Introduction to Number Theoretic Transform

Banhirup Sengupta[1,2], Peenal Gupta[2], and Souvik Sengupta[3]

[1]Center for Applicable Mathematics, Tata Institute Of Fundamental Research, Bangalore, India
[2]Research Group, PinakashieldTech OÜ, Tallinn, Estonia
[3]Digital Ecosystems, IONOS SE, Karlsruhe, Germany

Abstract

The Number Theoretic Transform (NTT) can be regarded as a variant of the Discrete Fourier Transform. NTT has been quite a powerful mathematical tool in developing Post-Quantum Cryptography and Homomorphic Encryption. The Fourier Transform essentially decomposes a signal into its frequencies. They are traditionally sine or cosine waves. NTT works more over groups or finite fields rather than on a continuous signal and polynomials work as the analog of sine waves in case of NTT. Fast Fourier Trnasform (FFT) style NTT or fast NTT has been proven to be useful in lattice-based cryptography due to its ability to reduce the complexity of polynomial multiplication from quadratic to quasilinear. We have introduced the concepts of cyclic, negacyclic convolutions along with NTT and its inverse and their fast versions.

1 Introduction

Lattice-based cryptography has emerged as a promising candidate for public-key cryptography which is quantum safe in nature. Many lattice-based schemes are based on operations in the ring of polynomials of the form $(\mathbb{Z}/q\mathbb{Z})[X]/(f(X))$, where f is an irreducible polynomial over $\mathbb{Z}$ and q is a prime number. Those schemes, whose security heavily depends on the hardness of the Ring-LWE problem, f is usually chosen to be a exponent of 2 cyclotomic $X^n + 1$, whose roots have order $2n$ and q satisfies the condition $q \equiv 1 \mod 2n$. The motivation for choosing exponent of 2 cyclotomic rings and fully splitting primes is that the splitting behavior of such primes in these rings allows for swift multiplication using FFT, which is called NTT when it is performed over the base field $\mathbb{Z}_q$. NTT transforms polynomials from the time domain to a frequency (NTT) domain, allowing for coefficient-wise multiplication. Then the inverse NTT is used to transform the output back to the time domain, thereby providing a speed-up for polynomial multiplications. An advantage of NTT-based multiplication over other methods is that NTTs can be saved by directly sampling polynomials in the NTT domain, by storing

NTT domain representations of polynomials for later use, and making use of the linearity of NTT when computing sums of products of polynomials as well. This helps in the speed-up as multiplication is one of the most time consuming operation in lattice-based signature schemes such as Dilithium and Falcon.

This note is mostly taken from [1], [2], and organized as follows. In Section 2, we give the definitions of cyclic (positive-wrapped) and negacyclic (negative-wrapped) convolutions. Section 3 deals with the positive-wrapped convolution based on NTT, whereas Section 4 is responsible for negative-wrapped convolution. In Section 5, we have discussed about the Cooley-Tukey and Gentleman-Sande algorithms for Fast-NTT and Fast-INTT respectively. Finally, NTT based multiplication has been described in Section 6 along with an example of Dilithium signature scheme.

2 Cyclic and Negacyclic Convolution

Definition 1. *Let $G(x)$ and $H(x)$ be polynomials of degree $n-1$ in the quotient ring $\mathbb{Z}_q[x]/(x^n-1)$, where $q \in \mathbb{Z}$. A cyclic convolution or positively wrapped convolution, $PWC(x)$ is defined as:*

$$PWC(x) = \sum_{k=0}^{n-1} c_k x^k$$

where $c_k = \sum_{i=0}^{k} g_i h_{k-i} + \sum_{i=k+1}^{n-1} g_i h_{k+n-i} \mod q$. If $Y(x)$ is the result of their linear convolution in the ring $\mathbb{Z}_q[x]$, it can be defined as well as :

$$PWC(x) = Y(x) \mod (x^n - 1).$$

Negacyclic convolution is exactly the same type of convolution, modulo the divisor. The cyclic convolution uses $x^n - 1$, while negacyclic convolution uses $x^n + 1$. Both these convolution techniques have $O(n^2)$ complexity.

3 Positive-Wrapped Convolution based on NTT

This section describes NTT and its inverse (INTT) based on the n-th root of unity, ω.

Definition 2. *Let $\mathbb{Z}_q$ be an integer ring modulo q, and $n-1$ is the polynomial degree of $G(x)$ and $H(x)$. We define ω as the primitive n-th root of unity in $\mathbb{Z}_q$ iff :*

$$\omega^n \equiv 1 \mod q$$

and

$$\omega^k \not\equiv 1 \mod q$$

for $k < n$.

One must note that the primitive n-th root of unity in a ring $\mathbb{Z}_q$ might not be unique. For example, in the ring $\mathbb{Z}_{7681}$, $\omega = 3383$ and $\omega = 4298$ are the primitive 4-th roots of unity.

Definition 3. *The Number Theoretic Transform (NTT) of a vector with polynomial coefficients* $\mathbf{v}$ *is defined as* $\hat{\mathbf{v}} = NTT(\mathbf{v})$*, where*

$$\hat{\mathbf{v}}_j = \sum_{i=0}^{n-1} \omega^{ij}\mathbf{v}_i \mod q,$$

$j = 0, 1, \ldots, n-1.$

Definition 4. *The inverse of an NTT vector* $\hat{\mathbf{v}}$ *is defined as* $\mathbf{v} = INTT(\hat{\mathbf{v}})$*, where*

$$\mathbf{v}_i = \frac{1}{n}\sum_{j=0}^{n-1} \omega^{-ij}\hat{\mathbf{v}}_j \mod q,$$

$j = 0, 1, \ldots, n-1.$

The only difference between NTT and INTT is ω replaced by its inverse in $\mathbb{Z}_q$ and a weighted product. It should be noted that $\mathbf{v} = INTT\,(NTT(\mathbf{v}))$. Since NTT is a variant of Discrete Fourier Transform in polynomial ring, one can use the convolution theorem of DFT to calculate positive wrapped convolution, [3], [4].

Proposition 1. *Let* $\mathbf{u}$ *and* $\mathbf{v}$ *be the multiplicands vectors with polynomial coefficients. The positive-wrapped convolution of* $\mathbf{u}$ *and* $\mathbf{v}$ *is*

$$\mathbf{w} = INTT\,(NTT(\mathbf{u}) \circ NTT(\mathbf{v}))\,, \tag{1}$$

where $\circ$ *is an element-wise vector multiplication in* $\mathbb{Z}_q$*.*

4 Negative-Wrapped Convolution based on NTT

The scope of implementation of positive-wrapped convolution or cyclic convolution is primarily outside the cryptography domain. For example, it is used in Schönhage-Strassen algorithm [5] for large integer multiplication. However, in the context of Post-Quantum Cryptography and Homomorphic Encryption, the chosen ring is mostly $\mathbb{Z}_q[x]/(x^n+1)$ instead of $\mathbb{Z}_q[x]/(x^n-1)$. So, we should calculate the polynomial multiplications using negative-wrapped or negacyclic convolution in such rings.
Next, let us define the $2n$-th root of unity which is essential to calculate negacyclic convolution.

Definition 5. *Let* $\mathbb{Z}_q$ *be an integer ring modulo* q*,* $G(x)$ *and* $H(x)$ *be* $n-1$ *degree polynomials, and* ω *is its primitive* n*-th root of unity. We define* ψ *as the primitive* $2n$*-th root of unity iff :*

$$\psi^2 \equiv \omega \mod q$$

and

$$\psi^n \equiv -1 \mod q.$$

For example, in a ring $\mathbb{Z}_{7681}$ and $n = 4$, when $\omega = 3383$, value of ψ can be either 1925 or 5756.

Definition 6. *The Negative-Wrapped Number Theoretic Transform of a vector with polynomial coefficients* $\mathbf{v}$ *is defined as* $\hat{\mathbf{v}} = NTT^{\psi}(\mathbf{v})$*, where*

$$\hat{\mathbf{v}}_j = \sum_{i=0}^{n-1} \psi^i \omega^{ij} \mathbf{v}_i \mod q,$$

$j = 0, 1, \ldots, n-1$. *Since* $\psi^2 \equiv \omega \mod q$*, one can substitute* $\omega = \psi^2$ *above :*

$$\hat{\mathbf{v}}_j = \sum_{i=0}^{n-1} \psi^{2ij+i} \mathbf{v}_i \mod q. \tag{2}$$

Definition 7. *Negative-Wrapped inverse of an NTT vector* $\hat{\mathbf{v}}$ *is defined as* $\mathbf{v} = INTT^{\psi^{-1}}(\hat{\mathbf{v}})$ *:*

$$\mathbf{v}_i = \frac{1}{n} \sum_{j=0}^{n-1} \psi^{-j} \omega^{-ij} \hat{\mathbf{v}}_j \mod q,$$

$j = 0, 1, \ldots, n-1$. *Substituting* $\omega = \psi^2$*, we get*

$$\mathbf{v}_i = \frac{1}{n} \sum_{j=0}^{n-1} \psi^{-(2ij+j)} \hat{\mathbf{v}}_j \mod q. \tag{3}$$

Proposition 2. *Let* $\mathbf{u}$ *and* $\mathbf{v}$ *be the multiplicands vectors with polynomial coefficients. The negative-wrapped convolution of* $\mathbf{u}$ *and* $\mathbf{v}$ *is*

$$\mathbf{w} = INTT^{\psi^{-1}} \left(NTT^{\psi}(\mathbf{u}) \circ NTT^{\psi}(\mathbf{v}) \right), \tag{4}$$

where $\circ$ *is an element-wise vector multiplication in* $\mathbb{Z}_q$*.*

The modulus q needs to satisfy the following to make NTT transformation possible :

- Then n-th root of unity ω exists in the ring $\mathbb{Z}_q$, so that one can perform positive-wrapped convolutions.
- The $2n$-th root of unity ψ exists in the ring $\mathbb{Z}_q$ to make negative-wrapped convolutions work. The modulus q has to satisfy the following theorems to make sure that ω and ψ exist respectively.

The modulus q has to satisfy the following theorem to ensure the existence of ω [3], [6], [7] :

Theorem 3. *If* q *is prime, then* n *must* $q-1$*. If* q *is composite such that :*

$$q = q_1^{m_1} \cdot q_2^{m_2} \cdots q_k^{m_k}$$

then n *must divide the GCD of* $(q_1 - 1, q_2 - 1, \cdots, q_k - 1)$*.*

However, the preceding theorem does not guarantee the existence of ψ in $\mathbb{Z}_q$. The next theorem will ensure that ψ exists in $\mathbb{Z}_q$.

Theorem 4. *If* q *is prime, then* $2n$ *must* $q-1$*. If* q *is composite such that :*

$$q = q_1^{m_1} \cdot q_2^{m_2} \cdots q_k^{m_k}$$

then $2n$ *must divide the GCD of* $(q_1 - 1, q_2 - 1, \cdots, q_k - 1)$*.*

Definition 8. *A PWC-NTT friendly modulus* q *is defined iff an* n*-th root of unity,* ω *exists in* $\mathbb{Z}_q$*.*

Definition 9. *A NWC-NTT friendly modulus* q *is defined iff an* n*-th root of unity,* ω *and* $2n$*-th root of unity* ψ *both exist in* $\mathbb{Z}_q$*.*

5 FFT-style NTT

Usually NTT has $O(n^2)$ complexity, thereby making no difference from that of negacyclic convolution. However, NTT is Discrete Fourier transform in ring of polynomials. So, DFT optimization techniques can be applied to NTT as well. The popular technique of DFT optimization is called Fast Fourier transform. It was proposed independently by Cooley-Tukey [8] and Gentleman-Sande [9]. Both of these methods use similar butterflies divide-and-conquer technique to achieve the quasilinear complexity $O(n \log n)$. One can use divide-and-conquer techniques to fasten the process of matrix multiplication needed for NTT by utilizing the periodicity and symmetry property of ψ :

$$\psi^{k+2n} = \psi^k$$

and

$$\psi^{k+n} = -\psi^k.$$

5.1 Cooley-Tukey (CT) Algorithm for Fast-NTT :

The summation in equation (2) can be separated into two parts :

$$\begin{aligned}
\hat{\mathbf{v}}_j &= \sum_{i=0}^{n-1} \psi^{2ij+i}\mathbf{v}_i \mod q \\
&= \sum_{i=0}^{n/2-1} \psi^{4ij+2i}\mathbf{v}_{2i} + \sum_{i=0}^{n/2-1} \psi^{4ij+2j+2i+1}\mathbf{v}_{2i+1} \mod q \\
&= \sum_{i=0}^{n/2-1} \psi^{4ij+2i}\mathbf{v}_{2i} + \psi^{2ij+1}\sum_{i=0}^{n/2-1} \psi^{4ij+2i}\mathbf{v}_{2i+1} \mod q
\end{aligned}$$

Using ψ's symmetry properties :

$$\hat{\mathbf{v}}_{j+n/2} = \sum_{i=0}^{n/2-1} \psi^{4ij+2i}\mathbf{v}_{2i} - \psi^{2ij+1}\sum_{i=0}^{n/2-1} \psi^{4ij+2i}\mathbf{v}_{2i+1} \mod q$$

Let $A_j = \sum_{i=0}^{\frac{n}{2}-1} \psi^{4ij+2i}\mathbf{v}_{2i}$ and $B_j = \sum_{i=0}^{\frac{n}{2}-1} \psi^{4ij+2i}\mathbf{v}_{2i+1}$. Then the above equations become

$$\hat{\mathbf{v}}_j = A_j + \psi^{2j+1}B_j \mod q,$$

and

$$\hat{\mathbf{v}}_{j+\frac{n}{2}} = A_j - \psi^{2j+1}B_j \mod q.$$

Here, A_j and B_j can be obtained as $\frac{n}{2}$ points NTT. It should be noted that the process can be repeated for all the coefficients if n is of exponent-of-two. The above two equar=tions are together called the *CT butterfly.*

5.2 Gentleman-Sande (GS) Algorithm for Fast-INTT :

Neglecting the weight $\frac{1}{n}$ in the equation (3),

$$\begin{aligned}
\mathbf{v}_i &= \sum_{j=0}^{n-1} \psi^{-(2i+1)j} \hat{\mathbf{v}}_j \mod q \\
&= \left[\sum_{j=0}^{\frac{n}{2}-1} \psi^{-(2i+1)j} \hat{\mathbf{v}}_j + \sum_{j=\frac{n}{2}}^{n-1} \psi^{-(2i+1)(j+\frac{n}{2})} \hat{\mathbf{v}}_{j+\frac{n}{2}} \right] \mod q \\
&= \psi^{-i} \left[\sum_{j=0}^{\frac{n}{2}-1} \psi^{-2ij} \hat{\mathbf{v}}_j + \sum_{j=\frac{n}{2}}^{n-1} \psi^{-2i(j+\frac{n}{2})} \hat{\mathbf{v}}_{j+\frac{n}{2}} \right] \mod q.
\end{aligned}$$

Next, using the symmetry and periodicity of ψ^{-1}, we get for the even term :

$$\begin{aligned}
\mathbf{v}_{2i} &= \psi^{-2i} \left[\sum_{j=0}^{\frac{n}{2}-1} \psi^{-4ij} \hat{\mathbf{v}}_j + \sum_{j=\frac{n}{2}}^{n-1} \psi^{-4i(j+\frac{n}{2})} \hat{\mathbf{v}}_{j+\frac{n}{2}} \right] \mod q \\
&= \psi^{-2i} \sum_{j=0}^{\frac{n}{2}-1} \left[\hat{\mathbf{v}}_j + \hat{\mathbf{v}}_{j+\frac{n}{2}} \right] \psi^{-4ij} \mod q.
\end{aligned}$$

It is easy to check that using the same derivation for the odd term gives :

$$\mathbf{v}_{2i+1} = \psi^{-2i} \sum_{j=0}^{\frac{n}{2}-1} \left[\hat{\mathbf{v}}_j - \hat{\mathbf{v}}_{j+\frac{n}{2}} \right] \psi^{-4ij} \mod q.$$

Let $A_i = \sum_{j=0}^{\frac{n}{2}-1} \hat{\mathbf{v}}_j \psi^{-4ij}$ and $B_i = \sum_{j=0}^{\frac{n}{2}-1} \hat{\mathbf{v}}_{j+\frac{n}{2}} \psi^{-4ij}$. Then we have,

$$\mathbf{v}_{2i} = (A_i + B_i)\, \psi^{-2i} \mod q$$

and

$$\mathbf{v}_{2i+1} = (A_i - B_i)\, \psi^{-2i} \mod q.$$

Above, A_i and B_i can be obtained as $\frac{n}{2}$ points INTT. It is clear that the process can be repeated for all the coefficients if n is exponent-of-two. The above two equations are together known as the *GS butterfly*.

To carry out polynomial multiplication, one can use CT butterflies to transform both inputs to the NTT domain. Then element-wise multiplication is performed to achieve the outputs. The outcome is inverted back using GS butterflies performing INTT. The butterflies play a major role in reducing the complexity of the polynomial multiplication from quadratic to quasilinear. The larger the degree of the polynomial, the greater the speed and minimal cost [10].

5.3 Normal and Bit-Reversed Order

Definition 10. *Let n be an exponent of 2, and b be a non-negative integer such that $b < n$. The bit-reversal of b is defined as :*

$$brv_n \left(b_{\log n-1} 2^{\log n-1} + \cdots + b_1 2 + b_0 \right) = b_0 2^{\log n-1} + \cdots + b_{\log n-2} 2 + b_{\log n-1},$$

where b_i is the i-th bit of the binary expansion of b [11].

The input of CT Butterfly is in Normal Order (NO) and the output is in Bit-Reversed Order (BO). To the contrary, the input of GS Butterfly is in BO, and the output in NO. However, one can reconfigure the CT butterfly to have BO-input and NO-output, and GS butterfly to have NO-input and BO-output. Usually normal order as NTT input is called decimation in time, whereas bit-reversed order input is called decimation in frequency [12].

6 NTT based multiplication

We describe NTT-based multiplication with an example explaining the reduction of the computation complexity of the polynomial multiplication.
Let q be prime such that $q \equiv 1 \mod 2n$. So, $2n$ divides the order $q-1$ of the cyclic group $\mathbb{Z}_q^\times$ (Unit group of $\mathbb{Z}_q$). Thus, $\mathbb{Z}_q$ contains n primitive $2n$-th roots of unity ψ^i, where $i = 1, 3, \cdots, 2n-1$. It follows that $x^n + 1$ factors into linear polynomials $x - \psi^i$ over $\mathbb{Z}_q$. In fact, the Chinese Remainder Theorem states that the natural ring homomorphism

$$f \mapsto \left(f(\psi), f(\psi^3), \cdots, f(\psi^{2n-1})\right) : \mathbb{Z}_q[x]/(x^n+1) \to \prod_i \mathbb{Z}_q[x]/(x-\psi^i)$$

is an isomorphism. The NTT computes this isomorphism and one can write NTT as a mapping from R_q to $\mathbb{Z}_q^n$. Then the product fg of two polynomials $f, g \in R_q$ can be calculated as

$$fg = INTT\left(NTT(f)NTT(g)\right).$$

This method involves two NTTs, one INTT and the pointwise multiplication in $\mathbb{Z}_q^n$. However, one can save NTTs. For example,

$$\sum_{i=1}^{t} f_i g_i = \sum_{i=1}^{t} INTT\left(NTT(f_i)NTT(g_i)\right) = INTT\left(\sum_{i=1}^{t} NTT(f_i)NTT(g_i)\right) \tag{5}$$

These sums of products of polynomials need to be computed in the matrix vector multiplication of schemes relying on the Module-LWE problem. As an example, we consider the Dilithium signature scheme with the parameters of the highest level security where the matrix A has dimensions 6×5. One can save 30 NTTs by sampling directly in the NTT representation, as A is sampled uniformly random. Then the vector needs to be transformed only once, saving another 25 NTTs. Now, we need only 6 INTTs instead of 30 INTTs, because of the linearity of NTT. So, instead of 90 NTTs, one needs only 11, 5 NTTs to transform the vector and 6 INTTs.

7 Conclusion

Achieving the quasilinear computational complexity from quadratic in the case of multiplication of polynomials with very high degrees is quite a big challenge. This has been achieved by NTTs. $O(n^2)$ time complexity is usually achieved when an alogrithm involves nested loops, where the inner loop iterates through all or a major portion of the input for each iteration of outer loop. The growth of n^2 is very rapid for large n. This makes such algorithms inefficient for large datasets. Traditional cyclic and negacyclic convolutions have $O(n^2)$ complexity. However, NTTs achieve $O(n \log n)$ time complexity due to its divide and conquer technique. In this type of algorithm, a problem is repeatedly divided into smaller subproblems, all of which are solved first and then combined, thereby bypassing the creation of nested loops, which would make the process relatively slower attaining quadratic complexity. Modern day cryptography, more precisely, lattice-based cryptography is heavily dependent on NTT for efficient polynomial multiplication, which is a core operation for these algorithms. NTTs are the sole reason to get it done in $O(n \log n)$ time instead of $O(n^2)$. NTTs are widely used in PQC algorithms such as Kyber and Falcon among others.

References

[1] A. Satriawan, R. Mareta, H. Lee, *A Complete Beginner Guide to the Number Theoretic Transform (NTT),* Cryptology ePrint Archive, Paper 2024/585 (2024).

[2] G. Seiler, *Faster AVX2 optimized NTT multiplication for Ring-LWE lattice cryptography,* Cryptology ePrint Archive, Paper 2018/039 (2018).

[3] R.C. Agarwal, C.S. Burrus, *Number theoretic transforms to implement fast digital convolution,* Proceedings of the IEEE 63(4), 550-560 (1975).

[4] H. Nussbaumer, *Fast polynomial transform algorithms for digital convolution,* IEEE Transactions on Acoustics, Speech, and Signal Processing 28(2), 205–215 (1980).

[5] A. Schonhage, *Schnelle multiplikation grosser zahlen,* Computing 7, 281–292 (1971).

[6] J.M. Pollard, *The fast fourier transform in a finite field,* Mathematics of computation 25(114), 365–374 (1971).

[7] V. Dimitrov, T. Cookley, B. Donevsky, *Generalized fermat-mersenne number theoretic transform,* IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing 41(2), 133–139 (1994).

[8] J.W. Cooley, J.W. Tukey, *An algorithm for the machine calculation of complex fourier series,* Mathematics of computation 19(90), 297–301 (1965).

[9] W.M. Gentleman, G. Sande, *Fast fourier transforms: for fun and profit,* Proceedings of the November 7-10, 1966, Fall Joint Computer Conference, pp. 563–578 (1966).

[10] P. Heckbert, *Fourier transforms and the fast fourier transform (fft) algorithm,* Computer Graphics 2, 15–463 (1995).

[11] Z. Liang, Y. Zhao, *Number theoretic transform and its applications in lattice-based cryptosystems: A survey,* arXiv preprint arXiv:2211.13546 (2022).

[12] A. Saidi, *Decimation-in-time-frequency fft algorithm,* Proceedings of ICASSP'94. IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 3, p. 453 (1994).