take the floor of this value. In this case, result would be decreased from mod q to mod p. Other operations, such as encryption and decryption, will stay unchanged. LWR is at least as hard as LWE for the relevant parameters, and it provides a worst-case assurance for LWR [21]. The advantage of LWR is that it eliminates the heavy error sampling process, requiring fewer random bits. Because of its predictable nature, LWR-based cryptosystems could be employed in symmetric cryptography.

For our PRNG, we employ the LWR one-way function to construct a cipher LWR vector that cannot be reversed without the secret information. As LWR is using huge multiplication such that $A \cdot s$ in Key Generation step, NTT could be used to speedup the multiplication operation.

Suppose $\bar{x_i} = x_i \gamma_{2N}^i, \bar{y_i} = y_i \gamma_{2N}^i$, and $\bar{z_i} = z_i \gamma_{2N}^i$ where $\gamma_{2N} = \sqrt{\omega_N}$. To compute $z = x \cdot y$ over $\mathbb{Z}_q[x]/\langle x^N + 1 \rangle$, Negative Wrapped Convolution (NWC) technique [22] is performed as:

$$\bar{z} = INTT_N(NTT_N(\bar{x}) \bigodot NTT_N(\bar{y}))$$

where the symbol $\bigodot$ denotes point-wise multiplication. If $q \equiv 1\ (mod\ 2N)$ is satisfied by the prime q, then ω_N and its square root γ_{2N} exist, as per the NWC technique. This approach, however, has a "scramble" that includes pre-processing before NTT and post-processing after Inverse NTT (INTT), as discussed below. The scaled vectors $\bar{x}$ and $\bar{y}$ are subjected to classic N-point NTT; the scaled vector $\bar{z}$ is produced after the classic N-point INTT; and the final result z may be retrieved by computing $z_i = \gamma_{2N}^{-i}$. The NWC technique [22] eliminates the explicit reduction and doubling of NTT / INTT, but it does need the coefficients to be scaled with γ_{2N}^{i} before NTT and the results to be scaled with γ_{2N}^{-i} after INTT.

The NTT algorithm without pre-processing [23] is as follows:
Suppose the vectors v and V indicate $v_0, v_1, \cdots, v_{N-1}$ and $(V_0, V_1, \cdots, V_{N-1})$ respectively, where $v_i \in \mathbb{Z}_q, V_i \in \mathbb{Z}_q, i = 1, 2, 3, \cdots, N-1$. Suppose ω_N be a primitive N^{th} root of unity in $\mathbb{Z}_q$ and $\gamma_{2N} = \sqrt{\omega_N}$.
Input : v, N, q, $\gamma_{2N}^{i}, i = 0, 1, 2, \cdots N-1$
Output : $V = NTT(v)$

i) $V \leftarrow scramble(v)$
ii) $for\ l = 1\ to\ \log_2 N\ do$
iii) $\quad r \leftarrow 2^l$
iv) $\quad for\ m = 0\ to\ r/2 - 1\ do$
v) $\quad\quad \omega = \gamma_{2N}^{(2m+1)N/r}$
vi) $\quad\quad for\ n = 0\ to\ N/r - 1\ do$
vii) $\quad\quad\quad f = V_{nr+m}$
viii) $\quad\quad\quad g = \omega \cdot V_{nr+m+r/2}\ mod\ q$
ix) $\quad\quad\quad V_{nr+m} = (f + g)\ mod\ q$
x) $\quad\quad\quad V_{nr+m+r/2} = (f - g)\ mod\ q$
xi) $\quad\quad end\ for$
xii) $\quad end\ for$
xiii) $end\ for$
xiv) $return\ V$

The Inverse NTT (INTT) algorithm without post-processing [23] is as follows:
Suppose the vectors v and V indicate $v_0, v_1, \cdots, v_{N-1}$ and $(V_0, V_1, \cdots, V_{N-1})$ respectively, where $v_i \in \mathbb{Z}_q, V_i \in \mathbb{Z}_q, i = 1, 2, 3, \cdots, N-1$. Suppose ω_N be a primitive N^{th} root of unity in $\mathbb{Z}_q$ and $\gamma_{2N} = \sqrt{\omega_N}$.
Input : v, N, q, $\gamma_{2N}^{-i}, i = 0, 1, 2, \cdots N-1$
Output : $V = INTT(v)$

i) $for\ l = \log_2 N\ to\ 1\ do$
ii) $\quad r \leftarrow 2^l$
iii) $\quad for\ m = 0\ to\ r/2 - 1\ do$

iv) $\omega = \gamma_{2N}^{-(2m+1)N/r}$
v) $for\ n = 0\ to\ N/r - 1\ do$
vi) $f = V_{nr+m}$
vii) $g = V_{nr+m+r/2}\ mod\ q$
viii) $V_{nr+m} = \frac{(f+g)}{2}\ mod\ q$
ix) $V_{nr+m+r/2} = \frac{(f-g)}{2}\ \cdot \omega\ mod\ q$
x) $end\ for$
xi) $end\ for$
xii) $end\ for$
xiii) $V \leftarrow scramble(v)$
xiv) $return\ V$

The pre-processing denotes the coefficient-wise multiplications of v_i and γ_{2N}^{-i} before NTT. The post-processing denotes the final scaling by N^{-1} in the classic INTT and the coefficient-wise multiplication by γ_{2N}^{-i} after the classic INTT.

4 Implementation

The flow chart of the Pseudo-Random Number Generator is shown in *Fig.* 3. The key Generation step of the Learning with Rounding (LWR) algorithm takes 128−bits as an initial value in a random manner and it generates the public key. Now encryption step of LWR uses a public key and encrypts the input bit to produce a cipher vector of length 256. The binary form of generated cipher vector is used as a secure seed by the Linear Feedback Shift Register (LFSR) [24]. A pseudo-random number stream is now generated by the LFSR.

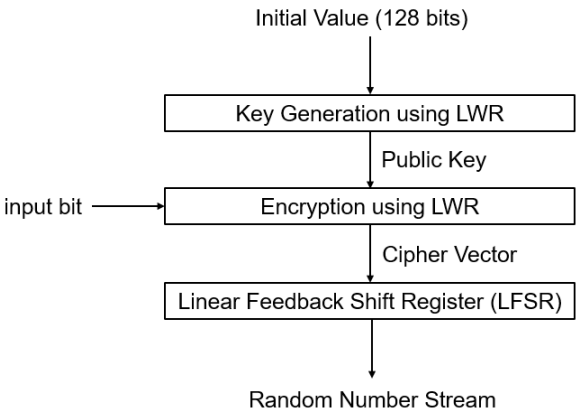

Fig. 3. Flow Chart

We have implemented PRNG in 'C' language using Visual Studio 2019 IDE. As we could see in Key generation step of LWE, we are multiplying matrix A and secret vector s such that $s \cdot A$. Similarly in Encryption step we are doing matrix-vector multiplication such that $A \cdot x$ and vector-vector multiplication such that $b \cdot x$. Since the construction of PRNG requires higher degree polynomial's multiplication and we are using polynomial of degree-256 therefore we have used NTT for fast multiplication.

5 Result

We keep splitting given two polynomials until we only have degree-0 polynomials remaining. After completing the transformation, we may perform all operations on the polynomials "pointwise," which has an $O(n)$ complexity. We recursed along a tree with $\log_2 n$ "layers," each of which performed n multiplications with some twiddle factor to change the polynomials. As a result, the transformation is $O(n\ log\ n)$ in terms of its complexity.

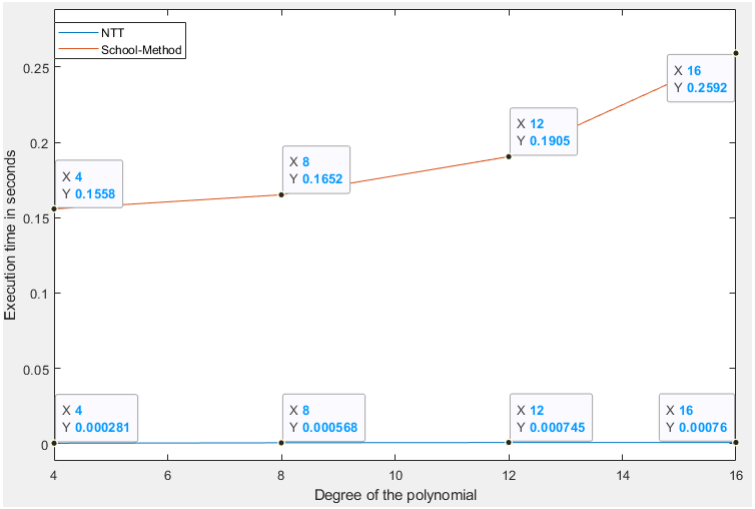

Fig. 4. Execution time of polynomial multiplication with variant degrees

Fig. 4 is showing the execution time of polynomial multiplication of degrees 4, 8, 12, and 16 respectively. We could easily observed that polynomial multiplication using NTT is quite fast as compare to polynomial multiplication using school-method. Therefore, NTT could play the important role in construction of efficient pseudo-random number generator.

6 Conclusion

In this paper, we have discussed about Quantum-Safe Pseudo-Random Number Generator. We have found that there is no algorithm exist classical as well as quantum which could break the lattice-based Learning with Rounding scheme therefore it is quantum-safe. We also discussed that NTT could be used for fast multiplication as NTT performed multiplication in $O(n \log n)$ time while school-method perform same multiplication in $O(n^2)$ time. Hence, we proposed that NTT could be used in construction of efficient pseudo-random number generator. Furthermore, in the future, multiplication by using NTT could be enhanced by identifying concurrent processes for parallel execution.

References

1. Cang, S., Kang, Z., & Wang, Z. (2021). Pseudo-random number generator based on a generalized conservative Sprott-A system. Nonlinear Dynamics, 104(1), 827-844.
2. Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2), 120-126.
3. Kuppuswamy, P., Appa, P. M., & Al-Khalidi, D. S. Q. (2012). A New Efficient Digital Signature Scheme Algorithm based on Block cipher. IOSR Journal of Computer Engineering (IOSRJCE), 7(1), 47-52.
4. Køien, G. M. (2015). A brief survey of nonces and nonce usage. In SECURWARE International Conference on Emerging Security Information, Systems and Technologies.
5. Chi, D. P., Choi, J. W., San Kim, J., & Kim, T. (2015). Lattice based cryptography for beginners. Cryptology ePrint Archive.
6. Regev, O. (2010). The learning with errors problem. Invited survey in CCC, 7(30), 11.
7. İlter, M. B., & Murat, C. E. N. K. (2017). Efficient big integer multiplication in cryptography. International Journal of Information Security Science, 6(4), 70-78.
8. Creutzburg, R., & Tasche, M. (1986). Number-theoretic transforms of prescribed length. Mathematics of computation, 47(176), 693-701.
9. Piazza, N. (2018). The Chinese Remainder Theorem.
10. Alhassan, E. A., Tian, K., Abban, O. J., Ohiemi, I. E., Adjabui, M., Armah, G., & Agyemang, S. (2021). On Some Algebraic Properties of the Chinese Remainder Theorem with Applications to Real Life.
11. Sandeep, S. (2020). Fast Integer Multiplication. 1–3.
12. Cooley, J. W., & Tukey, J. W. (1965). An algorithm for the machine calculation of complex Fourier series. Mathematics of computation, 19(90), 297-301.
13. Schupp, S. (2003). Lifting a butterfly–A component-based FFT. Scientific Programming, 11(4), 291-307.
14. Yamini Gayathri T. (2017). An Efficient Multi Precision Floating Point Complex Multiplier Unit in FFT. International Journal of Engineering Research And, V6(06). https://doi.org/10.17577/ijertv6is060395
15. Peikert, C. (2013). SVP, Gram-Schmidt, LLL. 1, 1–5
16. Micciancio, D., Regev, O. (2009). Lattice-based Cryptography. Post Quantum Cryptography, 015848, 147–191. https://doi.org/10.1007/978-3-540-88702-7 5

17. Daniele Micciancio, University of California, S. D. (2003). Closest Vector Problem. https://doi.org/10.1007/978-1-4615-0897-7 3
18. Micciancio, D. (2008). Efficient reductions among lattice problems. Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms, 84–93.
19. Chi, D. P., Choi, J. W., Kim, J. S., Kim, T. (2015). Lattice Based Cryptography for Beginners. EPrint
20. Regev, O. (2009). On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM, 56(6), 1–37. https://doi.org/10.1145/1568318.1568324
21. Banerjee, A., Peikert, C., & Rosen, A. (2012, April). Pseudorandom functions and lattices. In Annual International Conference on the Theory and Applications of Cryptographic Techniques (pp. 719-737). Springer, Berlin, Heidelberg.
22. Chen, D. D., Mentens, N., Vercauteren, F., Roy, S. S., Cheung, R. C., Pao, D., & Verbauwhede, I. (2014). High-speed polynomial multiplication architecture for ring-LWE and SHE cryptosystems. IEEE Transactions on Circuits and Systems I: Regular Papers, 62(1), 157-166.
23. Zhang, N., Yang, B., Chen, C., Yin, S., Wei, S., & Liu, L. (2020). Highly efficient architecture of NewHope-NIST on FPGA using low-complexity NTT/INTT. IACR Transactions on Cryptographic Hardware and Embedded Systems, 49-72.
24. Hassan, S., & Bokhari, M. U. (2019). Design of Pseudo Random Number Generator using Linear Feedback Shift Register. International Journal of Engineering and Advanced Technology.